**CHAPTER 1**

**INTRODUCTION**

The Book Bank system is a user-friendly android application for library to manage the book issuing process and to take proper maintenance of the books. The Students can retrieve information about the available books and can search the books they want.

This designed for mainly two types of users. They are Students and Librarian. It lends books to member, who is registered in the system. Also, it handles the purchase of new titles for the Book Bank. Popular titles are brought into multiple copies. Old books and magazines are removed when they are out or date or poor in condition. A member can reserve a book or magazine that is not currently available in the book bank, so that when it is returned or purchased by the book bank, that person is notified. The Librarian can easily add details about the books.

When the orders are arrived then he/she will be notified by the app upon arrival. Then, they can go to the library to send the books.He or she can pay the price for the actual book or can rent it out.

Also they will be notified about the due date for the payment and the return for the book.When he/she forgets to return the book at time the due and the fine will be calculated along with the receipt and shown in the app itself.

**CHAPTER 2**

**LANGUAGE CHOICE**

Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically, for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

ART and Dalvik are compatible runtimes running Dex bytecode, so apps developed for Dalvik should work when running with ART. However, some techniques that work on Dalvik do not work on ART.

**ART Features**:

ART introduces ahead-of-time (AOT) compilation, which can improve app performance. ART also has tighter install-time verification than Dalvik.

At install time, ART compiles apps using the on-device dex2oat tool. This utility accepts DEX files as input and generates a compiled app executable for the target device. The utility should be able to compile all valid DEX files without difficulty. However, some post-processing tools produce invalid files that may be tolerated by Dalvik but cannot be compiled by ART.

**2.1 JAVA**

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Meanwhile, others have developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Class path (standard libraries), and Iced Tea-Web (browser plugin for applets).

**Advantages of using JAVA**:

*Object oriented*- Everything is considered to be an “object which possess some state, behavior and all the operations are performed using these objects.

*Dynamic state* – Ability to adapt to an evolving environment which supports dynamic memory allocation due to which memory wastage is reduced and performance of application is increased.

*Distributed* – Java provides a feature which helps to create distributed applications.

*Robust*- Java has a strong memory management system. It helps in eliminating error as it checks the code during compile and runtime.

**2.2 The Android platform:**

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019. The core Android source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License. This has allowed variants of Android to be developed on a range of other electronics, such as game consoles, digital cameras, PCs and others, each with a specialized user interface. Some well-known derivatives include Android TV for televisions and Wear OS for wearables, both developed by Google.

Android's source code has been used as the basis of different ecosystems, most notably that of Google which is associated with a suite of proprietary software called Google Mobile Services (GMS), that frequently comes pre-installed on said devices. This includes core apps such as Gmail, the digital distribution platform Google Play and associated Google Play Services development platform, and usually apps such as the Google Chrome web browser. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google. Other competing Android ecosystems include Amazon.com's Fire OS, or LineageOS. Software distribution is generally offered through proprietary application stores like Google Play Store or Samsung Galaxy Store, or open source platforms like Aptoide or F-Droid, which use software packages in the APK format.

**2.3 SQLite**

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen ().

SQLite is a compact library. With all features enabled, the library size can be less than 600KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function in lining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

SQLite is very carefully tested prior to every release and has a reputation for being very reliable. Most of the SQLite source code is devoted purely to testing and verification. An automated test suite runs millions and millions of test cases involving hundreds of millions of individual SQL statements and achieves 100% branch test coverage. SQLite responds gracefully to memory allocation failures and disk I/O errors. Transactions are ACID even if interrupted by system crashes or power failures. All of this is verified by the automated tests using special test harnesses which simulate system failures. Of course, even with all this testing, there are still bugs. But unlike some similar projects (especially commercial competitors) SQLite is open and honest about all bugs and provides bugs lists and minute-by-minute chronologies of code changes.

The SQLite code base is supported by an international team of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the published interface spec, SQL syntax, and database file format. The source code is absolutely free to anybody who wants it, but professional support is also available.

**CHAPTER 3**

**SYSTEM ANALYSIS**

System analysis is the process of gathering and interpreting facts, diagnosing the problem and using the information to recommend improvements on the system. System analysis is problem solving activity that requires intensive communication between the system users and the system developers. This is an important phase of any system development process. The system is studied in the minutest detail and analyzed.

The system analyst plays the role of an interrogator and dwells into the working of the present system. The system is viewed as a whole and inputs to the system are identified. Outputs from system are traced through various phases of the processing of the inputs. The process of the system analysis is largely concerned with determining, developing and agreeing users’ requirements perceived in business term. The system analysis phase is the prime opportunity to communicate the users has conceived the join understanding to determine the system features.

A detailed study of these processes must be made by the various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions.

Now this existing system is subjected to close study and the problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as a proposal. The proposal is the weighted with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. This is a repeating process that ends as soon as the user is satisfied with this proposal.

**FEASIBILITY STUDY**

All projects are feasible when given unlimited resources and indefinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. A feasible study is not warranted for system in which economic justification is observed, technical risk is low, few legal problems are expected and no reasonable alternatives exist. An alternative is made of whether the identified user needs may be satisfied using our recent software and hardware technologies. The study will decide if the proposed system will be cost effective, from the business point of view and it can be developed in the existing budgetary constraints. The feasibility study should be relatively sharp and quick. The gesture should inform the decision of whether to go ahead with a more detailed analysis. Feasibility study may be documented as a separated report to higher officials of the top-level management and can be included as appendices to the system specification. Feasibility and risk analysis is detailed in many worries. If there is more project risk then the feasibility of producing the quality software is reduced. The study is done in three phases.

* Feasibility
* Operational Technical Feasibility
* Economic Feasibility

**A. Operational Feasibility**

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, the test of feasibility asks if the system will work when it is developed and installed. Are there any major barriers to implementations? Is there sufficient support for the project from the management? Are current business methods acceptable to the users? Have the users been involved in the planning and development of the project? Will the proposed system cause any harm?

The purpose of the operational feasibility study is to determine whether the new system will be used if it is developed and installed and whether there will be resistance from users that will be resistance from users that will undermine the possible application benefit.

There was no difficulty in implemented the software and proposed system is so effective, user friendly, functionally reliable so that the users in the company will find that the new system reduces the hard steps.

**B. Technical feasibility**

A study of function, performances and constraints and improve the ability to create an acceptable system. Technical feasibility is frequently the most difficult area to achieve at the stage of product engineering process. Considering that are normally associated with the technical feasibility includes

* Development risk
* Resource availability
* Technology

Technical feasibility study deals with the hardware as well as the software requirements. The scope was whether the work for the project is done with the current requirements and the existing software technology has to be examined in the feasibility study. The outcome was found to be positive.

**C. Economic feasibility**

The cost evaluation is weighted against ultimate income or benefit derived from

the developed system or project. Economic justification is generally the "Button line “consideration that includes cost benefit analysis, long term corporate income strategies, impact on other profit centers or products, cost of resources needed for development and potential market growth. When compared to the advantage obtained from implementing the system its cost is affordable. Also, the system is designed to meet the modification required in the future. Therefore, most of the modifications can be done without much re-work.

**REQUIREMENT ANALYSIS**

The following steps that give the detailed information of the need of proposed system are:

*Performance:*

During past several decades, the records are supposed to be manually handled for all activities. The manual handling of the record is time consuming and highly prone to error. To improve the performance of the Book bank system, the computerized system is to be undertaken. This project is fully computerized and user friendly even that any students can register and use the app.

*Efficiency:*

The basic need of this app is efficiency. The app should be efficient so that whenever a new user submits his/her details the app is updated automatically.

*Control:*

The complete control of the project is under the hands of authorized person who has the password to access this project and illegal access is not supposed to deal with. All the control is under the administrator and the other members have the rights to just seethe records not to change any transaction or entry.

*Security:*

Security is the main criteria for the proposed system since illegal access may corrupt the database. So, security has to be given in this project.

**3.1 PRESENT SYSTEM**

The Existing system is full of manual process. Due to the book bank system is increasing day by day gradually. There is need for automation to compensate the work load by providing faster service.

The previous system works by the student searching for book at the library sections. If book is available, he/she can pay the money to buy or they can rent the book out when the book is not available in library, they can request the librarian for that book by writing a request letter. This will take at least take a week to be received by the vendor. Sometimes it may be lost on the process.

LIMITATIONS OF PRESENT SYSTEM:

* There is a high possibility for human error in this Existing system.
* Delay will be higher when on busy hours like college opening.
* Manual work is high and requires more people for proper functionality.

**3.2 PROPOSED SYSTEM**

The proposed system works by automating the work that don’t requires any specific knowledge like queuing the request, gathering information about the student. This system uses the database to store the information and retrieve when needed. So, it reduces need of papers. The student just opens the application and can search for the books they want. When the book they want is not available they can send and request directly through the application via the request function which will send the book details to the librarian. Via his application. Once the librarian verifies the request, they can add it to the request queue and can order as Bundle. To the vendor

Once the book is received by the librarian, he /she will send the students via the application indicating the arrival of the book to the library. Then the students can get the book by showing the proof. So, half of the manual work will be eliminated by automation. And the students need not to check the library every day for the arrival of the books.

**BENEFITS OF PROPOSED SYSTEM**

* + - Time will be saved for both student & librarian.
    - Books can be searched anywhere, anytime.
    - More reliable.
    - Less human error in the process.
    - More adaptive than manual system/existing system.

**3.3 SOFTWARE REQUIREMENT SPECIFICATION**

**3.3.1 INTRODUCTION**

A system requirements specification is a document where the requirements of a system is planned to be developed are listed. Requirement analysis is a software engineering task that brings the gap between the system and software designs. Requirement analysis enables the system engineer to specify software functional performance indicates software’s interface with other system elements and establishes constraints that the software must meet. Finally, requirement specification provides the developer and the customer with the means to assess quality once the software is built. Requirement specification places a great deal of responsibility on the system analyst, because the point shows up later in the characteristics of the new system. Requirements analysis is critical to the success of a project.

A main purpose of software requirement specification is the clear definition and specification of functionality and of the software product. It allows the developer to be carried out, performance level to be obtained and corresponding interface to be established. The following Requirements are only the minimal requirements to run this utility more successfully and efficiently. There should be sufficient memory and software tools for efficient processing.

**3.3.1.1 Purpose**

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to users. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops.

In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality.

**3.3.1.2 Scope**

This SRS is also aimed at specifying requirements of software to be developed to assist general hospitals, clinics and other health centers to manage the donor registration and user maintenance. The SRS can be used as a model for defining a project specific standard.

**3.3.1.3 Reference**

* developer.android.com

**3.3.1.4 Overview**

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product.

**3.3.2 GENERAL DESCRIPTION**

The Book bank system is developed mainly for colleges and other institutions to manage the book issuing and maintenance. The students can retrieve information about the current available books via the app. Students can also register via the app itself. The Librarian will receive the request list for the books.

**3.3.2.1 Product Perspective**

Current system uses the manual file system which is also known as a simple database. Ledgers and logbooks are wisely used to record the information and events of the book issuing. More over the current system is only for retrieving information about students. The proposed system is the enhancement for the current system. It can be a single point of access for. students and librarian. The students can be registered just by clicking the register button in front of their mobile at home and save their time.

**3.3.2.2 Product function**

This subsection contains the requirements for book bank system. These requirements are organized by the features discussed in the vision document. Features from vision are then refined into use case diagrams and to sequence diagram to best capture the functional requirements of the system. All these functional requirements can be traced using tractability matrix.

**3.3.2.2.1 Provide personalized profiles**

The system shall allow the user to register their profile.

The system shall allow the user to get a user id and password during registration.

The system shall allow the user to search the book they want.

The system shall allow user to send request.

The system shall allow user to cancel the registration.

The system shall allow user to logout.

**3.3.2.2.2 Provide User and Librarian support**

The system shall allow user to know about app and its operation.

The system allows students to get support from the librarian.

**3.3.2.2.3 Provide Search facility**

The system shall enable user to enter the search text on the textbox.

The system shall display all the matching books based on the search.

The system shall enable user to navigate between the search results.

The system shall notify the user when no matching book is found.

**3.3.2.3 User characteristics**

The specific users for this system are:

* *Student* - They are the people who desire to obtain the books and submit the information to the database.
* *Librarian* - He has the certain privileges to add the books and to approval of the reservation of books.

**3.3.2.4 General Constraints**

* The Students require a mobile to submit their information.
* Although the security is given high importance, there is always a chance of intrusion in the mobile app which requires constant monitoring.
* The Students has to be careful while submitting the information. Much care is required.

**3.3.3 FUNCTIONAL REQUIREMENT**

**3.3.3.1 Functional requirements for user profile creation**

**3.3.3.1.1 Introduction**

This screen provides an interface to input information specific to both users. i.e. Students and librarian. then, the values are stored in the local database.

**3.3.3.1.2 Input**

Each user can enter their personal details in the corresponding registration forms. Their Selection and entries are done using keypad.

**3.3.3.1.3 Processing**

When the user enters the inputs and click on the submit button provided in these pages, the system responds by displaying an activity containing predefined text along with the allotted user id.

**3.3.3.1.4 Output**

The activity will contain a predefined toast message which will show up on successful registration.

**3.3.3.2 Functional requirements for support to students and librarians**

**3.3.3.2.1 Introduction**

This screen provides an interface to the students for getting helps related to the system. It also provides FAQ facilities to these users. It also helps in providing a communication between the students and the librarian/Administrator.

**3.3.3.2.2 Input**

The students can post their request messages to the librarian. The response can be provided as notification.

**3.3.3.2.3 Processing**

The request posted by the users are stored in the database and they are displayed in the list view.

**3.3.3.2.4 Output**

A page containing a predefined text regarding the request given by the user is displayed. Also, the replies or responses from other users can be displayed in the corresponding activity.

**3.3.3.3 Functional requirements for searching**

**3.3.3.3.1 Introduction**

This screen provides an interface to the users for searching for the donors or acceptors based on certain search keys. The searches can be filtered according to the search keys provided by the users

**3.3.3.3.2 Input**

Inputs can be given by the students via the recycler view textbox.

**3.3.3.3.3 Processing**

The data will be processed by predefined queries. The search is processed according to the search key by checking the matching parameters.

**3.3.3.3.4 Output**

The search result is displayed in a recycler view or list view.

**3.3.4.1 User interface**

The UI is designed using only the native libs and assets. So, it will be compatible and future proof.

**3.3.4.2 H/W interface**

Processor: MT6580 or equivalent Quadcore processor SoC.

RAM:1 GB or Higher.

**3.3.4.3 Software Requirement**

The software shall be designed to run on the following platform:

Operating system:

Minimum – Android 5.1(Lollipop)

Targeted - Android 9.0(Pie)

**3.3.5 Performance Requirement**

**Reliability**

This deals with how reliable the application being built is. The application gives accurate results and hence is highly reliable.

**Efficiency**

Every application consumes some resources if it is to perform and complete its task. The application being built must consume as less memory as possible.

**Usability**

This requirement ensures that the application is user friendly and any person with or without operational knowledge must be able to use the application with ease. The terminology used in this application is simple and easy to understand.

**Maintainability**

The use of applications generally spans over a large period of time and hence their performance deteriorates as time progresses. Maintainability ensures that the application must be maintainable delivering the same performance level whenever used.

**Flexibility**

As time progresses, the expectations from the application may change since user requirements change with time. Hence, the application must be built in a way to adapt new changes.

**3.3.6 Design constraints**

The device must have an dpi of 225 or higher to render the icons and text properly.

**CHAPTER 4**

**SYSTEM SPECIFICATION**

To understand the nature of the software to be built, the software engineer must understand the information domain of the software, as well as the required function, performance, and interface. Requirement definition is the activity of translating information gathered during analysis activity into a document that defines a set of requirements. Requirement definition is the high abstract description of requirements. Requirements may be functional or non-functional.

**4.1 HARDWARE SPECIFICATION**

The decision to acquire computer hardware or software must be handled in the same way as any other business decision. The variety of sizes and types of computing resources available puts a burden on the analyst who must select suitable hardware, software or services and advice the top management accordingly.

Today, selecting a system is a serious and time-consuming business. The time spent on the selection process is a function of the application and whether the system is a basic microcomputer or a mainframe. In either case, planning system selection and acquiring experienced help were necessary payoff in the long run.

The selection process should be viewed as a project and a project team should be formed with the help of management.

Hardware Specification for Development, Implementation

Processor: i5(8th Gen or higher)

RAM: 8GB or higher

HDD: 250GB

**Implementation Configuration**

Processor: MT6580 or equivalent Quadcore processor SoC.

RAM:1 GB or Higher.

**4.2 SOFTWARE SPECIFICATION**

The software requirements explain the software components that we need to develop our project. It is selected such a way that it reduces our work and easy to implement the project anywhere.

Minimum Software’s Used

Operating system: Windows 10 1807

Front end: Java & XML for design

Backend: SQLite

Android studio 3.6

The project is done using android studio.

**Android Studio**

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google’s preferred language for Android app development. Java is still supported, as is C++.

The following features are provided in the current stable version:

**Gradle-based build support**

Android-specific refactoring and quick fixes

Lint tools to catch performance, usability, version compatibility and other problems

ProGuard integration and app-signing capabilities

Template-based wizards to create common Android designs and components

A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations.

Support for building Android Wear apps

Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine.

Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

**Android SDK**

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows 7 or later. As of March 2015, the SDK is not available on Android itself, but software development is possible by using specialized Android applications.

Until around the end of 2014, the officially-supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others, but Google made it clear that ADT was officially deprecated since the end of 2015 to focus on Android Studio as the official Android IDE. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand-in-hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

**Android SDK Platform Tools**

The Android SDK Platform Tools are a separately downloadable subset of the full SDK, consisting of command-line tools such as adb and fastboot.

**CHAPTER 5**

**SYSTEM DESIGN**

This is the phase in which the details of the system selected in the study phase is designed. Software design is the preliminary step and is also building block of software engineering. The efficiency software is promoted through design phase. The design phase begins when the requirement specification document for the software to be developed is available. Design is the first step to moving from the problem domain to the solution domain. Design is essentially the bridge between requirement specification and the final solution for satisfying the requirements.

5.1 INPUT DESIGN

It is the part of the overall system design. The input methods an be broadly classified into batch and online. Internal control must be established for monitoring number of input and for ensuring that the data is valid. the basic step involved in the system design are

* Review input requirement.
* Decide how the input data flow will be maintained.
* Design the source document.
* Prototype online input screens.

The quality of the system input determines the quality of the system output. Input specification describes the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce results from the accurate data, or they can result in production the input design also determines whether the user can interact efficiently with the system.

The major inputs screens of librarian module are

* Add books name.
* Add Author name.
* Add edition.
* Add publisher.

The major inputs screens of student’s module are

* Submit login form.
* Search books.
* Request Books.

**5.2 OUTPUT DESIGN**

It is the part of overall system design. The goal of the output design is to capture the output and get the data into format suitable for the computer. Data flow diagram identifies the data tone captured and the output to the system. One of the important features of an information system for users is the output it produces. Output is the information delivered to the users through the information system. Without quality output the entire system appears to be unnecessary that users will avoid using it. Users generally merit the system solely by its output in order to create the most useful output possible. One works closely with the user through an interactive process, until the result is considered to be satisfactory.

**CHAPTER 6**

**DETAILED DESIGN**

Detailed design of a project deals with the entire functionality it handles. A well-defined detailed description includes major functional components in that application along with sub modules included and table handled. It shows actual data flow within the system and how each function handles the data.

**6.1 Module**

SQLite provides 5 functionalities.

**6.1.1 - Provide personalized profiles**

The system shall allow the user to register their profile. Each user gets an user id and password during registration. Users can select the books they want by searching using the search bar.

**Tables used**

* UserTable
* BookTable

**6.1.2 User Support**

Users can view the book details and request the system for the books they want.

**Tables used**

* UserTable

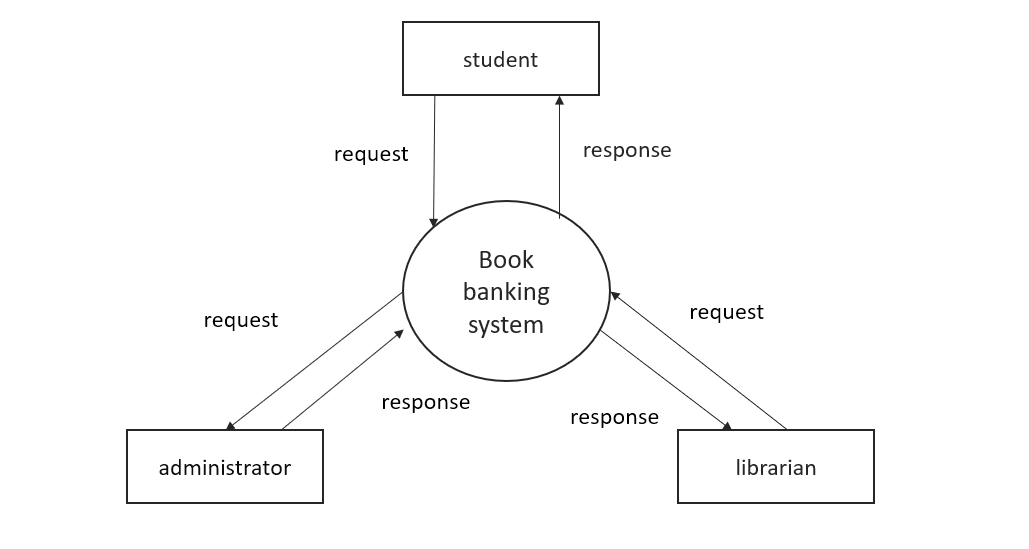
6.1.3 Search and selection books

The primary aim of this module is to provide search facilities for the users based on various criteria. The user can search the database for the books using the query.

**Tables used**

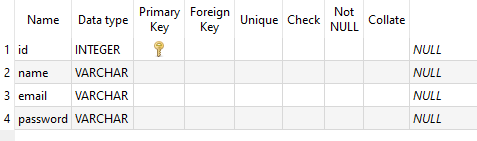
* BookTable

**6.2 DATA FLOW DIAGRAMS**

****

**6.3 DATABASE DESIGN**

**UserTable**

****

**BookTable**

****

**CHAPTER 7**

**TESTING**

**7.1 SYSTEM TESTING**

System testing is a critical element of quality assurance and represents the ultimate review of analysis, design and coding. Test case design focuses on a set of techniques for the creation of test cases that meet overall testing objective. When a system is developed it is hoped that it performs properly. The main purpose of testing an information system is to find the errors and correct them. The scope of system testing should include both manual and computerized operations. System testing is a comprehensive evaluation of the programs, manual procedures, computer operations and controls.

System testing is the process of checking whether the developed system is working according to the objective and requirement. All testing is to be conducted in accordance to the test conditions specified earlier. This will ensure that the test coverage meets the requirements and that testing is done in a systematic manner.

**7.2 TESTING PROCESS**

The strategy for system testing integrates system test cases and design techniques into a well – planned series of steps that result in the successful construction of software. The testing must co-operate with test planning, test case design, test execution and resultant data collection and re – evaluation. A strategy for software testing must accommodate a low-level test and that are necessary to verify that a small code segment has correctly implemented as well as high level text that validate major system functions against user requirements. Software testing is a critical element of software quality assurance and represents the ultimate review specification, design and coding. A series of testing is performed for the proposed system before the system is ready for acceptance testing.

**7.3 UNIT TESTING**

In unit testing, different modules are tested against the specification produced during the design of modules. Unit testing is essential for verification during the coding phase. The aim is to test the internal logic of modules. The tests are carried out during the programming stage itself.

**7.4 INTEGRATION TESTING**

Integration testing focuses on the design and construction of the software architecture. The data can be lost across the interface or one module can cause an adverse effect on another. The sub functions when combined may not produce the major function. The integration testing is a systematic technique for the program structure, while at the same conducting the test to uncover errors associated with the interface.

**7.5 VALIDATION TESTING**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been recovered and corrected and final series of a software test – validation test begin. Validation testing can be defined in many ways but simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer.

**7.6 SYSTEM TESTING**

System testing is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements. System testing falls within category of a Black-Box testing and such as, should require no knowledge of the inner design of the code or logic. Alpha testing and Beta Testing are sub- categories of system testing. As a rule, system testing takes, as its input, all of the “integrated” software components that have successfully passed the integration testing.

**7. 7 WHITEBOX TESTING**

White box testing, glass box testing or structural testing is used in computer programming, software engineering and software testing to check that the outputs of a program, given inputs, conform to the structural specification of the program. The term White Box (glass box) indicates that the testing is done with the knowledge of the code used to execute certain functionality. For this reason, a programmer is usually requiring to perform white box tests. Often multiple programmers will write based on certain code, so as to gain perspectives in possible outcomes.

**7.8 BLACK BOX TESTING**

Black Box testing, concrete box or functional testing is used in computer programming, software engineering and software testing to check that the outputs of a program, given certain inputs, conform to the functional specification of the program. The term black box indicates the internal implementation of the program being executed is not examined by the tester.

**USER ACCEPTANCE TESTING**

User acceptance testing of the system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly in touch with the perspective system at the time of development and making range whenever required. This is done with regard to the input screen, design and the output screen design.

**SYSTEM IMPLEMENTATION**

System implementation is the final phase that is putting the utility into action. Implementation is the state in the project where theoretical design turned into working system. The most crucial stages involve achieving a new successful system and giving confidence in new system that it will work efficiently and effectively. The system is implemented only after thorough checking is done and if it is working in accordance with the specifications.

It involves careful planning, investigation of the current system and constraints on implementation, design of methods to achieve. Two checking are done and if they are found working according to the specification, major task of repairing the implementation are educating, training the users.

**IMPLEMENTATION PROCEDURES**

The major implementation procedures are,

* Test plan
* Training
* Equipment installation
* Conversation

**TEST PLANS**

The implementation of a computer – based system requires that test data be prepared and that the system and its elements be tested in a planned, structured manner. The computer program component is a major sub system of the computer - based information system, and particular attention should be given to the testing of the system element as it is developed.

**TRAINING**

The purpose of the training is to ensure that all the personnel who are to be associated with the computer – based administration system possess the necessary knowledge skills. Operating, programming, and user personnel are trained using reference manuals as training aids.

**EQUIPMENT INSTALLATION**

As the name suggests, it deals with installing the equipment. This part deals with actually the hardware installation.

**CONVERSATION**

Conversation is the process of performing all of the operations that result directly in the turnover of the new system to the user. Conversation has 2 parts;

a. The creation of a conversation plan at the start of the development phase and the implementation of this plan throughout the development phase.

b. The creation of a system changes over plan the end of the development phase and the implementation of the plan at the beginning of the operation phase.

**REFINEMENT MADE OF FEEDBACK**

Most popular methods of exchanging data in the IT world can be viewed from 2 perspectives; how companies format data they want to exchange and how they transmit that. To accomplish these needs of today’s business, developers require a technology that can be integrated across much different types of systems and that is exactly what web services do.

**CHAPTER 8**

**CONCLUSION**

The Book Bank system is an android application which is based on library book issuing system. It can be used by any students. The best feature of the application is that we can search for the available books even without going to the library by the application.

Within a few clicks the system provides the details about the book. Another thing is that it provides user login and registration for keeping record of users.

This app is built with Google’s material guidelines and properly made every component to material theme. So, we can experience smooth app when navigating. This app also doesn’t need internet connection when using for demo purposes. The app is constantly updated with data about the books at the library and the books can be searched anywhere at any time at the comfort of the user. More complex systems such as RecyclerView are used here for Search functionality. There are immense possibilities for this system.

**CHAPTER 9**

**ANNEXURE**

**9.1 CODING**

**AndroidManifest.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.r2017.bookbank"**>  
  
 <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />  
 <**uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"** />  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:fullBackupContent="@xml/backup\_descriptor"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"** />  
 <**activity android:name=".RequestActivity"** />  
 <**activity android:name=".LibrarianEntry"** />  
 <**activity android:name=".HomeActivity"** />  
 <**activity android:name=".LibrarianLogin"** />  
 <**activity android:name=".DashboardActivity"** />  
 <**activity android:name=".RegisterActivity"** />  
 <**activity  
 android:name=".SplashActivity"  
 android:theme="@style/AppTheme.NoActionBar"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
  
 </**activity**>  
 </**application**>  
  
</**manifest**>

**Book.java**

**package** com.r2017.bookbank;  
  
**public class** Book {  
 **public int id**;  
 **private** String **bookName**, **authorName**;  
  
 Book() {  
  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 String getBookName() {  
 **return bookName**;  
 }  
  
 **void** setBookName(String bookName) {  
 **this**.**bookName** = bookName;  
 }  
  
 String getAuthorName() {  
 **return authorName**;  
 }  
  
 **void** setAuthorName(String authorName) {  
 **this**.**authorName** = authorName;  
 }  
}

**BookDatabase.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteQueryBuilder;  
  
**import** com.readystatesoftware.sqliteasset.SQLiteAssetHelper;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**class** BookDatabase **extends** SQLiteAssetHelper {  
  
 **private static final** String ***DB\_NAME*** = **"AndroidJSonDataBase"**;  
 **private static final int *DB\_VER*** = 1;  
  
  
 BookDatabase(Context context) {  
 **super**(context, ***DB\_NAME***, **null**, ***DB\_VER***);  
  
 }  
  
 List<Book> getBook() {  
 SQLiteDatabase db = getReadableDatabase();  
 SQLiteQueryBuilder qb = **new** SQLiteQueryBuilder();  
  
 String[] sqlSelect = {**"id"**, **"Book\_name"**, **"Author\_name"**};  
 String tableName = **"AndroidJSonTable"**;  
  
 qb.setTables(tableName);  
 Cursor cursor = qb.query(db, sqlSelect, **null**, **null**, **null**, **null**, **null**);  
 List<Book> result = **new** ArrayList<>();  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 Book book = **new** Book();  
 book.setId(cursor.getInt(cursor.getColumnIndex(**"id"**)));  
 book.setBookName(cursor.getString(cursor.getColumnIndex(**"Book\_name"**)));  
 book.setAuthorName(cursor.getString(cursor.getColumnIndex(**"Author\_name"**)));  
 result.add(book);  
 } **while** (cursor.moveToNext());  
 }  
 **return** result;  
 }  
  
 List<String> getNames() {  
 SQLiteDatabase db = getReadableDatabase();  
 SQLiteQueryBuilder qb = **new** SQLiteQueryBuilder();  
  
 String[] sqlSelect = {**"Book\_name"**};  
 String tableName = **"AndroidJSonTable"**;  
  
 qb.setTables(tableName);  
 Cursor cursor = qb.query(db, sqlSelect, **null**, **null**, **null**, **null**, **null**);  
 List<String> result = **new** ArrayList<>();  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 result.add(cursor.getString(cursor.getColumnIndex(**"Book\_name"**)));  
 } **while** (cursor.moveToNext());  
 }  
 **return** result;  
  
 }  
  
 List<Book> getBookByName(String name) {  
  
 SQLiteDatabase db = getReadableDatabase();  
 SQLiteQueryBuilder qb = **new** SQLiteQueryBuilder();  
  
 String[] sqlSelect = {**"id"**, **"Book\_name"**, **"Author\_name"**};  
 String tableName = **"AndroidJSonTable"**;  
  
 qb.setTables(tableName);  
 Cursor cursor = qb.query(db, sqlSelect, **"Book\_name LIKE ?"**, **new** String[]{**"%"** + name + **"%"**}, **null**, **null**, **null**);  
 List<Book> result = **new** ArrayList<>();  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 Book book = **new** Book();  
 book.setId(cursor.getInt(cursor.getColumnIndex(**"id"**)));  
 book.setBookName(cursor.getString(cursor.getColumnIndex(**"Book\_name"**)));  
 book.setAuthorName(cursor.getString(cursor.getColumnIndex(**"Author\_name"**)));  
 result.add(book);  
 } **while** (cursor.moveToNext());  
 }  
 **return** result;  
  
  
 }  
}

**DashboardActivity.java**

**package** com.r2017.bookbank;  
  
**import** android.annotation.SuppressLint;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** DashboardActivity **extends** AppCompatActivity {  
  
 String **NameHolder**;  
 TextView **Name**;  
 Button **LogOUT**, **GetStarted**;  
  
 @SuppressLint(**"SetTextI18n"**)  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_dashboard***);  
  
 **Name** = findViewById(R.id.***tvHello***);  
 **LogOUT** = findViewById(R.id.***logout***);  
 **GetStarted** = findViewById(R.id.***btnGetStarted***);  
  
 Intent intent = getIntent();  
  
 *// Receiving User Email Send By MainActivity.* **NameHolder** = intent.getStringExtra(MainActivity.***userName***);  
  
 *// Setting up received email to TextView.* **Name**.setText(**Name**.getText().toString() + **" "** + **NameHolder**);  
  
 *// Adding click listener to Log Out button.* **LogOUT**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *//Finishing current DashBoard activity on button click.* finish();  
  
 Toast.*makeText*(DashboardActivity.**this**, **"Logout Successful"**, Toast.***LENGTH\_SHORT***).show();  
  
 }  
 });  
 **GetStarted**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent1 = **new** Intent(DashboardActivity.**this**, HomeActivity.**class**);  
 startActivity(intent1);  
  
 }  
 }  
 );  
  
  
 }  
  
}

**HomeActivity.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.text.Editable;  
**import** android.text.TextWatcher;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** androidx.recyclerview.widget.LinearLayoutManager;  
**import** androidx.recyclerview.widget.RecyclerView;  
  
**import** com.mancj.materialsearchbar.MaterialSearchBar;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** HomeActivity **extends** AppCompatActivity {  
  
 RecyclerView **recyclerView**;  
 RecyclerView.LayoutManager **layoutManager**;  
 SearchAdapter **adapter**;  
 MaterialSearchBar **materialSearchBar**;  
 List<String> **suggestList** = **new** ArrayList<>();  
  
 BookDatabase **database**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_home***);  
  
 *//init view* **recyclerView** = findViewById(R.id.***recycler\_search***);  
 **layoutManager** = **new** LinearLayoutManager(**this**);  
 **recyclerView**.setLayoutManager(**layoutManager**);  
 **recyclerView**.setHasFixedSize(**true**);  
  
 **materialSearchBar** = findViewById(R.id.***search\_bar***);  
  
 *//init DB* **database** = **new** BookDatabase(**this**);  
  
 *//search bar* **materialSearchBar**.setHint(**"Search"**);  
 **materialSearchBar**.setCardViewElevation(10);  
 loadSuggestionList();  
 **materialSearchBar**.addTextChangeListener(**new** TextWatcher() {  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count, **int** after) {  
  
 }  
  
 @Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before, **int** count) {  
  
 List<String> suggest = **new** ArrayList<>();  
 **for** (String search : **suggestList**) {  
 **if** (search.toLowerCase().contains(**materialSearchBar**.getText().toLowerCase()))  
 suggest.add(search);  
 }  
 **materialSearchBar**.setLastSuggestions(suggest);  
  
 }  
  
 @Override  
 **public void** afterTextChanged(Editable s) {  
  
 }  
 });  
 **materialSearchBar**.setOnSearchActionListener(**new** MaterialSearchBar.OnSearchActionListener() {  
 @Override  
 **public void** onSearchStateChanged(**boolean** enabled) {  
 **if** (!enabled)  
 **recyclerView**.setAdapter(**adapter**);  
 }  
  
 @Override  
 **public void** onSearchConfirmed(CharSequence text) {  
 startSearch(text.toString());  
  
 }  
  
 @Override  
 **public void** onButtonClicked(**int** buttonCode) {  
  
 }  
 });  
  
 *//init adapter set all result* **adapter** = **new** SearchAdapter(**database**.getBook());  
 **recyclerView**.setAdapter(**adapter**);  
  
  
 }  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.***request***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(@NonNull MenuItem item) {  
 Intent intent3 = **new** Intent(HomeActivity.**this**, RequestActivity.**class**);  
 startActivity(intent3);  
 **return false**;  
 }  
  
 **private void** startSearch(String text) {  
 **adapter** = **new** SearchAdapter(**database**.getBookByName(text));  
 **recyclerView**.setAdapter(**adapter**);  
  
 }  
  
 **private void** loadSuggestionList() {  
  
 **suggestList** = **database**.getNames();  
 **materialSearchBar**.setLastSuggestions(**suggestList**);  
 }  
}

**LibrarianEntry.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.text.TextUtils;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** LibrarianEntry **extends** AppCompatActivity {  
  
 SQLiteDatabase **sqLiteDatabaseObj**;  
 EditText **editTextBookName**, **editTextAuthorName**;  
 String **BookHolder**, **AuthorHolder**, **SQLiteDataBaseQueryHolder**;  
 Button **EnterData**;  
 Boolean **EditTextEmptyHold**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_librarian\_entry***);  
  
 **EnterData** = findViewById(R.id.***btnAddBook***);  
 **editTextBookName** = findViewById(R.id.***etBookName***);  
 **editTextAuthorName** = findViewById(R.id.***etAuthorName***);  
  
 **EnterData**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
  
 SQLiteDataBaseBuild();  
  
 SQLiteTableBuild();  
  
 CheckEditTextStatus();  
  
 InsertDataIntoSQLiteDatabase();  
  
 EmptyEditTextAfterDataInsert();  
  
  
 }  
 });  
  
  
 }  
  
 **public void** SQLiteDataBaseBuild() {  
  
 **sqLiteDatabaseObj** = openOrCreateDatabase(**"AndroidJSonDataBase"**, Context.***MODE\_PRIVATE***, **null**);  
  
 }  
  
 **public void** SQLiteTableBuild() {  
  
 **sqLiteDatabaseObj**.execSQL(**"CREATE TABLE IF NOT EXISTS AndroidJSonTable(id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, Book\_name VARCHAR, Author\_name VARCHAR);"**);  
  
 }  
  
 **public void** CheckEditTextStatus() {  
  
 **BookHolder** = **editTextBookName**.getText().toString();  
 **AuthorHolder** = **editTextAuthorName**.getText().toString();  
  
 **EditTextEmptyHold** = !TextUtils.*isEmpty*(**BookHolder**) && !TextUtils.*isEmpty*(**AuthorHolder**);  
 }  
  
 **public void** InsertDataIntoSQLiteDatabase() {  
  
 **if** (**EditTextEmptyHold**) {  
  
 **SQLiteDataBaseQueryHolder** = **"INSERT INTO AndroidJSonTable (Book\_name,Author\_name) VALUES('"** + **BookHolder** + **"', '"** + **AuthorHolder** + **"');"**;  
  
 **sqLiteDatabaseObj**.execSQL(**SQLiteDataBaseQueryHolder**);  
  
 Toast.*makeText*(LibrarianEntry.**this**, **"Data Inserted Successfully"**, Toast.***LENGTH\_LONG***).show();  
  
 } **else** Toast.*makeText*(LibrarianEntry.**this**, **"Please Fill All The Required Fields."**, Toast.***LENGTH\_LONG***).show();  
  
 }  
  
 **public void** EmptyEditTextAfterDataInsert() {  
  
 **editTextBookName**.getText().clear();  
  
 **editTextAuthorName**.getText().clear();  
  
 }  
  
}

**LibrarianLogin.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** LibrarianLogin **extends** AppCompatActivity {  
  
 EditText **Name**;  
 EditText **Password**;  
 Button **Login**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_librarian\_login***);  
 **Name** = findViewById(R.id.***etLibUserName***);  
 **Password** = findViewById(R.id.***etLibPass***);  
 **Login** = findViewById(R.id.***btnLibLogin***);  
  
 **Login**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 LibrarianLogin.**this**.validate(**Name**.getText().toString(), **Password**.getText().toString());  
 }  
 });  
  
  
 }  
  
 **public void** validate(String id, String pass) {  
 **if** ((id.equals(**"admin"**)) && (pass.equals(**"aec@123"**))) {  
 Intent intent = **new** Intent(LibrarianLogin.**this**, LibrarianEntry.**class**);  
 Toast.*makeText*(**this**, **"Login Successful"**, Toast.***LENGTH\_SHORT***).show();  
 startActivity(intent);  
  
 } **else** {  
 Toast.*makeText*(**this**, **"Username or password incorrect"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
  
}

**MainActivity.java**

**package** com.r2017.bookbank;  
  
  
**import** android.content.Intent;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.text.TextUtils;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
  
 **public static final** String ***userName*** = **""**;  
 Button **LogInButton**, **RegisterButton**;  
 EditText **Email**, **Password**;  
 String **EmailHolder**, **PasswordHolder**;  
 Boolean **EditTextEmptyHolder**;  
 SQLiteDatabase **sqLiteDatabaseObj**;  
 SQLiteHelper **sqLiteHelper**;  
 Cursor **cursor**;  
 String **TempPassword** = **"NOT\_FOUND"**;  
 String **UserName**;  
  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
  
 setContentView(R.layout.***activity\_main***);  
  
 **LogInButton** = findViewById(R.id.***btnLogin***);  
  
 **RegisterButton** = findViewById(R.id.***btnSignUp***);  
  
 **Email** = findViewById(R.id.***etRollNo***);  
 **Password** = findViewById(R.id.***etPass***);  
  
 **sqLiteHelper** = **new** SQLiteHelper(**this**);  
  
 *//Adding click listener to log in button.* **LogInButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *// Calling EditText is empty or no method.* CheckEditTextStatus();  
  
 *// Calling login method.* LoginFunction();  
  
  
 }  
 });  
  
 *// Adding click listener to register button.* **RegisterButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *// Opening new user registration activity using intent on button click.* Intent intent = **new** Intent(MainActivity.**this**, RegisterActivity.**class**);  
 startActivity(intent);  
  
 }  
 });  
  
 }  
  
  
 @Override  
 **public boolean** onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.***main\_menu***, menu);  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(@NonNull MenuItem item) {  
 Intent intent2 = **new** Intent(MainActivity.**this**, LibrarianLogin.**class**);  
 startActivity(intent2);  
 **return false**;  
 }  
  
 *// Login function starts from here.* **public void** LoginFunction() {  
  
 **if** (**EditTextEmptyHolder**) {  
  
 *// Opening SQLite database write permission.* **sqLiteDatabaseObj** = **sqLiteHelper**.getWritableDatabase();  
  
 *// Adding search email query to cursor.* **cursor** = **sqLiteDatabaseObj**.query(SQLiteHelper.***TABLE\_NAME***, **null**, **" "** + SQLiteHelper.***Table\_Column\_2\_Email*** + **"=?"**, **new** String[]{**EmailHolder**}, **null**, **null**, **null**);  
  
 **while** (**cursor**.moveToNext()) {  
  
 **if** (**cursor**.isFirst()) {  
  
 **cursor**.moveToFirst();  
  
 *// Storing Password associated with entered email.* **TempPassword** = **cursor**.getString(**cursor**.getColumnIndex(SQLiteHelper.***Table\_Column\_3\_Password***));  
 **UserName** = **cursor**.getString(**cursor**.getColumnIndex(SQLiteHelper.***Table\_Column\_1\_Name***));  
  
 *// Closing cursor.* **cursor**.close();  
 }  
 }  
  
 *// Calling method to check final result ..* CheckFinalResult();  
  
 } **else** {  
  
 *//If any of login EditText empty then this block will be executed.* Toast.*makeText*(MainActivity.**this**, **"Please Enter UserName or Password."**, Toast.***LENGTH\_SHORT***).show();  
  
 }  
  
 }  
  
 *// Checking EditText is empty or not.* **public void** CheckEditTextStatus() {  
  
 *// Getting value from All EditText and storing into String Variables.* **EmailHolder** = **Email**.getText().toString();  
 **PasswordHolder** = **Password**.getText().toString();  
  
 *// Checking EditText is empty or no using TextUtils.* **EditTextEmptyHolder** = !TextUtils.*isEmpty*(**EmailHolder**) && !TextUtils.*isEmpty*(**PasswordHolder**);  
 }  
  
 *// Checking entered password from SQLite database email associated password.* **public void** CheckFinalResult() {  
  
 **if** (**TempPassword**.equalsIgnoreCase(**PasswordHolder**)) {  
  
 Toast.*makeText*(MainActivity.**this**, **"Login Successfully"**, Toast.***LENGTH\_SHORT***).show();  
  
 *// Going to Dashboard activity after login success message.* Intent intent = **new** Intent(MainActivity.**this**, DashboardActivity.**class**);  
  
 *// Sending Email to Dashboard Activity using intent.* intent.putExtra(***userName***, **UserName**);  
  
 startActivity(intent);  
  
  
 } **else** {  
  
 Toast.*makeText*(MainActivity.**this**, **"UserName or Password is Wrong, Please Try Again."**, Toast.***LENGTH\_SHORT***).show();  
  
 }  
 **TempPassword** = **"NOT\_FOUND"**;  
  
 }  
  
}

**RegisterActivity.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.text.TextUtils;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
  
**public class** RegisterActivity **extends** AppCompatActivity {  
  
 *//public static boolean NameHolder;* EditText **Email**, **Password**, **Name**;  
 Button **Register**;  
 String **NameHolder**, **EmailHolder**, **PasswordHolder**;  
 Boolean **EditTextEmptyHolder**;  
 SQLiteDatabase **sqLiteDatabaseObj**;  
 String **SQLiteDataBaseQueryHolder**;  
 SQLiteHelper **sqLiteHelper**;  
 Cursor **cursor**;  
 String **F\_Result** = **"Not\_Found"**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_register***);  
  
 **Register** = findViewById(R.id.***btnSignUp***);  
  
 **Email** = findViewById(R.id.***etRollNo***);  
 **Password** = findViewById(R.id.***etPass***);  
 **Name** = findViewById(R.id.***etName***);  
  
 **sqLiteHelper** = **new** SQLiteHelper(**this**);  
  
 *// Adding click listener to register button.* **Register**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
  
 *// Creating SQLite database if dose n't exists* SQLiteDataBaseBuild();  
  
 *// Creating SQLite table if dose n't exists.* SQLiteTableBuild();  
  
 *// Checking EditText is empty or Not.* CheckEditTextStatus();  
  
 *// Method to check Email is already exists or not.* CheckingEmailAlreadyExistsOrNot();  
 *//Sending confirmation email.* Confirm();  
  
 *// Empty EditText After done inserting process.* EmptyEditTextAfterDataInsert();  
  
  
 }  
 });  
  
 }  
  
 **public void** Confirm() {  
  
 }  
  
 *// SQLite database build method.* **public void** SQLiteDataBaseBuild() {  
  
 **sqLiteDatabaseObj** = openOrCreateDatabase(SQLiteHelper.*DATABASE\_NAME*, Context.***MODE\_PRIVATE***, **null**);  
  
 }  
  
 *// SQLite table build method.* **public void** SQLiteTableBuild() {  
  
 **sqLiteDatabaseObj**.execSQL(**"CREATE TABLE IF NOT EXISTS "** + SQLiteHelper.***TABLE\_NAME*** + **"("** + SQLiteHelper.***Table\_Column\_ID*** + **" INTEGER PRIMARY KEY AUTOINCREMENT, "** + SQLiteHelper.***Table\_Column\_1\_Name*** + **" VARCHAR, "** + SQLiteHelper.***Table\_Column\_2\_Email*** + **" VARCHAR, "** + SQLiteHelper.***Table\_Column\_3\_Password*** + **" VARCHAR);"**);  
  
 }  
  
 *// Insert data into SQLite database method.* **public void** InsertDataIntoSQLiteDatabase() {  
  
 *// If editText is not empty then this block will executed.* **if** (**EditTextEmptyHolder**) {  
  
 *// SQLite query to insert data into table.* **SQLiteDataBaseQueryHolder** = **"INSERT INTO "** + SQLiteHelper.***TABLE\_NAME*** + **" (name,email,password) VALUES('"** + **NameHolder** + **"', '"** + **EmailHolder** + **"', '"** + **PasswordHolder** + **"');"**;  
  
 *// Executing query.* **sqLiteDatabaseObj**.execSQL(**SQLiteDataBaseQueryHolder**);  
  
 *// Closing SQLite database object.* **sqLiteDatabaseObj**.close();  
  
 *// Printing toast message after done inserting.* Toast.*makeText*(RegisterActivity.**this**, **"User Registered Successfully"**, Toast.***LENGTH\_SHORT***).show();  
  
  
 }  
 *// This block will execute if any of the registration EditText is empty.* **else** {  
  
 *// Printing toast message if any of EditText is empty.* Toast.*makeText*(RegisterActivity.**this**, **"Please Fill All The Required Fields."**, Toast.***LENGTH\_SHORT***).show();  
  
 }  
  
 }  
  
 *// Empty edittext after done inserting process method.* **public void** EmptyEditTextAfterDataInsert() {  
  
 **Name**.getText().clear();  
  
 **Email**.getText().clear();  
  
 **Password**.getText().clear();  
  
 }  
  
 *// Method to check EditText is empty or Not.* **public void** CheckEditTextStatus() {  
  
 *// Getting value from All EditText and storing into String Variables.* **NameHolder** = **Name**.getText().toString();  
 **EmailHolder** = **Email**.getText().toString();  
 **PasswordHolder** = **Password**.getText().toString();  
  
 **EditTextEmptyHolder** = !TextUtils.*isEmpty*(**NameHolder**) && !TextUtils.*isEmpty*(**EmailHolder**) && !TextUtils.*isEmpty*(**PasswordHolder**);  
 }  
  
 *// Checking Email is already exists or not.* **public void** CheckingEmailAlreadyExistsOrNot() {  
  
 *// Opening SQLite database write permission.* **sqLiteDatabaseObj** = **sqLiteHelper**.getWritableDatabase();  
  
 *// Adding search email query to cursor.* **cursor** = **sqLiteDatabaseObj**.query(SQLiteHelper.***TABLE\_NAME***, **null**, **" "** + SQLiteHelper.***Table\_Column\_2\_Email*** + **"=?"**, **new** String[]{**EmailHolder**}, **null**, **null**, **null**);  
  
 **while** (**cursor**.moveToNext()) {  
  
 **if** (**cursor**.isFirst()) {  
  
 **cursor**.moveToFirst();  
  
 *// If Email is already exists then Result variable value set as Email Found.* **F\_Result** = **"Roll no Found"**;  
  
 *// Closing cursor.* **cursor**.close();  
 }  
 }  
  
 *// Calling method to check final result and insert data into SQLite database.* CheckFinalResult();  
  
 }  
  
  
 *// Checking result* **public void** CheckFinalResult() {  
  
 *// Checking whether email is already exists or not.* **if** (**F\_Result**.equalsIgnoreCase(**"Roll no found"**)) {  
  
 *// If email is exists then toast msg will display.* Toast.*makeText*(RegisterActivity.**this**, **"Roll no Already Exists"**, Toast.***LENGTH\_SHORT***).show();  
  
 } **else** {  
  
 *// If email already dose n't exists then user registration details will entered to SQLite database.* InsertDataIntoSQLiteDatabase();  
  
 }  
  
 **F\_Result** = **"Not\_Found"**;  
  
 }  
  
}

**RequestActivity.java**

**package** com.r2017.bookbank;  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** RequestActivity **extends** AppCompatActivity {  
 TextView **bookName**, **authorName**;  
 Button **reqBook**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_request***);  
  
 **bookName** = findViewById(R.id.***etReqBook***);  
 **authorName** = findViewById(R.id.***etReqAuthor***);  
 **reqBook** = findViewById(R.id.***btnRequest***);  
  
 **reqBook**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Toast.*makeText*(getBaseContext(), **"Added to request List"**, Toast.***LENGTH\_LONG***).show();  
 }  
 });  
 }  
}

**SearchAdapter.java**

**package** com.r2017.bookbank;  
  
  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.TextView;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.recyclerview.widget.RecyclerView;  
  
**import** java.util.List;  
  
**class** SearchViewHolder **extends** RecyclerView.ViewHolder {  
  
 TextView **bookName**, **authorName**;  
  
 SearchViewHolder(@NonNull View itemView) {  
 **super**(itemView);  
 **bookName** = itemView.findViewById(R.id.***bookName***);  
 **authorName** = itemView.findViewById(R.id.***authorName***);  
  
 }  
}  
  
**public class** SearchAdapter **extends** RecyclerView.Adapter<SearchViewHolder> {  
  
 **private** List<Book> **books**;  
  
 SearchAdapter(List<Book> books) {  
 **this**.**books** = books;  
 }  
  
 @NonNull  
 @Override  
 **public** SearchViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  
 LayoutInflater inflater = LayoutInflater.*from*(parent.getContext());  
 View itemView = inflater.inflate(R.layout.***layout\_item***, parent, **false**);  
  
 **return new** SearchViewHolder(itemView);  
 }  
  
 @Override  
 **public void** onBindViewHolder(@NonNull SearchViewHolder holder, **int** position) {  
  
 holder.**bookName**.setText(**books**.get(position).getBookName());  
 holder.**authorName**.setText(**books**.get(position).getAuthorName());  
  
 }  
  
 @Override  
 **public int** getItemCount() {  
 **return books**.size();  
 }  
}

**SplashActivity.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.os.Handler;  
**import** android.view.WindowManager;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**public class** SplashActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
  
 getWindow().setFlags(WindowManager.LayoutParams.***FLAG\_FULLSCREEN***,  
 WindowManager.LayoutParams.***FLAG\_FULLSCREEN***);  
 *//This method is used so that your splash activity  
 //can cover the entire screen.* setContentView(R.layout.***activity\_splash***);  
 *//this will bind your MainActivity.class file with activity\_main.* **int** SPLASH\_SCREEN\_TIME\_OUT = 2000;  
 **new** Handler().postDelayed(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Intent i = **new** Intent(SplashActivity.**this**,  
 MainActivity.**class**);  
 *//Intent is used to switch from one activity to another.* startActivity(i);  
 *//invoke the SecondActivity.* finish();  
 *//the current activity will get finished.* }  
 }, SPLASH\_SCREEN\_TIME\_OUT);  
 }  
}

**SQLiteHelper.java**

**package** com.r2017.bookbank;  
  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**public class** SQLiteHelper **extends** SQLiteOpenHelper {  
  
 **static final** String ***TABLE\_NAME*** = **"UserTable"**;  
 **static final** String ***Table\_Column\_ID*** = **"id"**;  
 **static final** String ***Table\_Column\_1\_Name*** = **"name"**;  
 **static final** String ***Table\_Column\_2\_Email*** = **"email"**;  
 **static final** String ***Table\_Column\_3\_Password*** = **"password"**;  
 **static** String *DATABASE\_NAME* = **"UserDataBase"**;  
  
 SQLiteHelper(Context context) {  
  
 **super**(context, *DATABASE\_NAME*, **null**, 1);  
  
 }  
  
 @Override  
 **public void** onCreate(SQLiteDatabase database) {  
  
 String CREATE\_TABLE = **"CREATE TABLE IF NOT EXISTS "** + ***TABLE\_NAME*** + **" ("** + ***Table\_Column\_ID*** + **" INTEGER PRIMARY KEY, "** + ***Table\_Column\_1\_Name*** + **" VARCHAR, "** + ***Table\_Column\_2\_Email*** + **" VARCHAR, "** + ***Table\_Column\_3\_Password*** + **" VARCHAR)"**;  
 database.execSQL(CREATE\_TABLE);  
  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 db.execSQL(**"DROP TABLE IF EXISTS "** + ***TABLE\_NAME***);  
 onCreate(db);  
  
 }  
  
}

**activity\_dashboard.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**TextView  
 android:id="@+id/tvHello"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/hello"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 android:layout\_gravity="center"  
 android:layout\_margin="20sp"  
 android:fontFamily="@font/google"  
 android:textColor="@color/colorPrimaryDark"** />  
  
 <**ImageView  
 android:id="@+id/ivBook"  
 android:layout\_width="265dp"  
 android:layout\_height="265dp"  
 android:layout\_gravity="center"  
 android:contentDescription="@string/library\_logo"  
 app:srcCompat="@drawable/ebook"** />  
  
 <**Button  
 android:id="@+id/btnGetStarted"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="@string/get\_started"  
 android:layout\_margin="20sp"  
 android:fontFamily="@font/google"  
 android:textStyle="bold"** />  
  
 <**Button  
 android:id="@+id/logout"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:layout\_gravity="center"  
 android:text="@string/logout"  
 android:textStyle="bold"** />  
</**LinearLayout**>

**activity\_home.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".HomeActivity"**>  
  
 <**com.mancj.materialsearchbar.MaterialSearchBar  
 android:id="@+id/search\_bar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 app:mt\_hint="Search"  
 app:mt\_speechMode="false"** />  
  
 <**androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recycler\_search"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_below="@+id/search\_bar"** />  
  
  
</**RelativeLayout**>

**activity\_librarian\_entry.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".LibrarianEntry"**>  
  
 <**TextView  
 android:id="@+id/add\_book"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_margin="15dp"  
 android:fontFamily="@font/google"  
 android:gravity="center"  
 android:text="@string/add\_book"  
 android:textColor="@color/colorPrimary"  
 android:textSize="30sp"  
 android:textStyle="bold"** />  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"  
 android:layout\_margin="30dp"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etBookName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:hint="@string/book\_name"** />  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30dp"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etAuthorName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:hint="@string/author\_name"** />  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**Button  
 android:id="@+id/btnAddBook"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/add\_book"  
 android:textStyle="bold"** />  
</**LinearLayout**>

**activity\_librarian\_login.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".LibrarianLogin"**>  
  
 <**ImageView  
 android:id="@+id/ivLibrarian"  
 android:layout\_width="191dp"  
 android:layout\_height="191dp"  
 android:layout\_gravity="center"  
 android:contentDescription="@string/librarian"  
 app:srcCompat="@drawable/librarian"** />  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30dp"  
 android:hint="@string/username"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etLibUserName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30dp"  
 android:hint="@string/password"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etLibPass"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:inputType="textPassword"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
 <**Button  
 android:id="@+id/btnLibLogin"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/login"  
 android:textStyle="bold"** />  
  
</**LinearLayout**>

**activity\_mail.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingLeft="16dp"  
 android:paddingTop="16dp"  
 android:paddingRight="16dp"  
 android:paddingBottom="16dp"  
 android:orientation="vertical"  
 tools:context="com.r2017.bookbank.MainActivity"**>  
  
 <**TextView  
 android:id="@+id/tvTopText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:layout\_gravity="center"  
 android:layout\_margin="15dp"  
 android:textColor="@color/colorPrimary"  
 android:fontFamily="@font/google"  
 android:text="@string/login"  
 android:textSize="30sp"  
 android:textStyle="bold"** />  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:hint="@string/enter\_rollno"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etRollNo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"  
 android:hint="@string/password"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etPass"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:inputType="textPassword"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**Button  
 android:id="@+id/btnLogin"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="19dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/login"  
 android:textStyle="bold"** />  
  
 <**Button  
 android:id="@+id/btnSignUp"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/not\_a\_user\_sign\_up"  
 android:textStyle="bold"** />  
  
</**LinearLayout**>

**activity\_register.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_register"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingLeft="16dp"  
 android:paddingTop="16dp"  
 android:paddingRight="16dp"  
 android:paddingBottom="16dp"  
 android:orientation="vertical"  
 tools:context="com.r2017.bookbank.RegisterActivity"**>  
  
 <**TextView  
 android:id="@+id/tvTopText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:layout\_margin="15dp"  
 android:textColor="@color/colorPrimary"  
 android:text="@string/sign\_up"  
 android:textAllCaps="false"  
 android:textSize="30sp"  
 android:textStyle="bold"** />  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="20dp"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:hint="@string/username"** />  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"  
 android:layout\_margin="20dp"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etRollNo"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:hint="@string/enter\_rollno"** />  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"  
 android:layout\_margin="20dp"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etPass"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"  
 android:hint="@string/password"  
 android:inputType="textPassword"** />  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
  
 <**Button  
 android:id="@+id/btnSignUp"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_width="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/sign\_up"  
 android:textStyle="bold"** />  
  
</**LinearLayout**>

**activity\_request.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".RequestActivity"**>  
  
 <**TextView  
 android:id="@+id/tvRequest"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:layout\_margin="15dp"  
 android:text="@string/request"  
 android:textColor="@color/colorPrimary"  
 android:textSize="30sp"  
 android:textStyle="bold"** />  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30dp"  
 android:hint="@string/book\_name"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etReqBook"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
 <**com.google.android.material.textfield.TextInputLayout  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="30dp"  
 android:hint="@string/author\_name"**>  
  
 <**com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/etReqAuthor"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="@font/google"** />  
  
 </**com.google.android.material.textfield.TextInputLayout**>  
  
 <**Button  
 android:id="@+id/btnRequest"  
 style="?attr/materialButtonOutlinedStyle"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/request"  
 android:textStyle="bold"** />  
  
</**LinearLayout**>

**activity\_splash.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
  
 tools:context=".SplashActivity"  
 tools:ignore="UseCompoundDrawables"**>  
  
 <**ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="top"  
 android:layout\_margin="20dp"  
 android:contentDescription="@string/library\_logo"  
 android:src="@drawable/ebook"** />  
  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:fontFamily="@font/google"  
 android:text="@string/copyright\_2020\_2021"  
 android:textSize="15sp"  
 android:textStyle="bold"** />  
  
  
</**LinearLayout**>

**layout\_item.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 app:cardElevation="8dp"  
 tools:ignore="ExtraText"**>  
  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="8dp"  
 android:background="@android:color/white"  
 android:orientation="horizontal"**>  
  
 <**ImageView  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:contentDescription="@string/book\_icon"  
 android:src="@drawable/coding"** />  
  
 <**LinearLayout  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="9"  
 android:orientation="vertical"**>  
  
 <**TextView  
 android:id="@+id/bookName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:gravity="center\_vertical|start"  
 android:text="@string/bookname"  
 android:textAllCaps="true"  
 android:textStyle="bold"** />  
  
 <**TextView  
 android:id="@+id/authorName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:gravity="center\_vertical|start"  
 android:text="@string/author"  
 android:textStyle="italic"** />  
  
  
 </**LinearLayout**>  
 </**LinearLayout**>  
  
</**androidx.cardview.widget.CardView**>

**mail\_menu.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"**>  
 <**item  
 android:id="@+id/Librarian"  
 android:title="@string/Librarian"  
 app:showAsAction="never"** />  
</**menu**>

**request.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"**>  
 <**item  
 android:id="@+id/request"  
 android:title="@string/request"  
 app:showAsAction="never"** />  
  
</**menu**>

**colors.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
 <**color name="colorPrimary"**>#6200EE</**color**>  
 <**color name="colorPrimaryDark"**>#3700B3</**color**>  
 <**color name="colorAccent"**>#03DAC5</**color**>  
 <**color name="title\_text\_color"**>#FFFFFF</**color**>  
</**resources**>

**ic\_launcher\_background.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
 <**color name="ic\_launcher\_background"**>#FFFFFF</**color**>  
</**resources**>

**ids.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
 <**item name="editName" type="id"** />  
</**resources**>

**string.xml**

<**resources**>  
 <**string name="app\_name"**>Bookbank</**string**>  
 <**string name="hello"**>Hello! </**string**>  
 <**string name="logout"**>Logout</**string**>  
 <**string name="user\_login"**>User Login</**string**>  
 <**string name="enter\_rollno"**>Roll no</**string**>  
 <**string name="login"**>Login</**string**>  
 <**string name="not\_a\_user\_sign\_up"**>Sign up</**string**>  
 <**string name="sign\_up"**>Sign up</**string**>  
 <**string name="get\_started"**>Get Started</**string**>  
 <**string name="Librarian"**>Librarian</**string**>  
 <**string name="username"**>Username</**string**>  
 <**string name="password"**>Password</**string**>  
 <**string name="add\_book"**>Add Book</**string**>  
 <**string name="author\_name"**>Author name</**string**>  
 <**string name="book\_name"**>Book name</**string**>  
 <**string name="library\_logo"**>LibraryLogo</**string**>  
 <**string name="search\_books"**>Search Books</**string**>  
 <**string name="search"**>Search</**string**>  
 <**string name="librarian"**>librarian</**string**>  
 <**string name="request"**>Request</**string**>  
 <**string name="book"**>Book</**string**>  
 <**string name="book\_icon"**>book</**string**>  
 <**string name="bookname"**>harish</**string**>  
 <**string name="author"**>hari</**string**>  
 <**string name="copyright\_2020\_2021"**>© Developed by CSEKINGS ❤️</**string**>  
</**resources**>

**styles.xml**

<**resources**>  
  
 *<!-- Base application theme. -->* <**style name="AppTheme" parent="Theme.MaterialComponents.DayNight.DarkActionBar"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
 </**style**>  
  
 <**style name="AppTheme.NoActionBar"**>  
 <**item name="windowActionBar"**>false</**item**>  
 <**item name="windowNoTitle"**>true</**item**>  
 </**style**>  
  
</**resources**>

**backup\_descriptor.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**full-backup-content**>  
 *<!-- Exclude specific shared preferences that contain GCM registration Id -->*</**full-backup-content**>

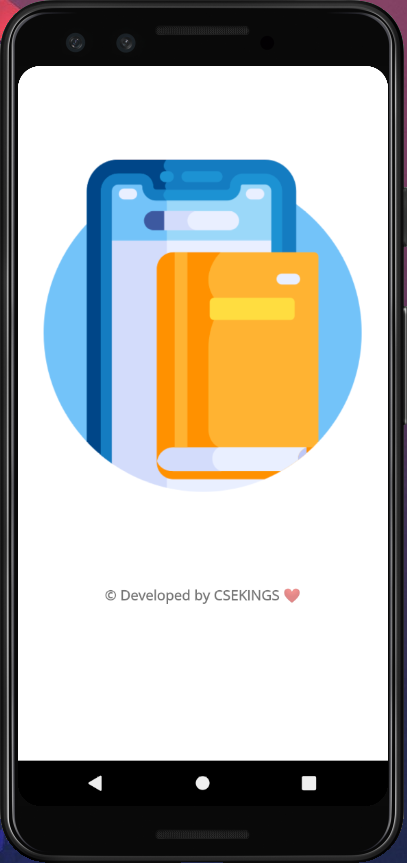
**built.gradle**

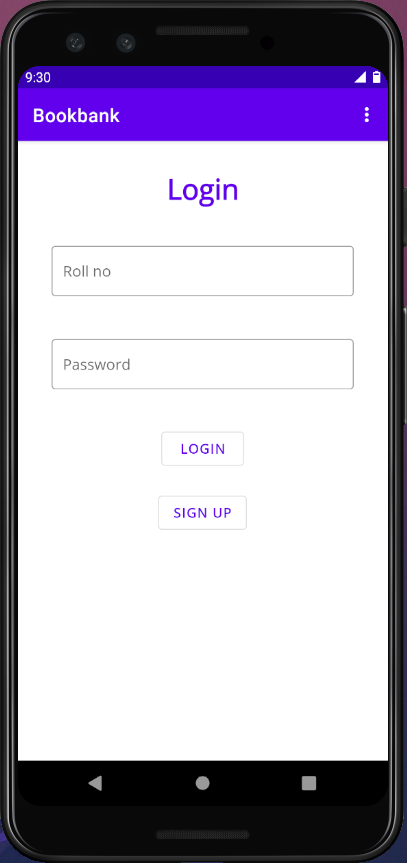
*// Top-level build file where you can add configuration options common to all sub-projects/modules.*buildscript {  
   
 repositories {  
 google()  
 jcenter()  
   
 }  
 dependencies {  
 classpath **'com.android.tools.build:gradle:3.6.1'** *// NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files* }  
}  
  
  
allprojects {  
 repositories {  
 google()  
 jcenter()  
 maven { url **"https://jitpack.io"** }  
   
 }  
}  
  
task clean(type: Delete) {  
 delete rootProject.buildDir  
}

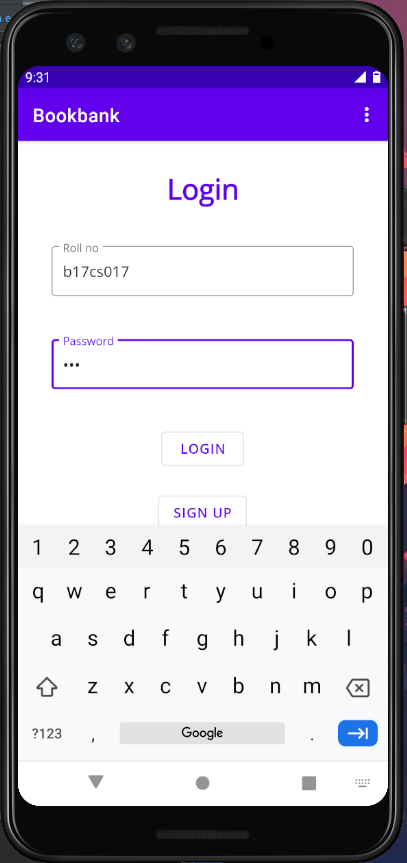
**built.gradle**

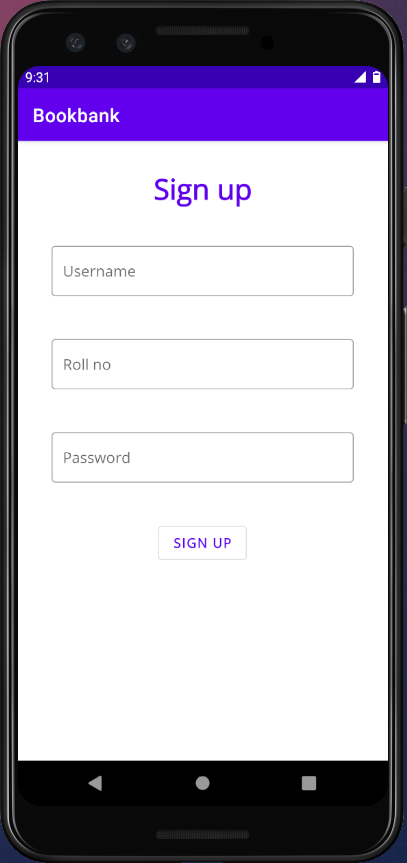
apply **plugin**: **'com.android.application'**android **{** compileSdkVersion 29  
 buildToolsVersion **"29.0.3"** defaultConfig **{** applicationId **"com.r2017.bookbank"** minSdkVersion 21  
 targetSdkVersion 29  
 versionCode 1  
 versionName **"1.0"** testInstrumentationRunner **"androidx.test.runner.AndroidJUnitRunner"  
 }** buildTypes **{** release **{** minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android-optimize.txt'**), **'proguard-rules.pro'  
 }  
 }  
  
  
}**dependencies **{** implementation fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])  
  
 implementation **'androidx.appcompat:appcompat:1.1.0'** implementation **'androidx.constraintlayout:constraintlayout:1.1.3'** testImplementation **'junit:junit:4.13'** androidTestImplementation **'androidx.test.ext:junit:1.1.1'** androidTestImplementation **'androidx.test.espresso:espresso-core:3.2.0'** implementation **'com.readystatesoftware.sqliteasset:sqliteassethelper:2.0.1'** implementation **'androidx.recyclerview:recyclerview:1.1.0'** implementation **"androidx.cardview:cardview:1.0.0"** implementation **'com.github.mancj:MaterialSearchBar:0.8.5'** implementation **'com.google.android.material:material:1.1.0'  
  
  
}**

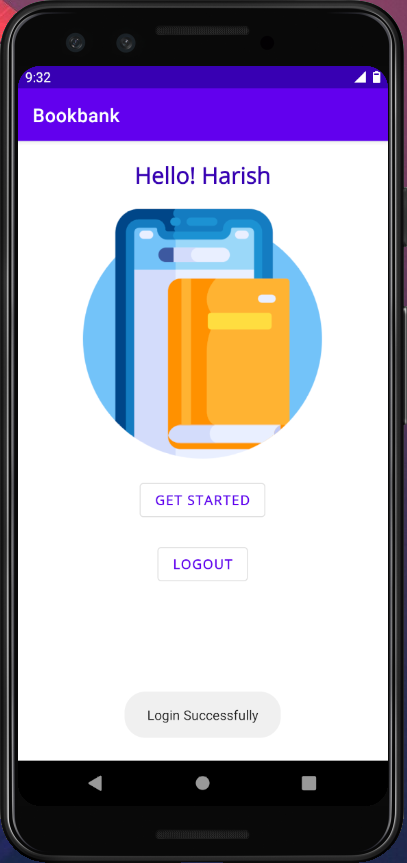
**9.2 Interface**

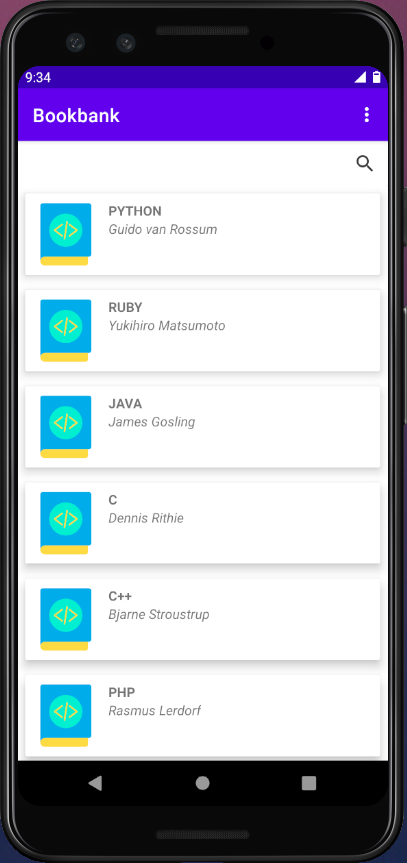
****

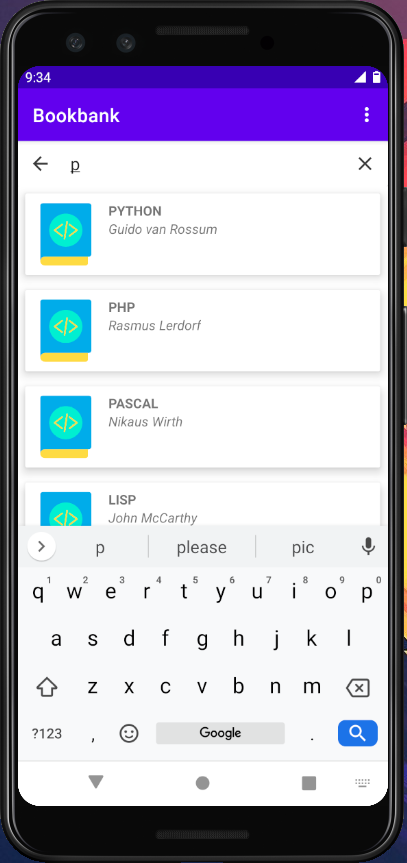
****

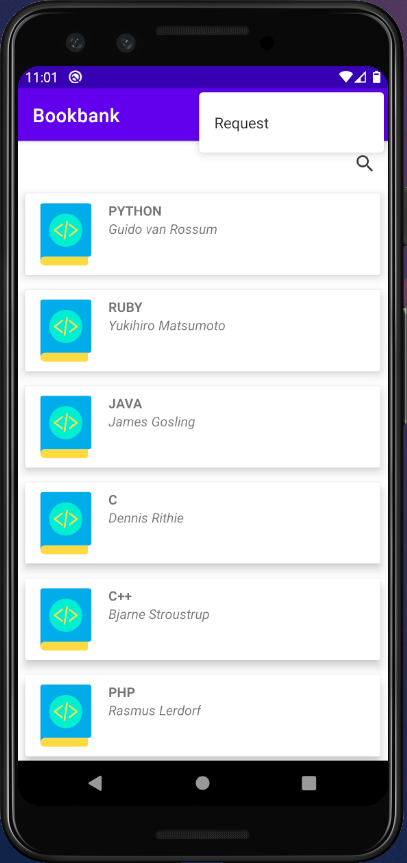
****

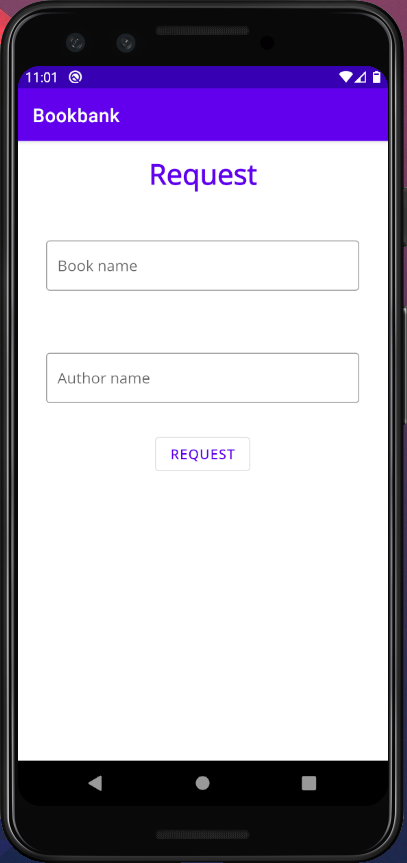
****

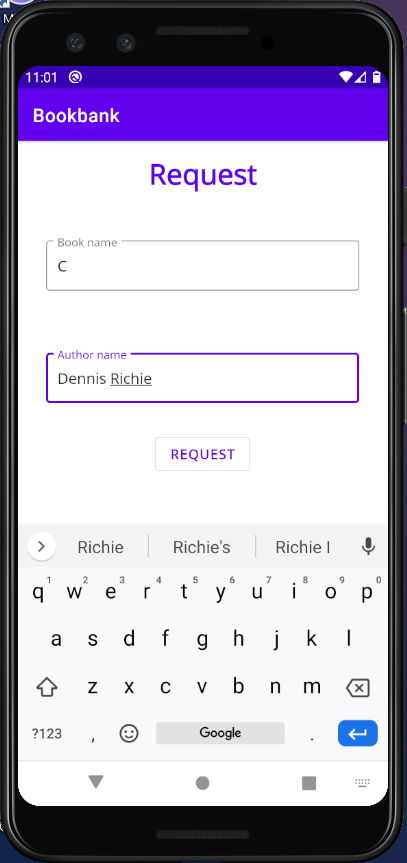
****

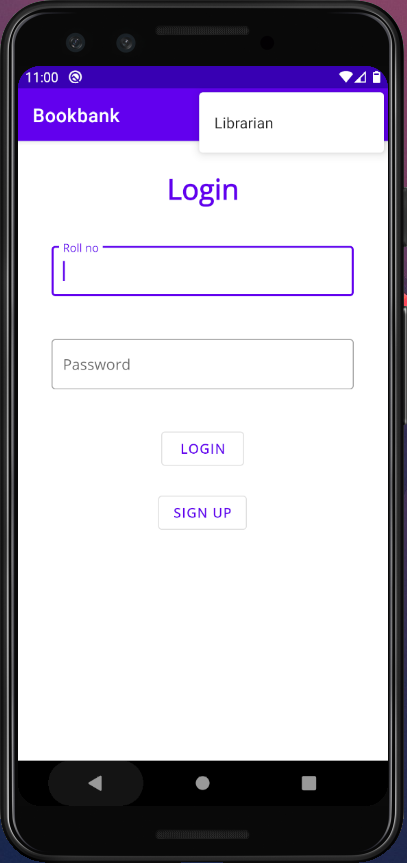
****

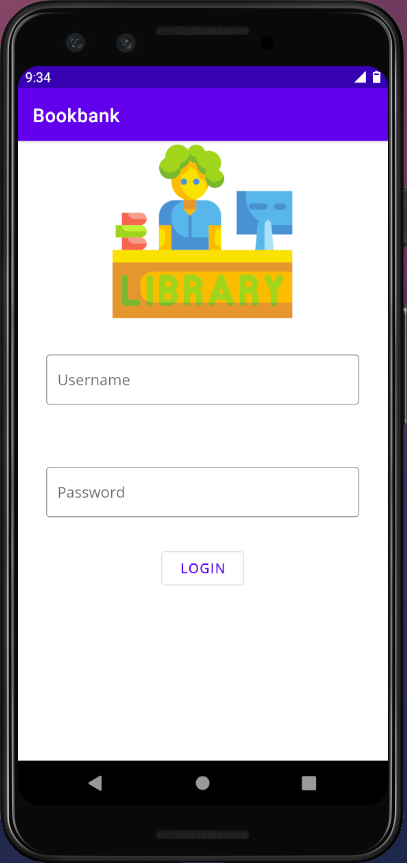
****

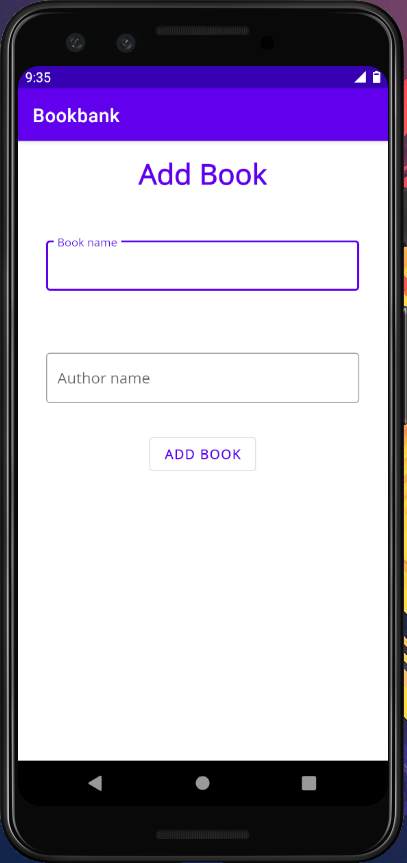
****

****

****

****

****

****