

ANDROID APP FOR SEARCHING BOOKS IN A LIBRARY

Submitted in partial fulfilment of the required for the award of

Masters of Computer Science

by

DANIEL ARNOLD R

REGISTER NO (42734005)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY**

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

APRIL 2024



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC

DEPARTMENT OF COMPUTER SCIENCE
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **DANIEL ARNOLD R (42734005)** who carried out the project entitled "ANDROID APP FOR SEARCHING BOOKS IN A LIBRARY" under our supervision from January 2024 to April 2024.

Internal Guide

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for viva voce examination held on 30.04.24.

Internal Examiner

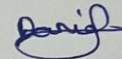
External Examiner

DECLARATION

I am, **DANIEL ARNOLD R (42734005)** hereby declare that the Project Report entitled “**ANDROID APP FOR SEARCHING BOOKS IN A LIBRARY**” done by me under the guidance of **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.**, Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20.04.24.

PLACE: CHENNAI



DANIEL ARNOLD R

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D., Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Dr. REKHA CHAKRAVARTHI M.E., Ph.D.,** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish my express my thanks to all Teaching and non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

DANIEL ARNOLD R

ABSTRACT

Many libraries continue to employ antiquated methods to manage their holdings, making it difficult for patrons to discover the books they want. This causes both library employees and users to be frustrated and lose time. This application is intended to make searching for books in a library faster, simpler, and more efficient. Users may immediately locate the books by using the application, which has a simple design and extensive search features. It will inform us of the book's location and whether it is available for checkout from the library. Knowing the book's title is necessary for this process because we will be using it to conduct our search. This eliminates the typical method of discovering books in a library and focuses on lowering the time required to find the book.

Keywords: Library management, Android application, Library Access System.

TABLE OF CONTENTS

Sl.NO	CHAPTERS	PAGE NO
1	INTRODUCTION 1.1 Overview of android studio. 1.2 Components of android studio. 1.3 Overview of firebase.	1
2	LITERATURE SURVEY	8
3	EXISTING SYSTEM	10
4	PROPOSED SYSTEM	12
5	ARCHITECTURE DIAGRAM 5.1 User's architecture explanation. 5.2 Admin architecture explanation. 5.3 Basic architecture explanation.	13
6	MODULES	20
7	RESULT AND DISCUSS	29
8	CONCLUSION	34
9	CODING 9.1 Sample codes. 9.2 Screenshots.	35
10	REFERENCES	61

LIST OF FIGURES

LIST OF FIGURES	NAME OF THE FIGURE	PAGE NO
1.1	Project files in Android project view.	2
1.2	Manifest file.	2
1.3	Build.gradle.	3
1.4	App inspection.	4
1.5	Firebase Realtime Database.	5
1.6	Firebase Authentication.	6
1.7	Firebase Cloud Messaging.	6
5.1	User's architecture.	13
5.2	Admin architecture.	16
5.3	Basic architecture.	19
6.1	Login page.	21
6.2	Register page.	21
6.3	Google sign-in page.	21
6.4	Admin login page.	21
6.5	Home page.	22
6.6	Admin page.	23
6.7	Fiction page.	23
6.8	Fiction Genre page.	23
6.9	Non-Fiction page.	24
6.10	Non-Fiction Genre page.	24
6.11	Request book page.	25
6.12	Add book page.	26
6.13	Delete book page.	26
6.14	Update book page.	27

6.15	Internal Notification page.	28
6.16	External Notification (FCM).	29
7.1	Fiction book details.	30
7.2	Non-Fiction book details.	30
7.3	Requesting book.	30
7.4	Database for request.	30
7.5	Adding book.	31
7.6	Book stored in database.	31
7.7	Deleting book.	31
7.8	Book deleted.	31
7.9	Updating book.	32
7.10	Updated book in database.	32
7.11	Internal notification.	33
7.12	Notification list.	33
7.13	External notification (FCM).	33
7.14	Google sign-in.	34
9.2	Screenshots.	59

1. INTRODUCTION

- The Android application for searching books in libraries makes it easier for people to find the information they need.
- Giving the book's details in a library is the main goal of the application.
- This application makes it simple to find out whether a book is available or even if it is present in the library.
- Previously, all library access procedures were manual, and finding a certain book took more time.
- Generally, the search calls for a librarian because they can assist us in locating the necessary books.
- Since individuals from all over the world utilize various communication devices for a variety of purposes these days, the internet can help replace this conventional way.
- The creation of Android applications can be helpful in locations such as libraries, where they can help automate the search process.
- With this software, anyone with an Android phone and an internet connection can search the library's collection of books from any location.
- This tool assists us in determining the exact location of each desired book in the library as well as its availability.
- This Android application's user interface is made to be as easy to use as possible for all users and administrators who are linked to the application.

1.1 OVERVIEW OF ANDROID STUDIO:

- Each project in Android Studio contains one or more modules with source code files and resource files.
- The types of modules include:
 - 1) Android app modules.
 - 2) Library modules.
 - 3) Google App Engine modules.

- By default, Android Studio displays your project files in the Android project view, as shown in figure 1.
- This view is organized by modules to provide quick access to your project's key source files.
- All the build files are visible at the top level, under **Gradle Scripts**.

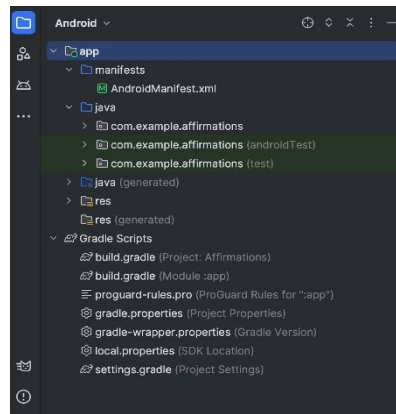


Figure 1.1: Project files in Android project view.

1.2 COMPONENTS OF ANDROID STUDIO:

- You may have encountered several Android Studio components that are crucial to the app development process while writing your application from the ground up to deployment.
- The list of a few Android Studio components is as follows.

1. MANIFEST FILE:

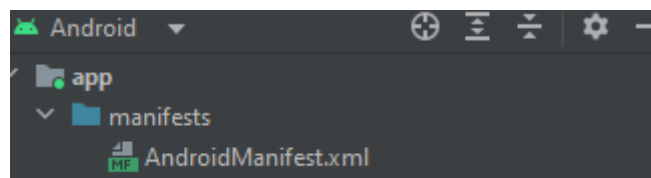


Figure 1.2: Manifest file.

- Any Android application must include the AndroidManifest.xml file.
- It gives the Android operating system access to crucial details about the program, such as its package name, version, permissions, activities, services, and receivers.

2. BUILD.GRADLE:

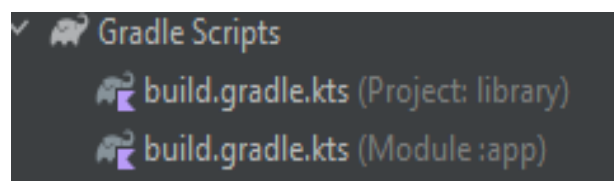


Figure 1.3: Build.gradle.

- A script file called build.gradle specifies how your Android application is created.
- It outlines several build process activities, dependencies, and configurations.
- An Android project contains two tiers of build.gradle files:

1.Build.gradle at the top level:

- Found in the project's root directory.
- It specifies parameters that are relevant to the whole project.

2.Build.gradle at the module level:

- Found inside the app module, which is often called app/build.gradle.
- It includes app module-specific parameters.

3. APP INSPECTION:

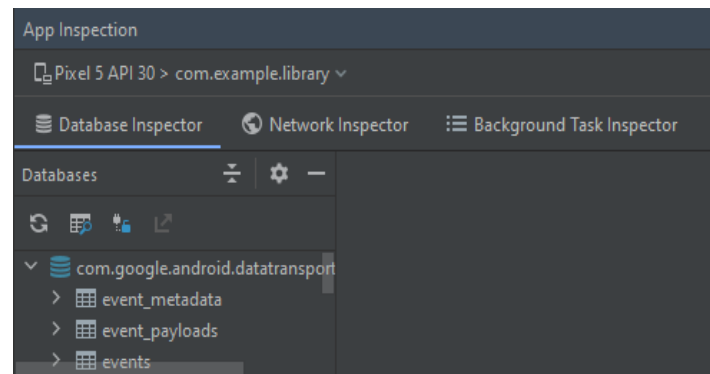


Figure 1.4: App Inspection.

- While your app is operating, you may examine, query, and edit its databases using Android Studio's App Inspection tool.
- It's especially helpful when debugging databases.
- This is how you apply it:

- [1] Open the Database Inspector.
- [2] View and Modify Data.
- [3] See Live Database Changes.
- [4] Query Your Databases.

1.3 OVERVIEW OF FIREBASE:

- Firebase can power your app's backend, including data storage, user authentication, static hosting, and more.
- Focus on creating extraordinary user experiences.
- We will take care of the rest.
- Build cross-platform native mobile and web apps with our Android, iOS, and JavaScript SDKs.
- You can also connect Firebase to your existing backend using our server-side libraries or our REST API.

FIREBASE FEATURES:

1. Firebase real-time database:

- Every linked client receives real-time data synchronization from the Firebase Realtime Database, which stores the data in JSON format.
- Building cross-platform apps with JavaScript SDKs, Android platforms, and Apple platforms allows all clients to share a single Realtime Database instance and get updates with the most recent data immediately.
- For contemporary applications that demand query-ability, scalability, higher availability, and richer data models, take into consideration utilizing Cloud Fire-store as an alternative.

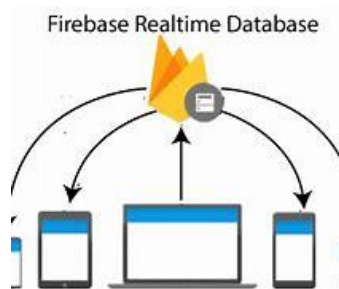


Figure 1.5: Firebase Realtime Database.

2. Firebase Authentication:

- A service called Firebase Authentication enables users to log in to your app in a number of ways.
- You may incorporate social networking, phone number, or password-based authentication into your app by using Firebase Authentication.
- To make the authentication process simpler, Firebase Authentication offers ready-made UI frameworks, backend services, and simple SDKs.



Figure 1.6: Firebase Authentication.

3. Firebase Cloud Messaging:

- Web, iOS, and Android application developers can send messages to each other with confidence thanks to Firebase Cloud Messaging (FCM), a robust cross-platform messaging system.
- You may send free messages and notifications to your app users using (FCM).
- (FCM) uses timely message delivery to encourage user retention and re-engagement.



Figure 1.7: Firebase Cloud Messaging.

FIREBASE ADVANTAGES:

- It is simple and user friendly. No need for complicated configuration.
- The data is real-time, which means that every change will automatically update connected clients.
- Firebase offers simple control dashboard.
- There are a number of useful services to choose.

FIREBASE LIMITATIONS:

- Despite its strength, Firebase is not without its restrictions.
- Let's investigate them:
 - a.) Query Duration: Before failing, a single query may run for up to fifteen minutes.
 - b.) Console Query Time: A query on the Firebase console has a maximum 5-second runtime before failing.
 - c.) Data Tree Depth: Your data tree can only have paths that are no more than 32 levels deep.
 - d.) Simultaneous Connection: Up to 100 connections can be made simultaneously using Firebase.

2.LITERATURE SURVEY

SL.NO	AUTHOR	TITLE	JOURNAL	PUBLISHING YEAR
1	*K.Prabhavathy, *S.Gogula Nandhini, *M.Dhanalakshmi, *R.Gayathri, **Dr.K.Sumathi *UG Students, ** Assistant Professor	An Android Application for Library Access System.	International conference on applied Soft computing techniques (ICASCT)	2018
2	C. Srujana, B. Rama Murthy, K.TanveerAlam, U. Sunitha, Mahammad D.V, P.Thimmaiah.	Development of RFID Based Library ManagementSystem Using MATLAB.	International Journal of Engineering and Advanced Technology (IJEAT)	June-2013
3	R.Dinesh, S.R.Arun Pravin, M.Aravindhana, D.Rajeswari	A library access system smartphone application using android.	International journal of Computer Science and Mobile Computing (IJCSMC)	March-2015
4	Pons, D., Vallés, R., Abarca, M. and Rubio, F.	“QR codes in use: the experience at the UPV Library”	The Journal for the Serials Community	2011
5	A. Lalwani,C. Bhure and P.R.L.	VESIT-Library – An Android Based Application.	International Journal of Engineering and Computer Science.	2017

6	Joju Mohan, Rahul K Raj, and Dhaneesh Chandran	RFID Based Library Automation with wireless 2.4GHz 802.11b wireless communication.	IEEE Conference	March 2012
7	Renold, A.P. ; Joshi, R.R	An internet based RFID library management system.	IEEE Conference on Information & Communication Technologies (ICT)	April 2013.
8	Polycarpou, A.C. ; Samaras, T. ; Sahalos, J.N.	An RFIDbased library management system using smart cabinets: A pilot project	IEEE European Conference on Antennas and Propagation (EuCAP)	April 2014.
9	Narayanan A., Sanjay Singh and Somasekharan M.	Implementing RFID in Library: Methodologies, Advantages and Disadvantages	Scientific Information Resource Division, IGCAR	March 2018.
10	Fennani, A. and NB ; Hamam, H.	An Optimized RFID-Based Academic Library.	IEEE Second International Conference on Sensor Technologies and Application.	August 2008

3.EXISTING SYSTEM

1. Ashutosh Tripathi & Ashish Srivastava proposed the paper “Online Library Management System” in Feb2012 [1]. The paper for Online Library Management System is developed in java which mainly focuses on basic operations in library. This proposed system provides facilities to the student to search for required books and this allows the Librarian to Issue and return the books to student. The proposed system is a windows application written for 32-bit windows operating systems, designed to help users maintain and organize library. Advantages for this is to provide user friendly interface for both beginners and advanced user, so easy and fast access to use for beginners and advanced users. A Disadvantage is to managing the library by manually at librarian side.
2. C. Srujana, B. Rama Murthy, K. Tanveer Alam, U. Sunitha, Mahammad D.V, P. Thimmaiah proposed the paper ” Development of RFID Based Library Management System” in June2013 [2]. RFID is a Radio Frequency Identification, it is an automated identification technique used for the fast transactions of books and journals. This system is based on the high frequency DLP RFID1 Read/Write having the frequency range is up to 13.5Hz. The database for this system is maintained with MySQL using MATLAB, which is which stores detailed information about the books. The advantages are provides intelligent and effective services to both librarian and student such as Reminders for due dates allows patrons to submit borrowed books and other materials in time. This RFID technology also provides Patrons will spend less time waiting in checking out lines by using Self Check in Check out systems. A Disadvantage is to High initialization Cost and difficult to handle and operate.

3. Libramatics is basically a Cloud-based library management service provider based in Dublin, Ireland. Libramatics retrieves books information such as its cover arts, title, author, publisher and an even abstract automatically based on its ISBN codes. Libramatic turns a Smartphone's camera or by using USB barcode scanner, allowing librarian to loan books, pull up author information, and locate specific titles on their shelves without having to squint or type in any numbers by hand. Advantage is to use of online library management system has increased drastically since the rise of the "Cloud" technology and also it allowing librarians to circulate books based on an ISBN. A Disadvantage is To import their student lists, libraries need to email Libramatic file with the name of their students, which some libraries may not feel comfortable doing for privacy reasons.
4. CERN Document Server Software Consortium proposed the paper "Invenio" in November 2012 [4]. Invenio is an open source free software package suite enabling and providing the tools for management of own digital library on the web. A Digital Library is a special library can vary immensely in size and scope, and focused on the collection of digital objects. The Digital Objects include visual material, electronic media formats and these electronic content may be stored locally, or accessed remotely via computer networks, it is a type of information retrieval system. Advantage is to it is web based and written in Python programming language. A Disadvantage is to Lack of Security and trying to find particular book consumes lots of time.

4.PROPOSED SYSTEM

Furthermore, the proposed application is extremely similar to the existing application. The basic concept here is to see the exact position of the books inside the library by simply adding elements like the rack or shelves in which the book is placed and the floor or blocks where they are kept. This application is valuable for students, faculty, and librarians since it allows them to browse books without having to manually search for them, which saves time. Knowing the book category will help us obtain the necessary book information through the application with ease. Since we will be utilizing the book's title to search for it, it is crucial that we are aware of the book's name or title. The information about the required books, including the shelf, floor, ISBN, bay, author, and call number of that specific book, will be provided by this operation. It will assist us in some manner in locating the books within the library. Users can sign in to the library application using their Google account as well. Users can also submit book requests through the application's request page, which helps the administrator determine what kinds of books users would like to see kept in the library. The administrator can manage the database through the program itself, including updating, deleting, and adding new books. Within the application, there is a separate module for the users and the admin. In order for users to communicate and receive notifications from the program, there must be a communication channel between the admin and users. This is accomplished by using the (FCM), which allows the admin to deliver notifications to users both inside and outside.

5.ARCHITECTURE DIAGRAM

1.User's architecture:

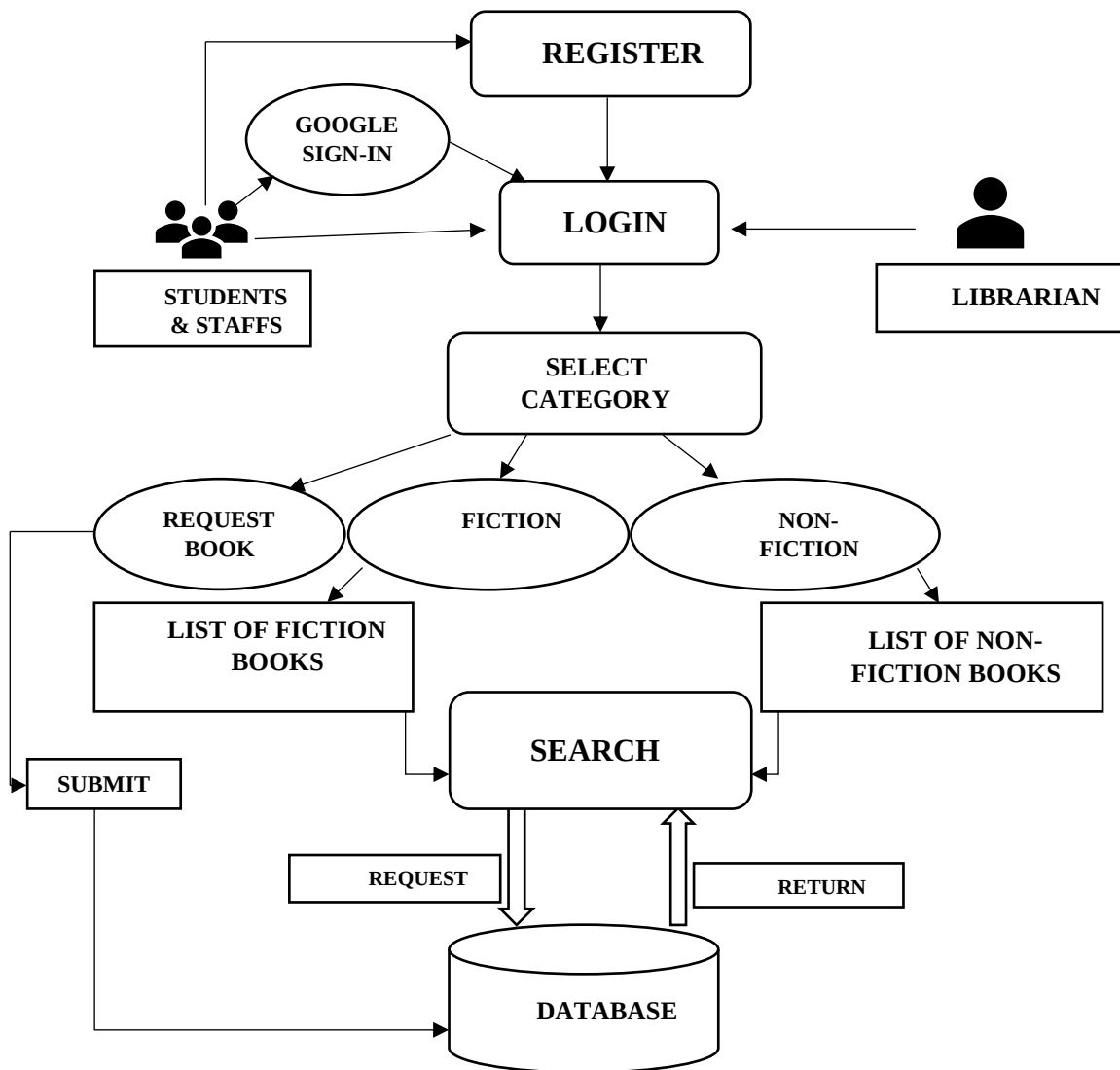


Fig 5.1: User's architecture.

5.1 USER'S ARCHITECTURE EXPLANATION:

1.LOGIN:

The login is the first thing the students and the staffs need to do. At first they have to login with the given id and password from their respective android phones. After successful login only the users can proceed to view the app.

2.REGISTER:

The students and the staffs can also register with the new credentials after the first login. The register activity is not for the librarian. After registering new credentials successfully the users can use that registered credentials for login to the library application.

3.GOOGLE SIGN-IN:

The google sign-in is another way and most effective way of authentication focusing on providing authorization to the users and staffs. In **Google's** own words, "Sign In With Google helps you to quickly and easily manage user authentication and sign-in to your website".

4.SELECT CATEGORY:

The select book is a category in which we have to view our book it can be either fictional books or non-fictional books. After selecting or clicking any of the category it will redirect into another activity which contains the next list or category to select from both the parent category.

5.FICTION & NON-FICTION:

They both contains the list of the books. For example, if a user is having a fictional book under the genre horror it will be present inside the list of other genres. From that user can search their books present under that particular genre.

6.REQUEST BOOK:

The request book is a useful way for the users to make or create a request of their own. This request method is useful when the users and staffs are not able to find the needed book in the library. With the help of this method the users and the staffs can create a new request for the book by giving certain details of the book such as the Book name, Author name, Genre and Year.

7.SUBMIT:

The submit is used to submit the data entered by the users in the request book to the database. The data given by users are stored in a separate node that are accessed by the admin.

8.SEARCH:

The search is the operation performed to get the detail of the book. Usually, the search requires the title of the book. The user gives the title as input and perform search operation. When the input is given in the application it will immediately request the database to give the details of the book.

9.DATABASE:

The database is the storage area of the books. As soon as the user inputs the title and the request is made the database sees on the valid response it will check for the availability of the book in its storage. When the book is not found or does not exist it will tell the user with a message such as book doesn't exist.

2.Admin architecture:

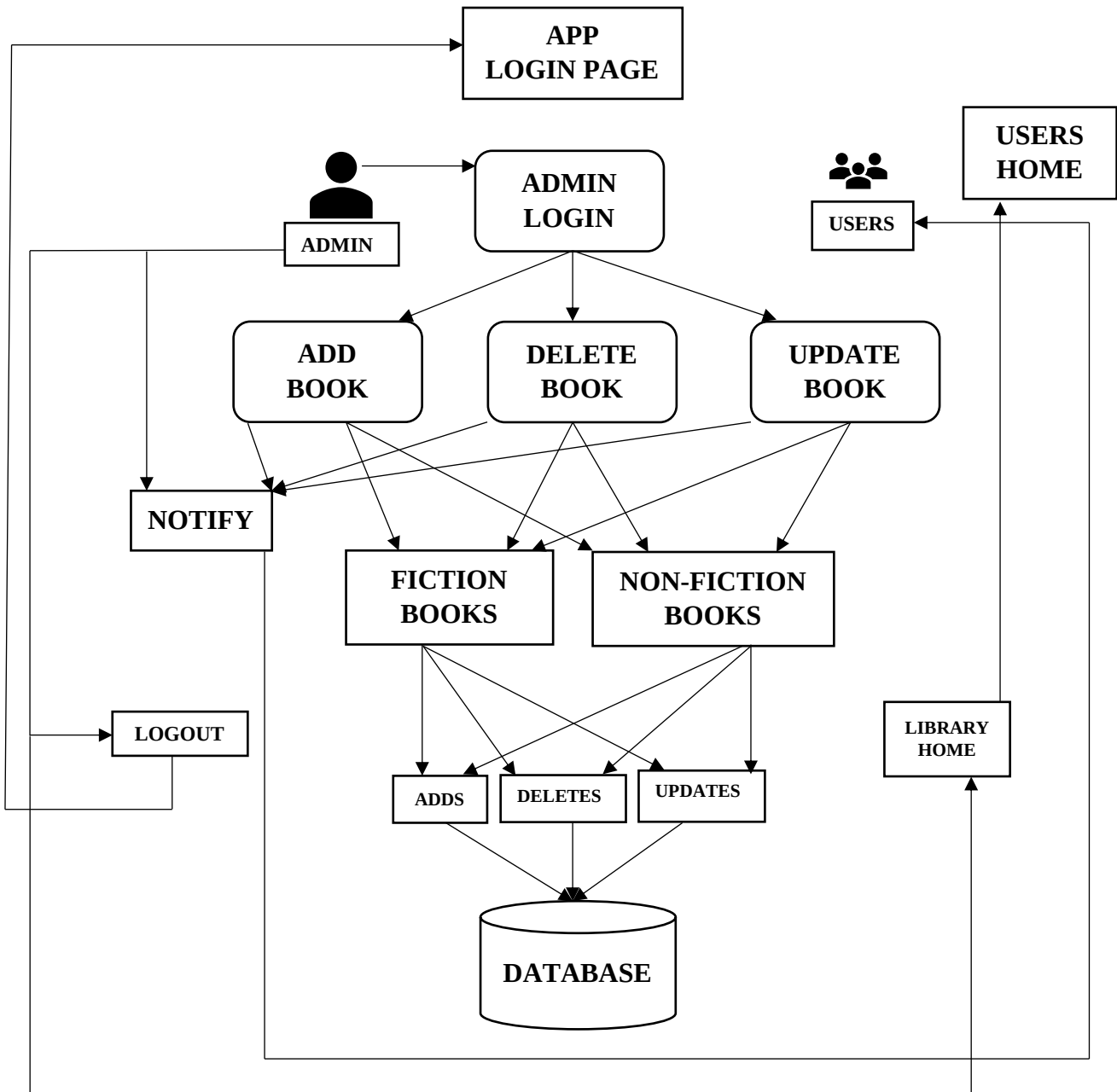


Fig 5.2: Admin architecture.

5.2 ADMIN ARCHITECTURE EXPLANATION:

1.APP LOGIN PAGE:

The app login page is the first or the launcher page of the library application.

2.ADMIN:

The admin is the one who administrates the library application, he/she monitors the application and has the authority to make changes to the library application.

3.ADMIN LOGIN:

The admin login is the page for the admin to login the application. From the login page the admin needs to provide their appropriate credentials in-order to view and use the application.

4.ADD BOOK:

The add book is the place where admin can be able to add or provide the details of the book within the application itself. The admin need not to sit in the database directly and work for hours in providing the data in to the database. This feature helps the admin to add data directly from the application itself and helps in reducing the time taken to work in the database.

5.DELETE BOOK:

The delete book is same in terms of feature compared to add book but here the work alone varies. The delete book is used to remove the particular book from the library. The admin simply needs to enter the book name in order to remove the book.

6.UPDATE BOOK:

The update book does the same thing as compared with addition and deletion of books. The update book is a useful tool to update the details of the existing book in the library.

7.FICTION & NON-FICTION BOOKS:

As we know the library application is separated into these two categories the flow of accessing, viewing and storing the data is also same for the admin. By this classification the admin can easily view and make changes to the application.

8.DATABASE:

The database is the storage area of the books. Whenever the admin performs some action such as adding, deleting and updating of the books it is connected with the database and the changes in the database can be made within the application itself.

9.LOGOUT:

The logout redirects the admin to the app login page. When the admin finishes the work from the application he can simply logout the application which will close the admin current page and opens the app login page that is the starting page of the application.

10.LIBRARY HOME:

Library home redirects the admin to view the users home screen from the admin page. It is useful as the admin can view or check the new changes made in the application.

11.NOTIFY:

The notify is connected with the three features such as add, delete and update. First the notify is nothing but making notifications for the application. Here admin can notify the users and staffs from the application itself. Whenever admin works on the library he can notify the users with the new changes made.

3.Basic Architecture:

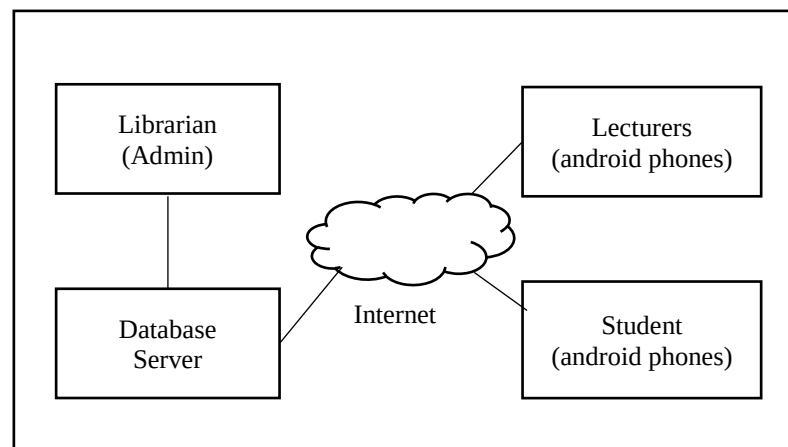


Fig 5.3: Basic architecture.

5.3 BASIC ARCHITECTURE EXPLANATION:

a.) Android Phones:

The android phone is the platform where we are going to use our application or the app is viewed through the android phones. The lecturers and the students are the primary users of the application through the mobile phones.

b.) Librarian (ADMIN):

The librarian is the admin or the staff responsible in making changes with the application such as modifying the books in the library.

c.) Database Server:

The database server is the one which can also be said as the provider here, as it is the one which is responsible to give out the details of the book to the users. The database is only permitted to the librarian and restricted to the other users as the librarian (Admin) is the one who manages the database with performing the modifiable tasks with the application. Here the librarian is directly connected to the database and from the database the data is sent through the internet and finally accessed by the users such as students and lecturers etc.

6.MODULES

There are several modules seen in the application:

- 1) Login & Register module.
- 2) Home module.
- 3) Admin module.
- 4) Fiction module.
- 5) Non-Fiction module.
- 6) Request module.
- 7) Add module.
- 8) Delete module.
- 9) Update module.
- 10) Notify module.

1.Login & Register module:

Only the registered students/staff can login. During registrations students are given an ID along with password through which they can login through their respective android phones. The login is used for authentication purpose and to make the application available for the authorized users only. The ID of the individual are stored in the database. The admin can add and delete the users account. This makes the application safer. The users are only allowed to view the application if the entered credentials are same and matching to that of in the database. The login can also be done through the app users individual google accounts as the app is integrated with the google sign-in method. This makes the application enabling further safety and easy way of gaining access to the application. The admin has separate way of logging into the application. They need to provide the appropriate credentials for gaining access to the application.

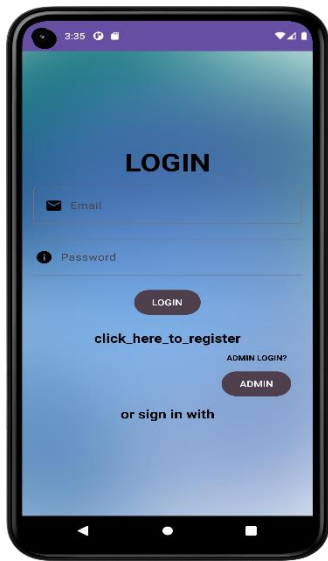


Fig 6.1: Login page.



Fig 6.2: Register page.



Fig 6.3: Google sign-in page.

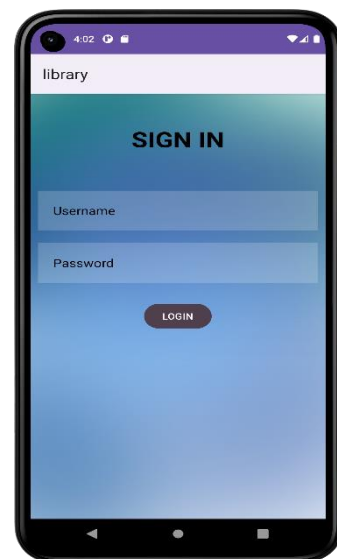


Fig 6.4: Admin login page.

2.Home module:

The home module is the user module which is the home Page of our library app. It has a very simple interface which can be easily accessible by the users. The home module Contains three categories to choose, if he/she chooses one from the three it will lead to another activity. The first button lets you View the page for fictional book genres and similarly second button Lets you view the page for non-fictional book genres and a separate button for making a request for books.

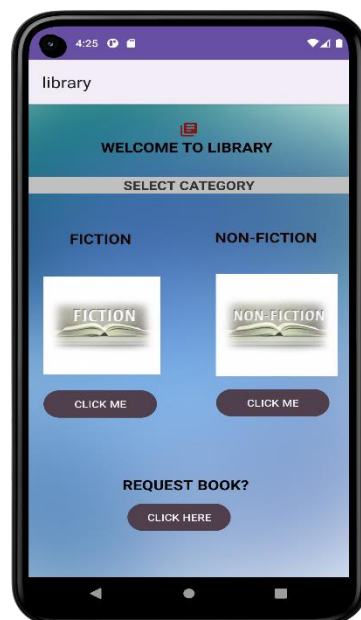


Fig 6.5: Home page.

3.Admin module:

The admin module is the module for the admins to work. The admins can do the major tasks performed in the database in the library app itself. To access the admin module the admin needs to login using their credentials. After logging in successfully the admin can start working in updating the library from the application itself.

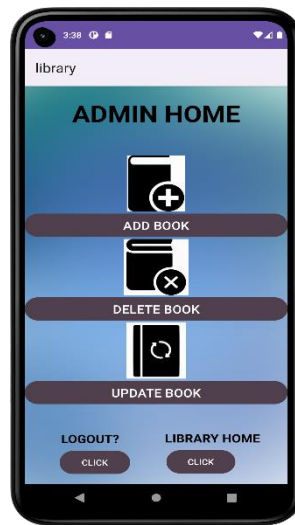


Fig 6.6: Admin page.

4.Fiction module:

The fictional module has the list of book genres available. The genres are uniquely related to the fictional type only. If the user wants to visit any genres they can simply click the names or button that will take the users to search page. The fiction genre shows the search page of the genres. The search will display the following details of the desired book. The author name, Bay, Call number, Floor, ISBN and the shelf. They will fetch the details from the firebase real time database storage.



Fig 6.7: Fiction page.

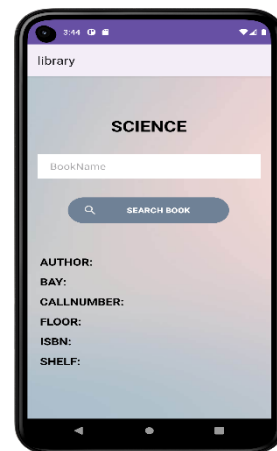


Fig 6.8: Fiction Genre page.

5.Non-Fiction module:

The Non-fictional module has the list of book genres available. The genres are uniquely related to the fictional type only. If the user wants to visit any genres they can simply click the names or button that will take the users to search page. The non-fiction genre shows the search page of the genres. The search will display the following details of the desired book. The author name, Bay, DDC, Floor, ISBN and the shelf. They will fetch the details from the firebase real time database storage. The difference between fiction and non-fiction is the call number and DDC way of identifying the books.

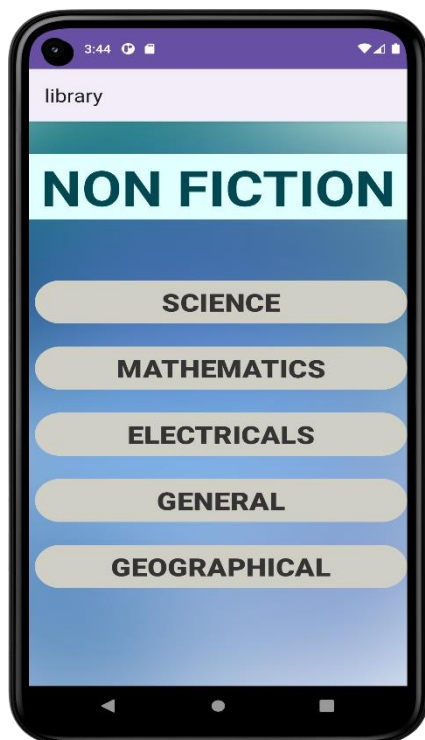


Fig 6.9: Non-Fiction page.

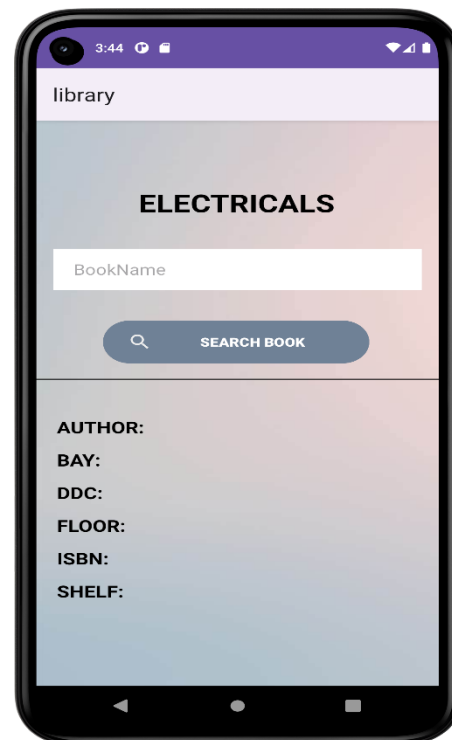


Fig 6.10: Non-Fiction Genre page.

6.Request module:

The request module is the place where the users or the staffs can make request of the needed books in the library. When the users can't find the needed books in the library they can make use of the request book. It is very simple to submit or make a request. The user or the staff

need to provide the details such as book name, author name, genre and the year of publishing to make a request. The request made are stored in the firebase real-time database.

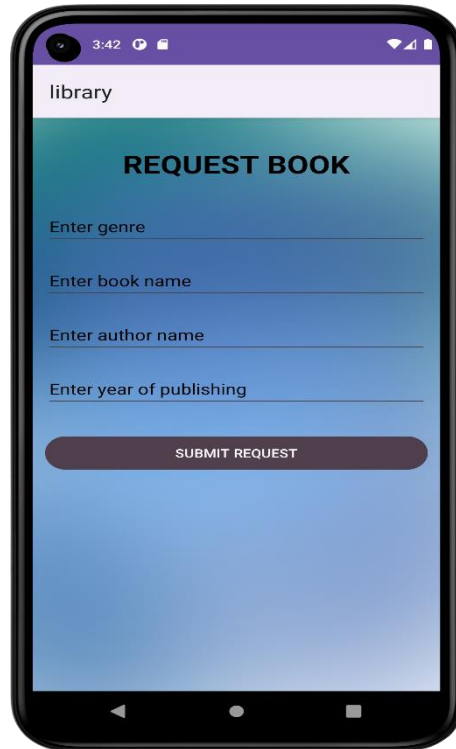


Fig 6.11: Request book page.

7.Add module:

The add module is used in the admin module. The admin is the one who can access the library and can make changes to the application. As in the name add module can be used to add or provide the new books to the library. The admin may not need to sit in the database for hours to do the work. The admin can easily add new books from the application itself, thus enabling ease of work and time consuming.

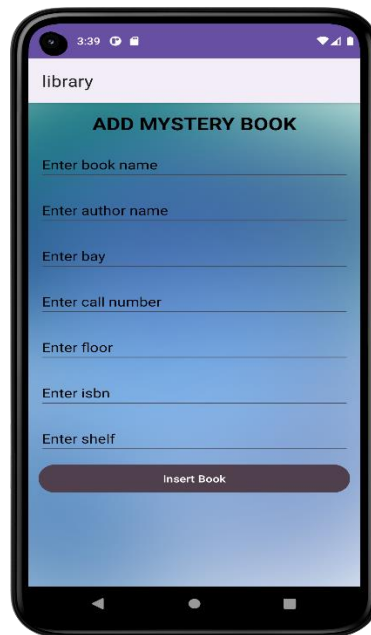


Fig 6.12: Add book page.

8.Delete module:

The delete module is also similar to the add module as both does the same work. The delete module is used to delete the book from the library. The book name alone is needed to perform the operation.

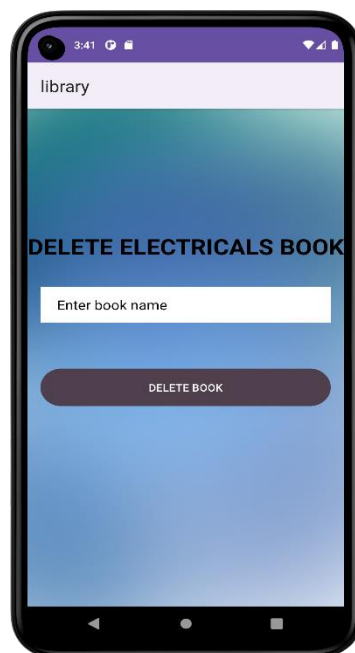


Fig 6.13: Delete book page.

9.Update module:

The update module is used to update the books which are existing books in the library. Whenever the books details are wrong and irrelevant or the actual data of the book is missing the update book can be used. This method will not create a new entry in the database but it simply changes or modify the existing data in the database. It is useful and time consuming when it is done through the application rather than working on the database. When the data is filled and on clicking update button it simply updates the book with new data given.

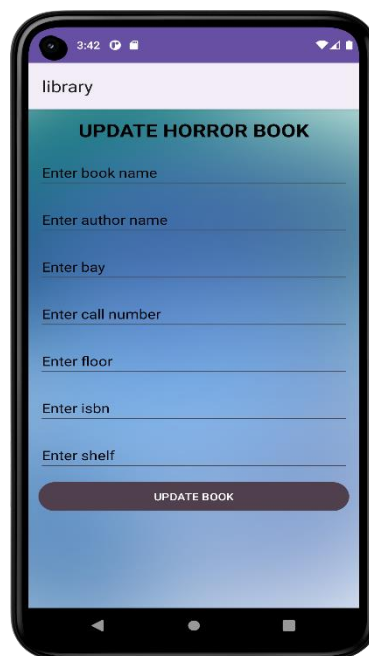


Fig 6.14: Update book page.

10.Notify module:

The notify is the notifications which are sent to the application. The admin or the app developer can able to communicate the users of the application by using this notification. In simple terms the library application can communicate with its users through the notification. It is the short messages that are displayed on the outside of the apps interface, thus whoever uses this application they can be notified with

certain message sent from the admin. The notify can be done in two ways internal and external way of notification.

INTERNAL NOTIFICATION:

The internal notification is where the messages are made from the internal that is from the application itself. The admin is the one with authority to send messages thus it is integrated with three features in the admin module such as add, delete and update book. The admin can notify the users whenever a change is made in the database like a book is added or removed the users are notified.

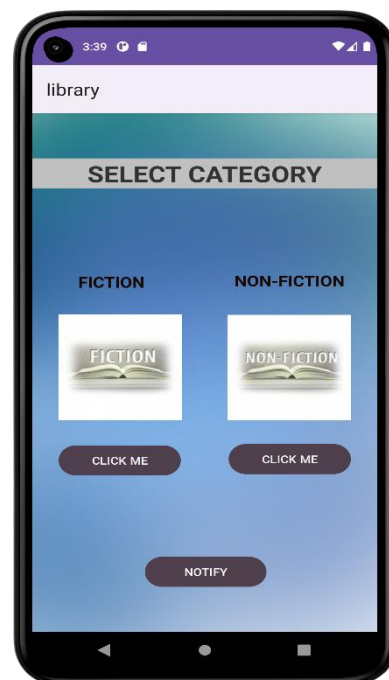
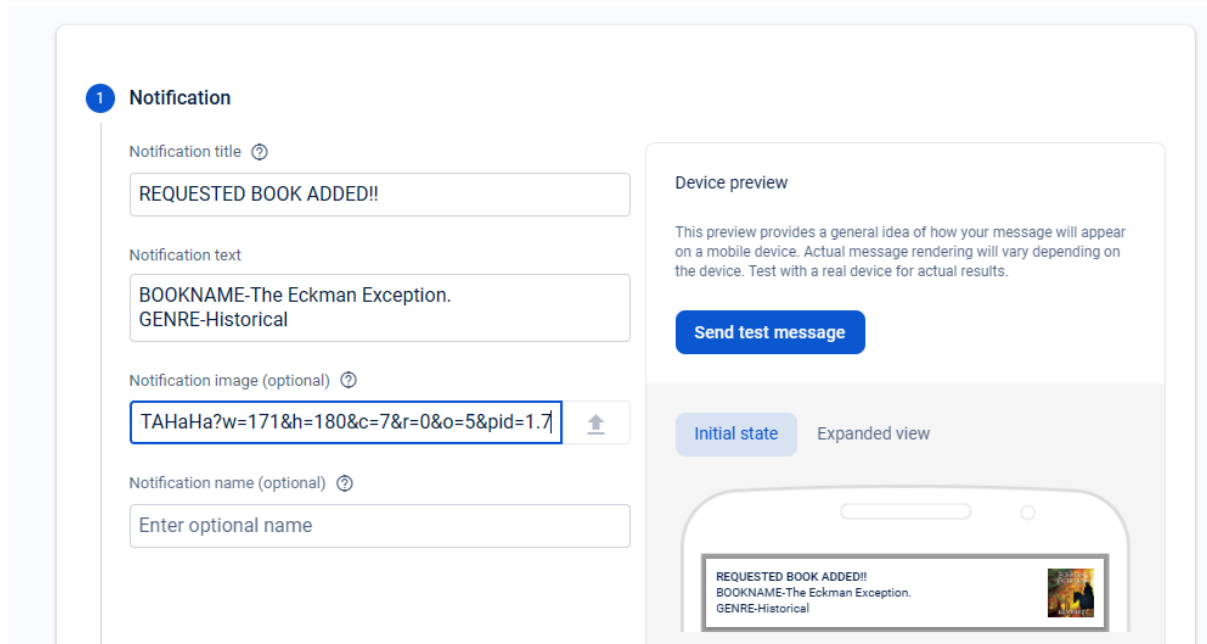


Fig 6.15: Internal Notification page.

EXTERNAL NOTIFICATION:

The external notification is the one where the notification can be made for the application through a different communication channel such as **FCM**- Firebase Cloud Messaging. The FCM provides the application to communicate with its users by means of notification. The FCM can be used to send instant messages to the users of

application. The admin can communicate with their users by providing notifications through FCM.



The screenshot shows the 'Notification' configuration page in the Firebase console. It includes several input fields for configuring a notification: 'Notification title' (containing 'REQUESTED BOOK ADDED!!'), 'Notification text' (containing 'BOOKNAME-The Eckman Exception. GENRE-Historical'), 'Notification image (optional)' (containing a placeholder URL), and 'Notification name (optional)' (with a placeholder 'Enter optional name'). To the right, there is a 'Device preview' section with a 'Send test message' button and a visual representation of how the notification will appear on a mobile device. The preview shows the title, text, and a small book cover image.

Fig 6.16: External Notification (FCM).

7.RESULT AND DISCUSS

Result 1: Searching fiction and non-fiction books.

The user must give their title or the book name as an input to the search. The search will start once the user enters the title they will request the service from the database for the given input and if the book details are found in the database it will return the valid book details. If the book is not found it will just simply toast a message that the book doesn't exist. The users can search books according to the genre they require.

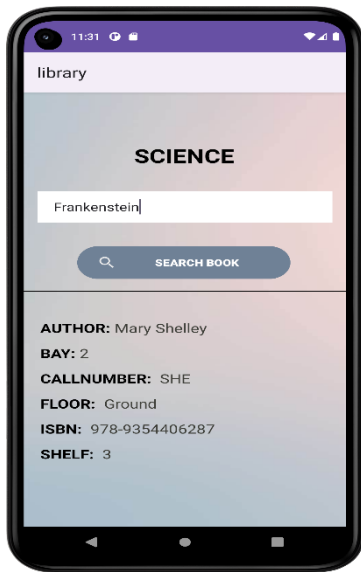


FIG 7.1: Fiction book details.

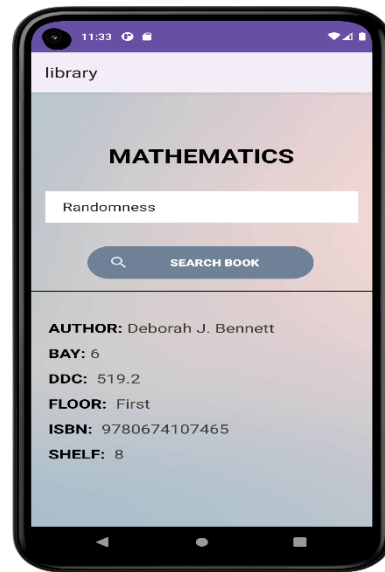


FIG 7.2: Non-Fiction book details.

Result 2: Submitting a book request.

The users can make a request of the book through this request book area. For making a request the user needs to make sure they provide the details of the book for making a request. The details such as book name, author name, genre and year of publishing are required to make a request.



FIG 7.3: Requesting book.

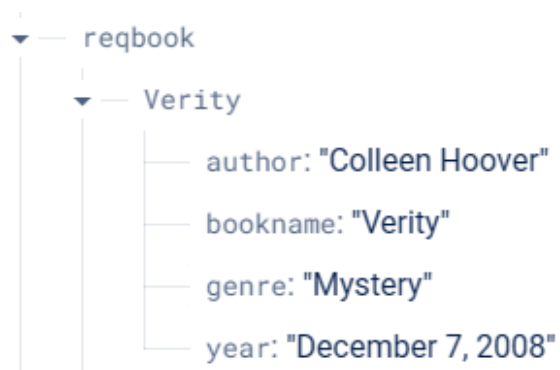


FIG 7.4: Database for request.

Result 3: Adding, deleting and updating books.

The operations like adding, deleting and updating books from the database can be done from the application by the admin. The adding of new books will be done after giving the details of the book and when clicking the insert button, the new book will be stored in the database. For deletion the book name alone is required and the updating of books will take place in the existing book where we can change the details of the existing book if the details are irrelevant.

ADDING:

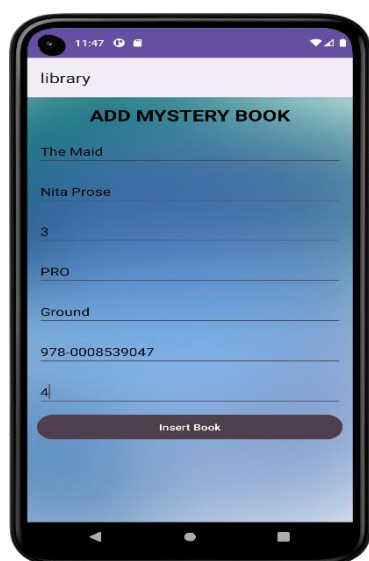


FIG 7.5: Adding book.



FIG 7.6: Book stored in database.

DELETING:

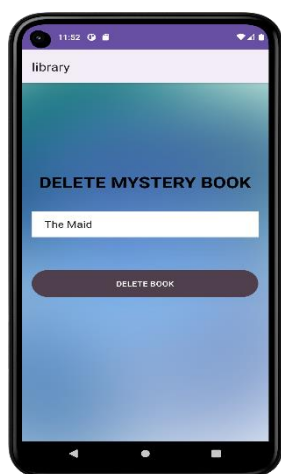


FIG 7.7: Deleting book.



FIG 7.8: Book deleted.

UPDATING:

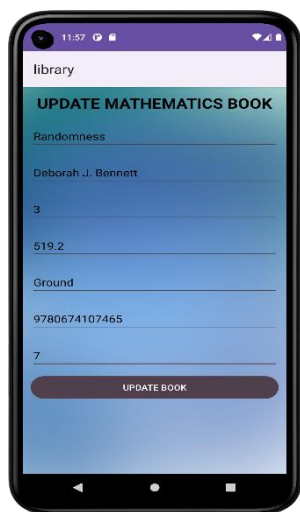


FIG 7.9: Updating book.

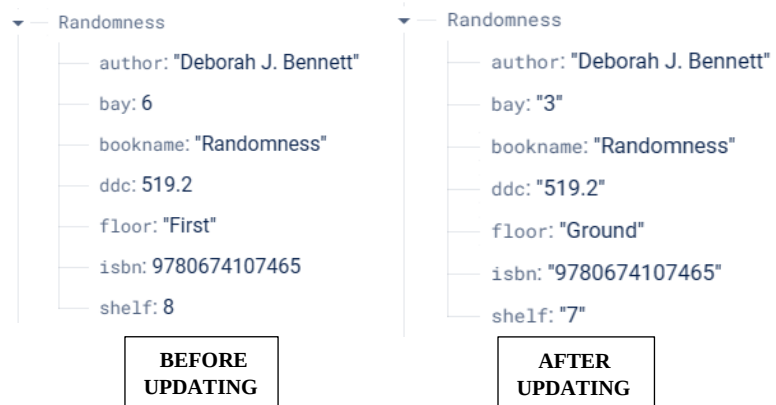


FIG 7.10: Updated book in database.

Result 4: Making notification for the application.

The notification for the application is the way of communication passed between the users and the application. The admin can be able to make notification to the users by two ways.

- ❖ Internal notification.
- ❖ External notification.

INTERNAL NOTIFICATION:

The internal notification is provided in the application in three areas like adding, deleting and updating in the admin module. The admin can be able to notify the users internally through the application. Whenever a new book is added or operations like delete and update are performed the admin can notify the users by clicking the notify.

EXTERNAL NOTIFICATION:

The external notification is provided to the application through the (FCM) externally. The admin can make use of the firebase cloud messaging to communicate with the users of the application. The admin can be able to pass messages and updated regarding the library app to the users.

INTERNAL:

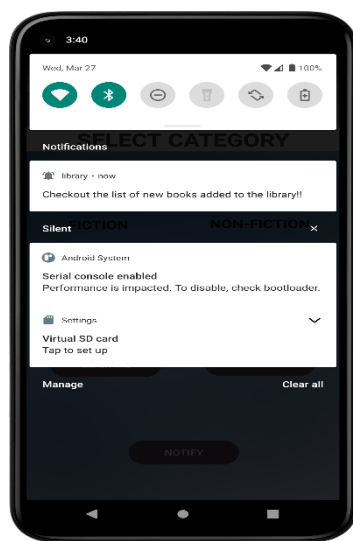


FIG 7.11: Internal notification.



FIG 7.12: Notification list.

EXTERNAL:

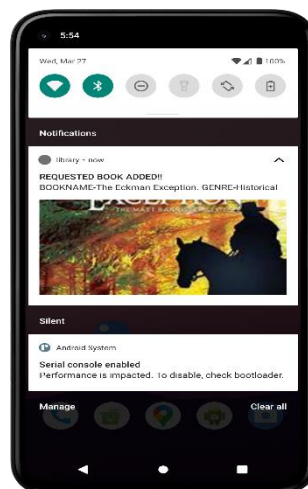


FIG 7.13: External notification (FCM).

Result 5: Integrating google sign-in.

The google sign-in method is a convenient way for the users to authenticate themselves on the android application using their own google credentials. User's are taken to Google's authentication system after choosing Sign In With Google. They give the platform permission to disseminate the details on their profile. Google securely exchanges pertinent information with the website or app, like name, email address, and profile image.

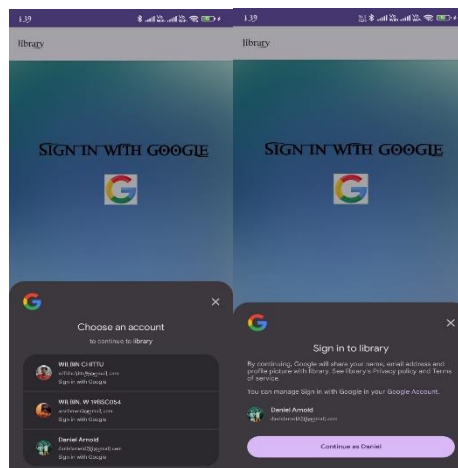


FIG 7.14: Google sign-in.

8.CONCLUSION

This project has the details for the android application which named as android app for searching books in a library. It is the application where android studio and firebase is used to store, build, communicate and search books. The users can also use their google account for accessing the application. The main aim of this application is to enable user friendly method for searching the books in a library. This application is greatly useful for the students and staffs who access the

library regularly. It aims in reducing the time taken to perform the searching of books manually and also, we can locate many books in the library from a single place. The admin can also add new books, delete and update from the application itself. The communication for the application is enabled through notification in two ways. Users can also request books which they needed in the library. Thus, these features enable's the fast access of the library.

9.CODING

9.1 Sample codes:

1. 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"
    android:padding="15dp"
    android:gravity="center"
    android:background="@drawable/bac"
    tools:context=".login">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:gravity="center"
        android:text="@string/LOGIN"
        android:textColor="#000"
        android:textSize="40sp"
        android:textStyle="bold" />

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
```

```

        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:drawableLeft="@drawable/baseline_email_24"
        android:drawablePadding="10dp"
        android:hint="@string/email" />
</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:layout_width="389dp"
    android:layout_height="wrap_content">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:drawableLeft="@drawable/baseline_info_24"
        android:drawablePadding="10dp"
        android:hint="@string/password" />
</com.google.android.material.textfield.TextInputLayout>

<ProgressBar
    android:id="@+id/progressBar"
    android:visibility="gone"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<Button
    android:id="@+id/btnlogin"
    android:text="@string/LOGIN"
    android:textColor="@color/white"
    android:backgroundTint="#50404d"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

<TextView
    android:textStyle="bold"
    android:textSize="20sp"
    android:gravity="center"
    android:layout_marginTop="20dp"
    android:id="@+id/registerNow"
    android:text="click_here_to_register"
    android:textColor="#000"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>

<TextView
    android:layout_width="match_parent"

```

```

        android:layout_height="wrap_content"
        android:layout_marginLeft="120dp"
        android:layout_marginTop="30px"
        android:gravity="center"
        android:text="ADMIN LOGIN?"
        android:textColor="#000"
        android:textSize="12sp"
        android:textStyle="bold" />

```

```

<Button
    android:id="@+id/btnadmin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="120dp"
    android:layout_marginTop="10dp"
    android:backgroundTint="#50404d"
    android:text="ADMIN"
    android:textColor="@color/white" />

```

```

<TextView
    android:id="@+id/orsignin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="30px"
    android:gravity="center"
    android:text="or sign in with"
    android:textColor="#000"
    android:textSize="20sp"
    android:textStyle="bold" />

```

```

</LinearLayout>

```

2.Login.java

```

package com.example.library;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

```

```

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.messaging.FirebaseMessaging;

public class login extends AppCompatActivity {
    TextInputEditText editTextEmail, editTextPassword;
    Button buttonLogin,admin;
    FirebaseAuth mAuth;
    ProgressBar progressBar;
    TextView textView,orsign;

    @Override
    public void onStart() {
        super.onStart();
        FirebaseUser currentUser = mAuth.getCurrentUser();
        if(currentUser != null){
            Intent intent = new Intent(getApplicationContext(), login.class);
            startActivity(intent);
            finish();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        mAuth=FirebaseAuth.getInstance();
        editTextEmail=findViewById(R.id.email);
        editTextPassword=findViewById(R.id.password);
        buttonLogin=findViewById(R.id.btnlogin);
        admin=findViewById(R.id.btnadmin);
        progressBar=findViewById(R.id.progressBar);
        textView =findViewById(R.id.registerNow);
        orsign=findViewById(R.id.orsignin);

        FirebaseMessaging.getInstance().subscribeToTopic("News")
            .addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    String msg = "Done";
                    if (!task.isSuccessful()) {
                        msg = "Failed";
                    }
                }
            });
    }
}

```

```

    }
});

admin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), adminlogin.class);
        startActivity(intent);
        finish();
    }
});

orsign.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), googlesign.class);
        startActivity(intent);
        finish();
    }
});

textView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), registration.class);
        startActivity(intent);
        finish();
    }
});

buttonLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        progressBar.setVisibility(View.VISIBLE);
        String email,password;
        email=String.valueOf(editTextEmail.getText());
        password=String.valueOf(editTextPassword.getText());

        if(TextUtils.isEmpty(email)){
            Toast.makeText(login.this, "Enter email", Toast.LENGTH_SHORT).show();
            return;
        }

        if(TextUtils.isEmpty(password)){
            Toast.makeText(login.this, "Enter password", Toast.LENGTH_SHORT).show();
            return;
        }
    }
});

```

```

        mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    progressBar.setVisibility(View.GONE);
                    if (task.isSuccessful()) {
                        Toast.makeText(getApplicationContext(), "Login Successful.",
                            Toast.LENGTH_SHORT).show();
                        Intent intent = new Intent(getApplicationContext(), home.class);
                        startActivity(intent);
                        finish();
                    } else {
                        Toast.makeText(login.this, "Authentication failed.",
                            Toast.LENGTH_SHORT).show();
                    }
                }
            });
    }
}
}
}

```

3.Homeactivity.xml

```

<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"
    android:background="@drawable/bac">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:drawableTop="@drawable/baseline_library_books_24"
        android:gravity="center"
        android:text="WELCOME TO LIBRARY "
        android:textColor="#000"
        android:textSize="20dp"
        android:textStyle="bold"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:background="#c0c0c0"
        android:gravity="center"

```

```
android:text="SELECT CATEGORY"
android:textColor="#333333"
android:textSize="18dp"
android:textStyle="bold" />
```

```
<RelativeLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
```

```
<TextView
```

```
    android:id="@+id/fict"
    android:layout_width="93dp"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="45dp"
    android:layout_marginLeft="60dp"
```

```
    android:layout_marginTop="52dp"
    android:layout_marginRight="250dp"
    android:layout_weight="1"
    android:text=" FICTION"
    android:textColor="#180703"
    android:textSize="20dp"
    android:textStyle="bold" />
```

```
<TextView
```

```
    android:id="@+id/non"
    android:layout_width="154dp"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_marginLeft="230dp"
    android:layout_marginTop="50dp"
    android:layout_marginEnd="10dp"
    android:layout_marginRight="25dp"
    android:layout_weight="1"
    android:text="NON-FICTION"
    android:textColor="#180703"
    android:textSize="20dp"
    android:textStyle="bold" />
```

```
<ImageView
```

```
    android:id="@+id/imageView"
    android:layout_width="143dp"
    android:layout_height="193dp"
    android:layout_below="@id/fict"
    android:layout_alignParentStart="true"
    android:layout_marginStart="18dp"
```

```
    android:layout_marginLeft="40dp"
    android:layout_marginTop="17dp"
    android:adjustViewBounds="true"
    app:srcCompat="@drawable/fiction1" />
```

```
<ImageView
    android:id="@+id/imageview2"
    android:layout_width="149dp"
    android:layout_height="211dp"
    android:layout_below="@id/non"
    android:layout_alignParentRight="true"
    android:layout_marginLeft="40dp"
    android:layout_marginTop="9dp"
    android:layout_marginRight="14dp"
    android:layout_marginBottom="-281dp"
    android:adjustViewBounds="true"
    app:srcCompat="@drawable/fiction2" />
```

```
<Button
    android:id="@+id/button4"
    android:layout_width="139dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/imageView"
    android:layout_alignParentStart="true"
    android:layout_marginStart="18dp"
    android:layout_marginLeft="40dp"
    android:layout_marginTop="-6dp"
    android:backgroundTint="#50404d"
    android:text="CLICK ME"
    android:textColor="#f5f5f5" />
```

```
<Button
    android:id="@+id/button5"
    android:layout_width="139dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/non"
    android:layout_alignRight="@id/button4"
    android:layout_alignParentRight="true"
    android:layout_marginTop="204dp"
    android:layout_marginRight="24dp"
    android:backgroundTint="#50404d "
    android:text="CLICK ME"
    android:textColor="#f5f5f5" />
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentEnd="true"
```

```

        android:layout_alignParentBottom="true"
        android:layout_marginStart="115dp"
        android:layout_marginEnd="120dp"
        android:layout_marginBottom="121dp"
        android:text="REQUEST BOOK?"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold" />

<Button
    android:id="@+id/buttonreq"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="145dp"
    android:layout_marginBottom="63dp"
    android:backgroundTint="#50404d "
    android:text="CLICK HERE"
    android:textColor="#f5f5f5" />
</RelativeLayout>
</LinearLayout>

```

4.Homeactivity.java

```

package com.example.library;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.widget.TintableCheckedTextView;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.messaging.FirebaseMessaging;

public class home extends AppCompatActivity {
    Button btnfiction, btnnonfiction, btnreq;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        btnfiction = findViewById(R.id.button4);
        btnnonfiction = findViewById(R.id.button5);
        btnreq = findViewById(R.id.buttonreq);

        btnfiction.setOnClickListener(new View.OnClickListener() {

```

```

        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getApplicationContext(), fiction.class);
            startActivity(intent);
        }
    });

    btnnonfiction.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getApplicationContext(), nonfiction.class);
            startActivity(intent);
        }
    });

    btnreq.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(getApplicationContext(), requestbook.class);
            startActivity(intent);
        }
    });
}
}

```

5.Fiction.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=".fiction"
    android:background="@drawable/bac">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="FICTION"
        android:background="#e0ffff"
        android:gravity="center"
        android:textSize="60sp"
        android:textStyle="bold"
        android:textColor="#004953"
        android:layout_marginTop="40dp"/>

    <Button
        android:id="@+id/button6"

```

```
android:layout_width="397dp"
android:layout_height="wrap_content"
android:layout_alignParentStart="true"
android:layout_marginStart="6dp"
android:layout_marginLeft="80dp"
android:layout_marginTop="190dp"
android:gravity="center"
android:text="HISTORICAL"
android:backgroundTint="#cfcfc4"
android:textColor="#343434"
android:textSize="30sp"
android:textStyle="bold" />
```

<Button

```
android:id="@+id/button7"
android:layout_below="@+id/button6"
android:layout_width="397dp"
android:layout_height="wrap_content"
android:layout_alignParentStart="true"
android:layout_marginStart="6dp"
android:layout_marginLeft="80dp"
android:layout_marginTop="20dp"
android:gravity="center"
android:text="SCIENCE"
android:backgroundTint="#cfcfc4"
android:textColor="#343434"
android:textSize="30sp"
android:textStyle="bold" />
```

<Button

```
android:id="@+id/button8"
android:layout_below="@+id/button7"
android:layout_width="397dp"
android:layout_height="wrap_content"
android:layout_alignParentStart="true"
android:layout_marginStart="6dp"
android:layout_marginLeft="80dp"
android:layout_marginTop="20dp"
android:gravity="center"
android:text="MYSTERY"
android:backgroundTint="#cfcfc4"
android:textColor="#343434"
android:textSize="30sp"
android:textStyle="bold" />
```

<Button

```
android:id="@+id/button9"
android:layout_below="@+id/button8"
android:layout_width="397dp"
android:layout_height="wrap_content"
```

```

        android:layout_alignParentStart="true"
        android:layout_marginStart="6dp"
        android:layout_marginLeft="80dp"
        android:layout_marginTop="20dp"
        android:gravity="center"
        android:text="HORROR"
        android:backgroundTint="#cfcfc4"
        android:textColor="#343434"
        android:textSize="30sp"
        android:textStyle="bold" />

```

```

<Button
    android:id="@+id/button10"
    android:layout_below="@+id/button9"
    android:layout_width="397dp"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_marginStart="6dp"
    android:layout_marginLeft="80dp"
    android:layout_marginTop="20dp"
    android:gravity="center"
    android:text="FANTASY"
    android:backgroundTint="#cfcfc4"
    android:textColor="#343434"
    android:textSize="30sp"
    android:textStyle="bold" />

```

```

</RelativeLayout>

```

6.Fiction.java

```

package com.example.library;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
public class fiction extends AppCompatActivity {
    Button histbtn,sciencebtn,mysterybtn,horrorbtn,fantasybtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fiction);
        histbtn=findViewById(R.id.button6);
        sciencebtn=findViewById(R.id.button7);
        mysterybtn=findViewById(R.id.button8);
        horrorbtn=findViewById(R.id.button9);
        fantasybtn=findViewById(R.id.button10);
    }
}

```

```

histbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), historical.class);
        startActivity(intent);
    }
});

sciencebtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), sciencefict.class);
        startActivity(intent);
    }
});

mysterybtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), mystery.class);
        startActivity(intent);
    }
});

horrorbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), horror.class);
        startActivity(intent);
    }
});

fantasybtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), fantasy.class);
        startActivity(intent);
    }
});

}
}

```

7.Googlesign-in.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"
    android:background="@drawable/bac"
    tools:context=".googlesign">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/others"
        android:text=" SIGN IN WITH GOOGLE"
        android:textStyle="bold"
        android:textSize="35sp"
        android:layout_marginTop="150dp"
        android:textColor="@color/black"
        android:layout_centerHorizontal="true"/>

    <ImageView
        android:id="@+id/google_btn"
        android:layout_width="58dp"
        android:layout_height="58dp"
        android:layout_below="@+id/others"
        android:layout_alignStart="@+id/others"
        android:layout_alignEnd="@+id/others"
        android:layout_margin="20dp"
        android:layout_marginStart="30dp"
        android:layout_marginTop="90dp"
        android:layout_marginEnd="11dp"
        android:layout_marginBottom="20dp"
        android:src="@drawable/googleimg" />

</RelativeLayout>

```

8.Googlesign-in.java

```

package com.example.library;
import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.IntentSenderRequest;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

```

```

import android.app.Activity;
import android.content.IntentSender;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.Toast;

import com.google.android.gms.auth.api.identity.BeginSignInRequest;
import com.google.android.gms.auth.api.identity.BeginSignInResult;
import com.google.android.gms.auth.api.identity.Identity;
import com.google.android.gms.auth.api.identity.SignInClient;
import com.google.android.gms.auth.api.identity.SignInCredential;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;

public class googlesign extends AppCompatActivity {
    SignInClient oneTapClient;
    BeginSignInRequest signUpRequest;
    ImageView google;
    private static final int REQ_ONE_TAP = 2; // Can be any integer unique to the
Activity.
    private boolean showOneTapUI = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_googlesign);

        google=findViewById(R.id.google_btn);
        oneTapClient = Identity.getSignInClient(this);
        signUpRequest = BeginSignInRequest.builder()

.setGoogleIdTokenRequestOptions(BeginSignInRequest.GoogleIdTokenRequestOptions.
builder()
        .setSupported(true)
        // Your server's client ID, not your Android client ID.
        .setServerClientId(getString(R.string.web_client_id))
        // Show all accounts on the device.
        .setFilterByAuthorizedAccounts(false)
        .build())
        .build();

        ActivityResultLauncher<IntentSenderRequest> activityResultLauncher =

```

```

        registerForActivityResult(new
ActivityResultContracts.StartIntentSenderForResult(), new
ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == Activity.RESULT_OK) {
                    try {
                        SignInCredential credential =
oneTapClient.getSignInCredentialFromIntent(result.getData());
                        String idToken = credential.getIdToken();
                        if (idToken != null) {
                            String email = credential.getId();
                            Toast.makeText(getApplicationContext(), "Email: "+email,
Toast.LENGTH_SHORT).show();
                        }
                    } catch (ApiException e) {
                        e.printStackTrace();
                    }
                }
            }
        });
        google.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                oneTapClient.beginSignIn(signUpRequest)
                    .addOnSuccessListener(googleSign.this, new
OnSuccessListener<BeginSignInResult>() {
                        @Override
                        public void onSuccess(BeginSignInResult result) {
                            IntentSenderRequest intentSenderRequest =
new
IntentSenderRequest.Builder(result.getPendingIntent().getIntentSender()).build();
                            activityResultLauncher.launch(intentSenderRequest);
                        }
                    })
                    .addOnFailureListener(googleSign.this, new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            // No Google Accounts found. Just continue presenting the signed-out
UI.
                            Log.d("TAG", e.getLocalizedMessage());
                        }
                    });
            }
        });
    }
});

```

```
}  
}
```

9.FirebaseMessagingService.java

```
package com.example.library;
```

```
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.media.RingtoneManager;  
import android.net.Uri;  
import android.os.Build;
```

```
import androidx.core.app.NotificationCompat;
```

```
import com.google.firebase.messaging.FirebaseMessagingService;  
import com.google.firebase.messaging.RemoteMessage;
```

```
public class MyFirebaseMessagingService extends FirebaseMessagingService {
```

```
    private static final String TAG = "MyFirebaseMsgService";
```

```
    @Override
```

```
    public void onMessageReceived(RemoteMessage remoteMessage) {  
        // TODO(developer): Handle FCM messages here.
```

```
        if (remoteMessage.getNotification() != null) {  
            // Show the notification  
            String notificationBody = remoteMessage.getNotification().getBody();  
            String notificationTitle = remoteMessage.getNotification().getTitle();  
  
            sendNotification(notificationTitle, notificationBody);
```

```
        }  
    }
```

```
    private void sendNotification(String title, String body) {  
        Intent intent = new Intent(this, login.class);  
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

```

        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */,
intent,
        PendingIntent.FLAG_IMMUTABLE);

        String channelId = "fcm_default_channel";
        Uri defaultSoundUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(this, channelId)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle(title)
                .setContentText(body)
                .setAutoCancel(true)
                .setSound(defaultSoundUri)
                .setContentIntent(pendingIntent);

        NotificationManager notificationManager =
            (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

        // Since android Oreo notification channel is needed.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(channelId,
                "Channel human readable title",
                NotificationManager.IMPORTANCE_DEFAULT);
            notificationManager.createNotificationChannel(channel);
        }

        notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
    }

}

```

10.Build.gradle.kts (:app)

```

plugins {
    id("com.android.application")
    id("com.google.gms.google-services")
}

android {
    namespace = "com.example.library"
    compileSdk = 33

```

```

defaultConfig {
    applicationId = "com.example.library"
    minSdk = 24
    targetSdk = 33
    versionCode = 1
    versionName = "1.0"

    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
}

buildTypes {
    release {
        isMinifyEnabled = false
        proguardFiles(getDefaultProguardFile("proguard-android-optimize.txt"),
"proguard-rules.pro")
    }
}
buildFeatures {

    viewBinding = true
}
compileOptions {
    sourceCompatibility = JavaVersion.VERSION_1_8
    targetCompatibility = JavaVersion.VERSION_1_8
}
}

dependencies {

    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.9.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    implementation("de.hdodenhof:circleimageview:3.1.0")
    implementation("com.github.bumptech.glide:glide:4.16.0")
    implementation("androidx.cardview:cardview:1.0.0")
    implementation("androidx.recyclerview:recyclerview:1.3.1")
    implementation("com.google.android.gms:play-services-auth:20.7.0")
    implementation(platform("com.google.firebase:firebase-bom:32.3.1"))
    implementation("com.firebaseui:firebase-ui-database:8.0.2")
    implementation("com.google.firebase:firebase-auth:22.3.0")
    implementation("com.google.firebase:firebase-database:20.3.0")
    implementation("com.google.firebase:firebase-messaging")
    implementation("com.google.firebase:firebase-messaging:23.4.1")
    testImplementation("junit:junit:4.13.2")
    implementation("com.google.firebase:firebase-core:21.1.1")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
}

```

```

    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}

```

11.AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Material3.Light"
        tools:targetApi="30">
        <activity
            android:name=".registertoken"
            android:exported="false" />
        <activity
            android:name=".updnotify"
            android:exported="false" />
        <activity
            android:name=".delnotify"
            android:exported="false" />
        <activity
            android:name=".addnotify"
            android:exported="false" />
        <activity
            android:name=".requestbook"
            android:exported="false" />
        <activity
            android:name=".delgeographical"
            android:exported="false" />
        <activity
            android:name=".delgeneral"
            android:exported="false" />
        <activity

```

```

        android:name=".delectricals"
        android:exported="false" />
<activity
    android:name=".delmathematics"
    android:exported="false" />
<activity
    android:name=".delsciencenonfic"
    android:exported="false" />
<activity
    android:name=".delfantasy"
    android:exported="false" />
<activity
    android:name=".delhorror"
    android:exported="false" />
<activity
    android:name=".delmystery"
    android:exported="false" />
<activity
    android:name=".delsciencefic"
    android:exported="false" />
<activity
    android:name=".delhistory"
    android:exported="false" />
<activity
    android:name=".delnonfiction"
    android:exported="false" />
<activity
    android:name=".delfiction"
    android:exported="false" />
<activity
    android:name=".delhome"
    android:exported="false" />
<activity
    android:name=".updgeographical"
    android:exported="false" />
<activity
    android:name=".updgeneral"
    android:exported="false" />
<activity
    android:name=".updelectricals"
    android:exported="false" />
<activity
    android:name=".updmathematics"
    android:exported="false" />
<activity
    android:name=".updfantasy"

```

```

        android:exported="false" />
<activity
    android:name=".updhorrer"
    android:exported="false" />
<activity
    android:name=".updmystery"
    android:exported="false" />
<activity
    android:name=".updsclencenonfic"
    android:exported="false" />
<activity
    android:name=".updsclencefic"
    android:exported="false" />
<activity
    android:name=".updhistory"
    android:exported="false" />
<activity
    android:name=".updnonfiction"
    android:exported="false" />
<activity
    android:name=".updfiction"
    android:exported="false" />
<activity
    android:name=".updhome"
    android:exported="false" />
<activity
    android:name=".addgeographic"
    android:exported="false" />
<activity
    android:name=".addgeneral"
    android:exported="false" />
<activity
    android:name=".addelectricals"
    android:exported="false" />
<activity
    android:name=".mathematicsadd"
    android:exported="false" />
<activity
    android:name=".addscclencenonfic"
    android:exported="false" />
<activity
    android:name=".fantasyadd"
    android:exported="false" />
<activity
    android:name=".horreradd"
    android:exported="false" />

```

```

<activity
    android:name=".mysteryadd"
    android:exported="false" />
<activity
    android:name=".addsciencefic"
    android:exported="false" />
<activity
    android:name=".historicaladd"
    android:exported="false" />
<activity
    android:name=".addnonfiction"
    android:exported="false" />
<activity
    android:name=".addfiction"
    android:exported="false" />
<activity
    android:name=".addhome"
    android:exported="false" />
<activity
    android:name=".adminhome"
    android:exported="false" />
<activity
    android:name=".adminlogin"
    android:exported="false" />
<activity
    android:name=".googlesign"
    android:exported="false" />
<activity
    android:name=".geographical"
    android:exported="false" />
<activity
    android:name=".general"
    android:exported="false" />
<activity
    android:name=".electricals"
    android:exported="false" />
<activity
    android:name=".mathematics"
    android:exported="false" />
<activity
    android:name=".sciencenonfic"
    android:exported="false" />
<activity
    android:name=".fantasy"
    android:exported="false" />
<activity

```

```

        android:name=".horror"
        android:exported="false" />
    <activity
        android:name=".mystery"
        android:exported="false" />
    <activity
        android:name=".sciencefict"
        android:exported="false" />
    <activity
        android:name=".historical"
        android:exported="false" />
    <activity
        android:name=".nonfiction"
        android:exported="false" />
    <activity
        android:name=".fiction"
        android:exported="false" />
    <activity
        android:name=".home"
        android:exported="false" />
    <activity
        android:name=".registration"
        android:exported="false"
        android:theme="@style/Theme.Material3.Light.NoActionBar" />
    <activity
        android:name=".login"
        android:exported="true"
        android:theme="@style/Theme.Material3.Light.NoActionBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".MainActivity"
        android:exported="true" />
    <service android:name=".MyFirebaseMessagingService"
        android:exported="false">
        <intent-filter>
            <action android:name="com.google.firebase.MESSAGING_EVENT" />
        </intent-filter>
    </service>

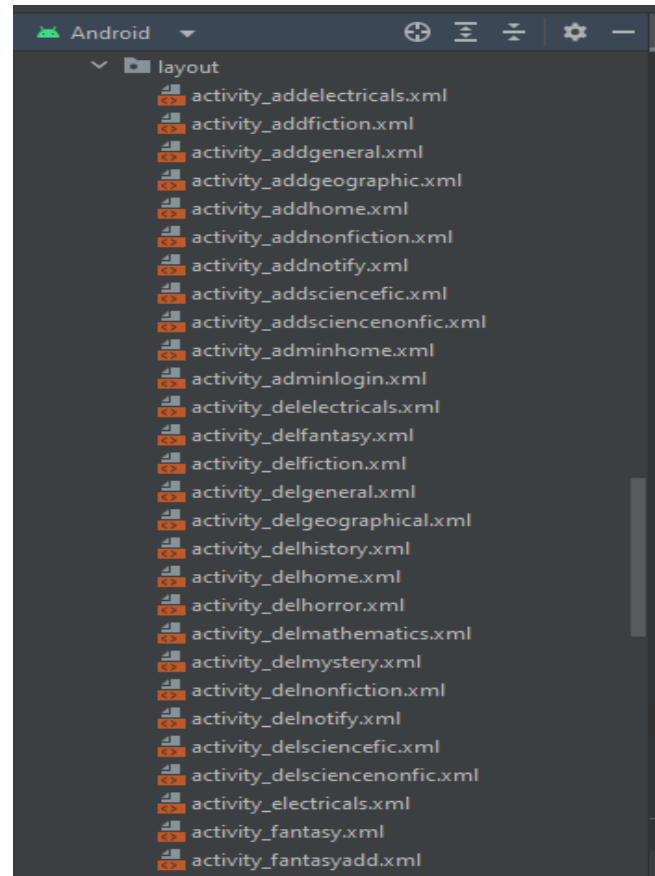
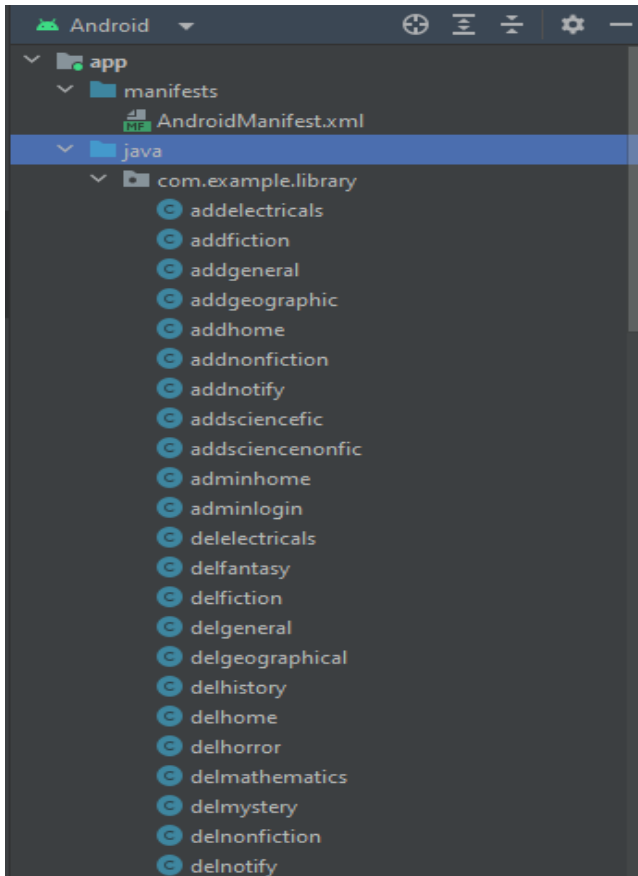
</application>

</manifest>

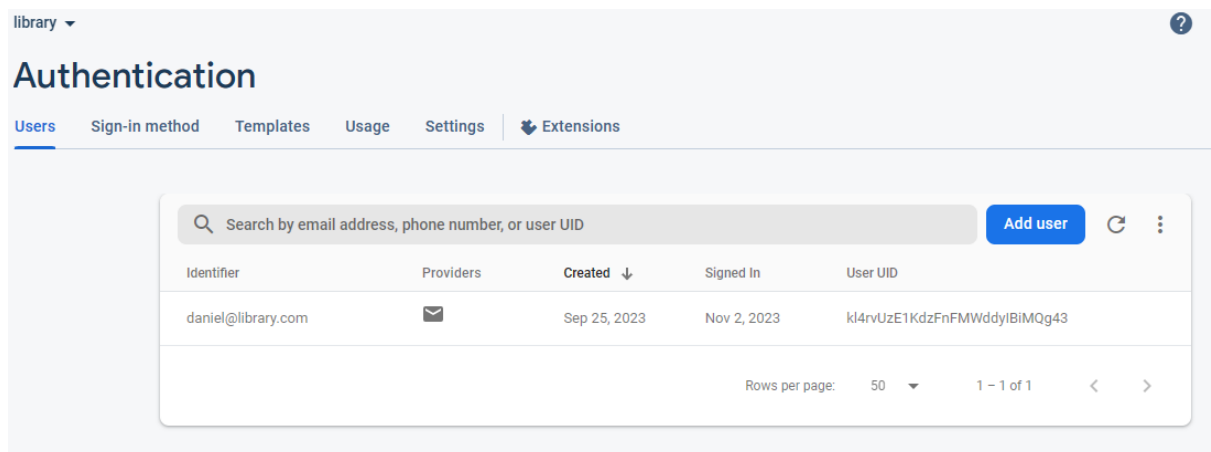
```

9.2 SCREENSHOTS:

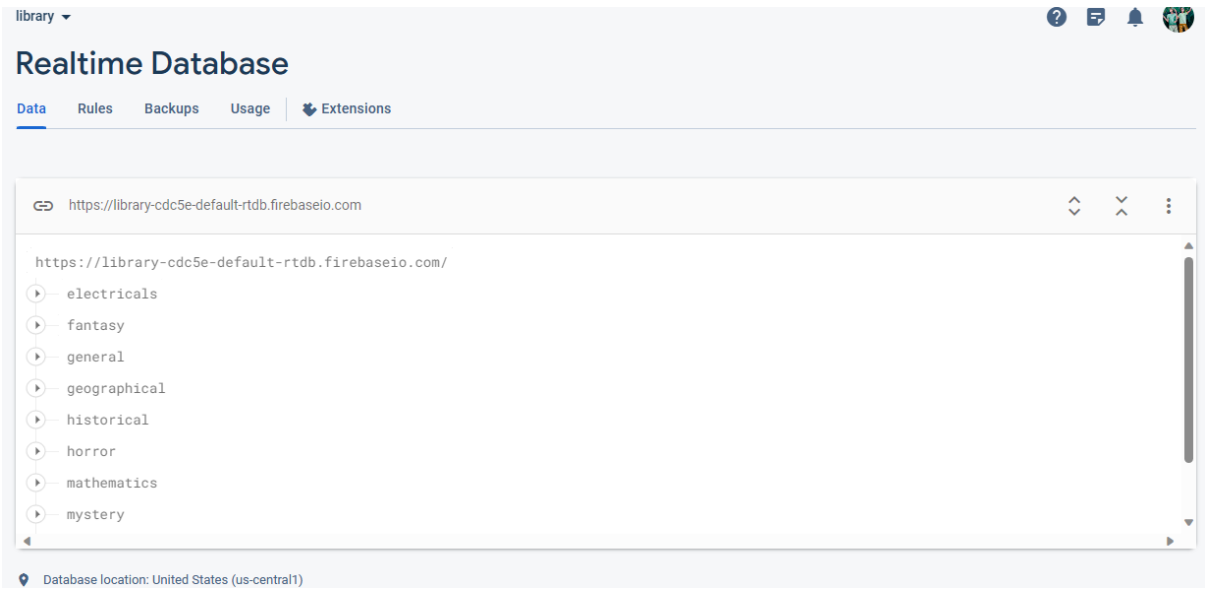
1. PROJECT WINDOW.



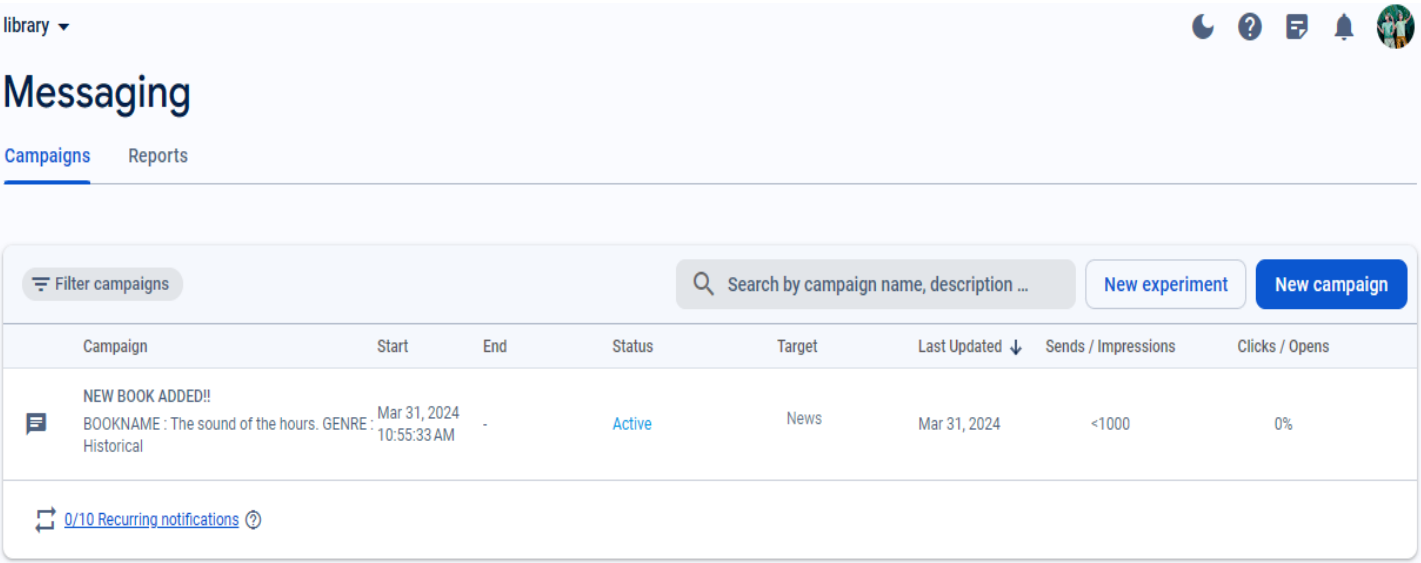
2. AUTHENTICATION.



3. REAL TIME DATABASE.



4. FIREBASE CLOUD MESSAGING (FCM).



10.REFERENCES

- [1] Android (operating system) - Wikipedia
- [2] apk (file format) - Wikipedia
- [3] Firebase Documentation (google.com)
- [4] Android Studio Giraffe | 2022.3.1 | Android Developers
- [5] http://en.wikipedia.org/wiki/Android_application_package.
- [6] Prasanna Pillai, Sonal Singn, Shreya Thakare “Android Application for Library Automation”.
- [7] R.Dinesh, S.R.Arun Pravin, M.Aravindhnan, D.Rajeshwari. “Implementing BRICKS, a Digital Library Management System “.
- [8] Bhupundra, Shraddha Panwar, Vijay Vaishnav. “ Online Library Management System ”.
- [9] R.Dinesh, S.R.Arun Pravin, M.Aravindhnan, D.Rajeshwari. “Library Access System Smartphone Application Using Android ”
- [10] L.J.Haravu, “ Emerging Initiatives in Library Management Systems ”
- [11] http://en.wikipedia.org/wiki/Mobile_technology.
- [12] Nowlan, G. (2013), “Going mobile: creating a mobile presence for your library”, New Library World, Vol. 114 Nos. 3/4, pp. 142-150.
- [13] Firebase Realtime Database (google.com)

