International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

IU CA Department Cross Platform App

Mr. Shoib Ahmad

Student
Computer Applications
Jamia University

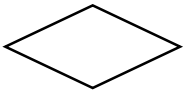
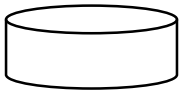
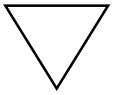
ABSTRACT

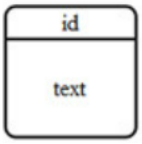
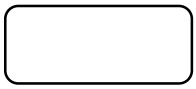
This project report documents the development of a cross-platform mobile application for the **Jamia Hamdard CA Department (IU CA) using Flutter and Dart.** The application aims to streamline student management, improve communication, and enhance productivity by consolidating essential information and features from scattered university websites, including **the Student Management System (IUSMS), the main University website (IUL), and the Student Learning Management System (ILI).**

The application incorporates user registration, secure login, password reset, personalized profile sections, and direct links to crucial university resources. It also features an admin panel that allows administrators to monitor logged-in users and broadcast announcements and notifications.

The development process utilizes an agile methodology with iterative sprints for optimal flexibility and adaptability. The paper concludes by discussing the project's significance in enhancing the educational experience at Jamia Hamdard University and highlights potential future enhancements, including personalized notifications for students. This application demonstrates the power of Flutter and Dart in crafting user-friendly and effective mobile solutions for educational institutions.

LIST OF SYMBOLS

Symbol	Description
	Entity
	Decision
	Manual Input
	Database
	Merge

	Process
	Data Store
	Process

LIST OF FIGURES

1. INTRODUCTION

1.1 roblem Definition

IUSMS (Student Management), IUL (Main University website), and ILI (Student LMS) are the major websites from where the content is taken. This IU CA Department application aims to combine the beneficial elements of the previously mentioned websites. After conducting extensive research with all of the students, we discovered that the ILI website is where one should go to view their LMS and the main IUL website is where one should go to see the results. Sometimes it becomes cumbersome to login or visit to these many websites. Therefore, in order to address this issue, we developed the IU CA Department cross-platform application, a department application.

1.2 User Panel

Access Points: The user panel likely serves as a central navigation menu or dashboard from which users can access various features.

Features:

- **Password Reset:** This enables users to recover their passwords if they've forgotten them. The process typically involves providing the email address used for registration, and the system sends a password reset link to that email.
- **Login:** The primary entry point for users to access their accounts after registration.
- **Registration:** A form for new users to create accounts. The system captures essential user information (name, email, password, etc.) and stores it securely in the backend database (presumably Firebase).
- **Home:** The main page of the application. Users can browse their profile information, access various URL redirects (links to other websites), and potentially view relevant information or announcements.

1.2.1 Home

- **Central Hub:** This serves as the main landing page or dashboard for users.
- **Content:** The home page likely displays user profile information, links to other websites or resources, and possibly relevant announcements or updates.

- **URL Redirects:** This is a key feature. The home page likely features buttons or links that redirect users to different websites, potentially for educational content, relevant services, or external platforms.

1.2.2 Registration form

- **New User Onboarding:** This form allows new users to create accounts within the application.
- **Data Collection:** The form will collect essential user information (name, email, password, etc.).
- **Database Integration:** The system stores this registration data securely in the Firebase backend database.

1.2.3 Password Reset

- **Password Recovery:** This feature allows users to retrieve their forgotten passwords.
- **Email Verification:** The system sends a reset link to the user's registered email address.

1.2.4 Logout Alert Dialogue

- **User Session Termination:** This dialogue provides a user-friendly way to log out of the application.
- **Confirmation:** The dialogue may include a confirmation message to ensure the user is intentionally logging out.
- **User Information Display:** The dialogue might display the user's name or other profile information to verify their identity before logging out.

1.2.5 Profile section

- **User Data Display:** This section displays the user's complete profile information after they've successfully logged in.
- **Verification:** This helps ensure the user is accessing the correct account.

1.2.6 URL redirect section

- **Organized Access:** This section provides a collection of links that redirect users to other websites or resources.
- **Student-Centric Content:** The links would likely focus on educational content, relevant tools, or external platforms often used by students.

1.2.7 Bottom Notification Sheet

- **Notifications –** All the notifications which are received from the admin will be listed in bottom notification sheet. Two options are provide alongwith the bottom navigation which are:
- **Mark as read –** Here all the notifications are listed according to the timestamp they are received.
- **Unread Only –** Here all the notifications which are received but not read by the user will be listed. If marked read they will be removed from this section.

1.3 Admin Panel

- **Administrative Controls:** This section is only accessible to administrators of the application, requiring special credentials (likely a username and password).

Features:

- **View Number of Logged-In Users:** The admin can see a real-time count of how many users are currently active in the application. This information is likely fetched from the Firebase database.

- **View list of Users:** The admin will also be able to view the total number of users along with their name, enrolment no. etc.
- **Send Notifications:** The admin can send personalized notifications to students. This could be used for announcements, updates, or to communicate important information.

1.3.1 View number of logged in user

- **Real-Time Monitoring:** The admin can see how many users are actively using the application at any given time.
- **Data Source:** The data for this information is fetched from the Firebase database, which likely keeps track of active user sessions.

1.3.2 Send notifications

- **Communication Channel:** This enables the admin to reach out to students directly.
- **Content:** Notifications can include a title and a message body.

1.3.3 Bottom Notification Sheet

- **Notifications** – All the notifications which are sent by the admin will be listed in bottom notification sheet.

2. HARDWARE & SOFTWARE REQUIREMENTS

2.1 Hardware Configuration

- **Operating System:** Windows, Linux, MacOS.
- **Hard Disk (HDD) / Solid State Drive (SSD):** Minimum space of 10 GB.
- **Random Access Memory (RAM):** 8 GB (16 GB recommended for smooth functioning).
- **Processor:** Intel i3 11th Gen (i7 is recommended for smooth functioning), M1 for Macbook.

2.2 Software Configuration

- **IDE (Integrated Development Environment):** Visual Studio code, IntelliJ Idea Community Edition, Android Studio.
- **Dart SDK**
- **Flutter Framework**
- **Android toolchain:** Develop for Android Application
- **Chrome:** To build a web application
- **Physical Device:** To directly check the application in the phone.
- **Android SDK tools**
- **Google Firebase**

2.3 Device Configuration

- **Android phone:** Any latest Android phone
- **Android version:** 12+ Android version
- **RAM:** 8 GB
- **ROM:** 100 MB of free space

2.4 Software Features

IDE (Integrated Development Environment) - An **IDE (Integrated Development Environment)** is software that combines commonly used developer tools into a compact GUI (graphical user interface) application. It is a combination of tools like a **code editor**, **code compiler**, and **code debugger** with an integrated terminal. Integrating features like software **editing, building, testing**, and packaging in a simple-to-use tool, IDEs help boost developer productivity. IDEs are commonly used by programmers and software developers to make their programming journey smoother. One certainly does not need an IDE to code or develop applications. Even a simple text editor like notepad can be used to write code. However, IDEs offer some stunning features that go beyond ordinary editing. By providing frequently used **developer tools** all in one simple interface, one can directly get on to building their applications without going through the hardship of manually configuring and integrating the development environment.

Dart - Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks.
Languages are defined by their technical envelope—the choices made during development that shape the capabilities and strengths of a language. Dart is designed for a technical envelope that is particularly suited to client development, prioritizing both development (sub-second stateful hot reload) and high-quality production experiences across a wide variety of compilation targets (web, mobile, and desktop).
Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.
The Dart language is type safe; it uses static type checking to ensure that a variable's value always matches the variable's static type. Sometimes, this is referred to as sound typing. Although types are mandatory, type annotations are optional because of type inference. The Dart typing system is also flexible, allowing the use of a dynamic type combined with runtime checks, which can be useful during experimentation or for code that needs to be especially dynamic.
Dart has built-in sound null safety. This means values can't be null unless you say they can be. With sound null safety, Dart can protect you from null exceptions at runtime through static code analysis. Unlike many other null-safe languages, when Dart determines that a variable is non-nullable, that variable can never be null. If you inspect your running code in the debugger, you see that non-nullability is retained at runtime; hence sound null safety.
The following code sample showcases several Dart language features, including libraries, async calls, nullable and non-nullable types, arrow syntax, generators, streams, and getters.

Features:

- Object-oriented: Supports inheritance, interfaces, and polymorphism
- Type-safe: A statically typed language that checks variable types during compilation
- Cross-platform: Runs on all major operating systems
- Garbage collection: Automatically manages memory
- Lists: Can infer data type to a variable, such as var myList = [1, 2, 3]
- Async/await: Allows writing asynchronous code that looks like synchronous code
- Optional parameters: Can be positional or named, and must be declared after required parameters
- Indentation: Two-character indentation makes code more compact

Flutter - Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems such as iOS and Android, while also allowing applications to interface directly with underlying platform services. The goal is to enable developers to deliver high-performance apps that feel natural on different platforms, embracing differences where they exist while sharing as much code as possible.
During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.

Features:

- Cross-platform: Code can be written once and deployed on multiple platforms
- Hot Reload: Developers can see changes to the UI immediately
- Expressive UI: Developers can create responsive user interfaces with customizable widgets
- Fast performance: Flutter uses Dart, a compiled language, and the Skia graphics library, which gives apps comparable performance to native applications
- Firebase: Flutter's backend is fast and reliable, with user authentication and usage tracking
- Integration with native device features: Flutter provides pre-built widgets for common device features like camera, GPS, and sensors
- Rich widgets: Widgets can be customized according to UX/UI guidelines, and help users navigate the app
- Transparent design: Apps are easier to understand
- Rapid app development: Apps can be accessed from any device and platform, and users can navigate effortlessly
- Data-driven user experience: Apps are available for iOS, Android, and Windows devices

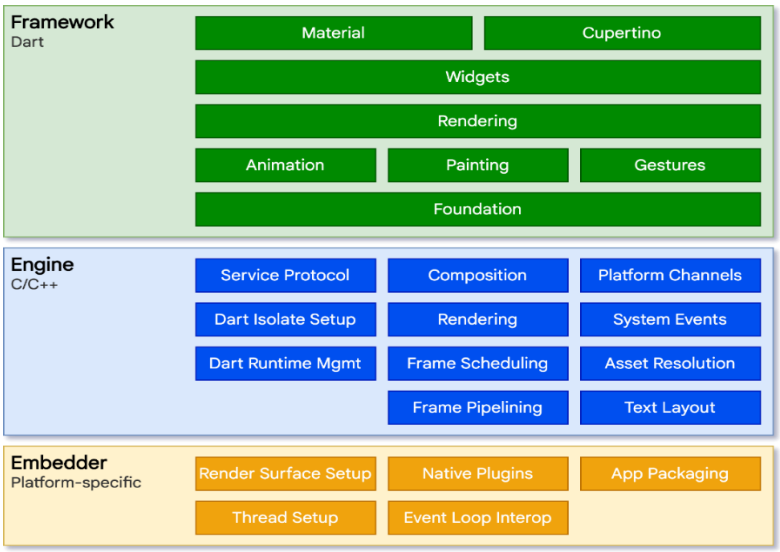


Fig. 1 Architecture of Dart

Android SDK Tools - Android SDK is a collection of libraries and Software Development tools that are essential for Developing Android Applications. Whenever Google releases a new version or update of Android Software, a corresponding SDK is also released with it. In the updated or new version of SDK, some more features are included which are not present in the previous version. Android SDK consists of some tools which are essential for the development of Android Applications. These tools provide a smooth development process flow from developing and debugging. Android SDK is compatible with all operating systems such as Windows, Linux, macOS, etc.

Google Firebase - Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data. Firebase initially was an online chat service provider to various websites through API and ran with the name Envolve. It got popular as developers used it to exchange application data like a game state in real time across their users more than the chats. This resulted in the separation of the Envolve architecture and it's chat system. The Envolve architecture was further evolved by it's founders James Tamplin and Andrew Lee,to what modern day Firebase is in the year 2012.

Features:

- Realtime Database: The Firebase Realtime Database is a cloud-based NoSQL database that manages your data at the blazing speed of milliseconds. In simplest term, it can be considered as a big JSON file.
- Cloud Firestore: The cloud Firestore is a NoSQL document database that provides services like store, sync, and query through the application on a global scale. It stores data in the form of objects also known as Documents. It has a key-value pair and can store all kinds of data like, strings, binary data, and even JSON trees.
- Authentication: Firebase Authentication service provides easy to use UI libraries and SDKs to authenticate users to your app. It reduces the manpower and effort required to develop and maintain the user authentication service. It even handles tasks like merging accounts, which if done manually can be hectic.
- Remote Config: The remote configuration service helps in publishing updates to the user immediately. The changes can range from changing components of the UI to changing the behavior of the applications. These are often used while publishing seasonal offers and contents to the application that has a limited life.
- Hosting: Firebase provides hosting of applications with speed and security. It can be used to host Stati or Dynamic websites and microservices. It has the capability of hosting an application with a single command.
- Firebase Cloud Messaging(FCM): The FCM service provides a connection between the server and the application end users, which can be used to receive and send messages and notifications. These connections are reliable and battery-efficient.

Google Firebase - In the era of rapid prototyping, we can get bright ideas, but sometimes they are not applicable if they take too much work. Often, the back-end is the limiting factor - many considerations never apply to server-side coding due to lack of knowledge or time.

Firebase is a Backend-as-a-Service(BaaS) that started as a YC11 startup. It grew up into a next-generation app-development platform on Google Cloud Platform. Firebase (a NoSQLjSON database) is a real-time database that allows storing a list of objects in the form of a tree. We can synchronize data between different devices.

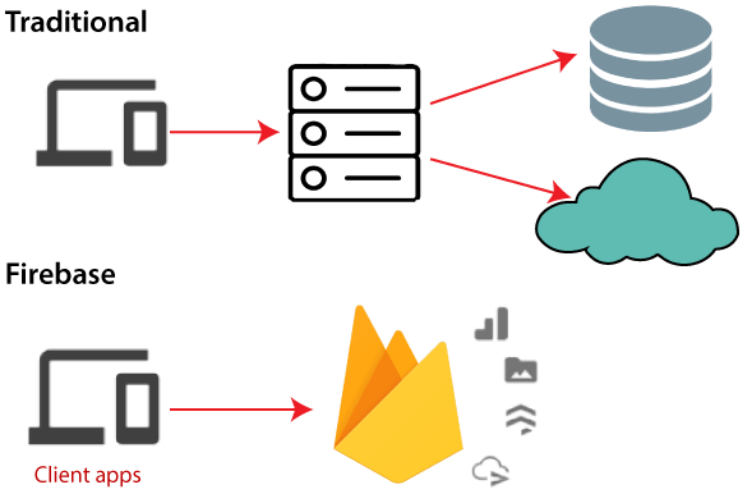


Fig 2. Firebase Architecture

Google Firebase is Google-backed application development software which allows developers to develop Android, IOS, and Web apps. For reporting and fixing app crashes, tracking analytics, creating marketing and product experiments, firebase provides several tools.

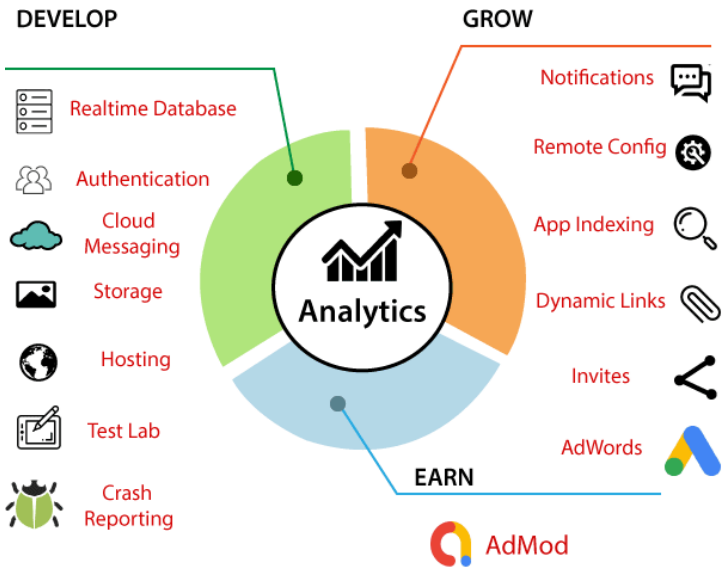


Fig 3. Firebase Analytics

Firebase has three main services, i.e., a real-time database, user authentication, and hosting. We can use these services with the help of the Firebase iOS SDK to create apps without writing any server code.

History of Firebase

Firebase evolved from Envolve. Envolve is a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API which allowed the integration of online chat functionality into their websites. After releasing the chat service, it found that the envelope was being used to pass

application data, which were not chat messages. Developers used Envolve to sync application to separate the real-time architecture and the chat system which powered it. In September 2011, Tamplin and Lee founded firebase as a separate company. It was lastly launched to the public in April 2012.
Firebase Real-time Database was the first product of firebase. It is an API which syncs application data across Android, iOS, and Web devices. It gets stored on Firebase's cloud. Then the firebase real-time database helps the developers to build real-time, collaborative applications.

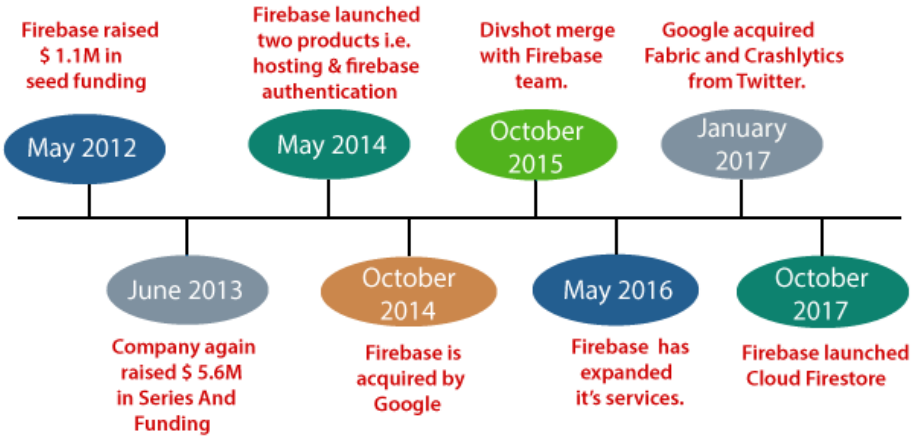


Fig 4. History of Firebase

In May 2012, after launching the beta, Firebase raised $1.1M in seed funding from Greylock Partners, venture capitalists Flybridge Capital Partners, New Enterprise Associates, and Founder Collective.
In June 2013, the company again raised $5.6M in Series A funding from Flybridge Capital Partners and venture capitalists Union Square Ventures.
Firebase launched two products in 2014, i.e., Firebase Hosting and Firebase Authentication. It positioned the company as a mobile backend as a service.
Firebase was acquired by Google in October 2014.
Google promoted Div shot to merge it with the Firebase team in October 2015.
In May 2016, Firebase expanded its services to become a unified platform for mobile developers. Now it has integrated with various other Google services, including AdMob, Google Cloud Platform, and Google Ads, to offer broader products and scale it for developers.
Google acquired Fabric and Crashlytics from Twitter in January 2017 to add Fabric and Crashlytics services to Firebase.
Firebase launched Cloud Firestore in October 2017. It is a real-time document database as the successor product for the original Firebase Realtime Database.

Why use Firebase?
Firebase manages real-time data in the database. So, it easily and quickly exchanges the data to and from the database. Hence, for developing mobile apps such as live streaming, chat messaging, etc., we can use Firebase.
Firebase allows syncing real-time data across all devices - iOS, Android, and Web - without refreshing the screen.
Firebase provides integration to Google Advertising, AdMob, Data Studio, BigQuery DoubleClick, Play Store, and Slack to develop our apps with efficient and accurate management and maintenance.

Everything from databases, analytics to crash reports are included in Firebase. So, the app development team can stay focused on improving the user experience.
Firebase applications can be deployed over a secured connection to the firebase server.

Pros and Cons of Firebase

Firebase has a lot of pros or advantages. Apart from the advantages, it has disadvantages too. Let's take a look at these advantages and disadvantages:

Pros:

- Firebase is a real-time database.
- It has massive storage size potential.
- Firebase is serverless.
- It is highly secure.
- It is the most advanced hosted BaaS solution.
- It has minimal setup.
- It provides three-way data binding via angular fire.
- It provides simple serialization of app state.
- We can easily access data, files, auth, and more.
- There is no server infrastructure required to power apps with data.
- It has JSON storage, which means no barrier between data and objects.

Cons:

- Firebase is not widely used, or battle-tested for enterprises.
- It has very limited querying and indexing.

Firebase Authentication - In the present era, user authentication is one of the most important requirements for Android apps. It is essential to authenticate users, and it is much harder if we have to write all this code on our own. This is done very easily with the help of Firebase.
Being able to authenticate our users securely, it offers a customized experience to them based on their interests and preferences.
We can ensure that they have no problems accessing their private data while using our app from multiple devices.
Firebase Authentication provides all the server-side stuff for authenticating the user. Firebase Authentication becomes easy with SDK. It makes API easy to use.
Firebase Authentication also provides some user interface libraries which enable screens for us when we are logging it.
Firebase authentication supports authentication using a password, phone numbers, popular identity provider like Google, Facebook, and Twitter, etc.
We can sign in users to our app by using the FirebaseUI.
It handles the UI flows for signing in user with an email address and password, phone numbers, and popular providers, including Google sign-In and Facebook Login.
It can also handle cases like account recovery.
It is not required to design a UI since it is already provided for us. It means we don't have to write the activities.

We can also sign-in users using the Firebase Authentication SDK to integrate one or several sign-in methods into our app manually.

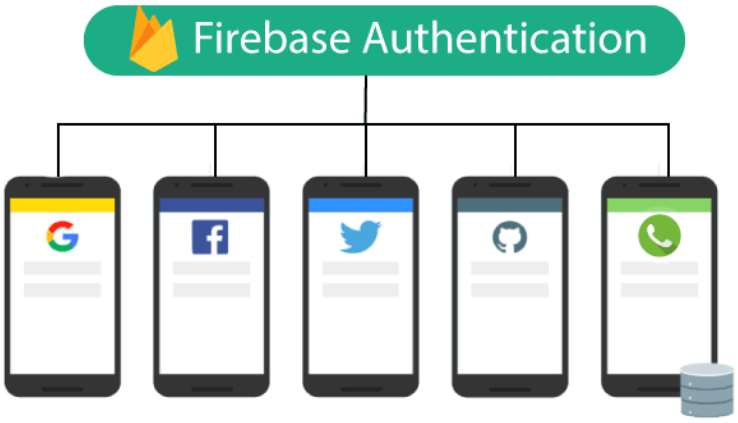


Fig 5. Firebase Authentication

Firebase UI Authentication Method - Firebase UI Authentication is a way to add a complete sign-in system to our app, where Firebase provides user interface to them. Firebase UI provides a drop-in auth solution which is used to implement authentication on mobile devices and websites. Firebase UI can be easily customized to fit with the rest of our app's visual style. It is open-source, so we are not constrained in modifying the user experience to meet our apps need.

There are the following steps to use Firebase UI Authentication:

- Set up sign-in methods:
  - Enable authentication method in the firebase console.
    - For email address and password, phone number sign-in, and any identity providers.
  - We have to complete the configuration if anyone requires for identity providers.
    - Setting our OAuth redirect URL.
- Customize the sign-in UI.
  - For customizing the sign-in and UI, we have to set some Firebase UI options or fork the code on GitHub.
- To perform the sign-in flow, use Firebase UI:
  - Import the Firebase UI library.
  - Specify the sign-in method which we want to support.
  - Initiate the Firebase UI sign-in flow.

Firebase SDK Authentication Method

- This is another type of authentication method. The Firebase SDK Authentication provides methods for creating and managing users who use their email addresses and password to sign in. SDK also handles sending password reset emails.
- We can also provide phone number authentication using SDK
  - The authentication of users by sending SMS messages to their phones.
- We can authenticate users by integrating with identity providers.

- SDK provides methods which allow users to sign-in with their Google, Facebook, Twitter, and GitHub accounts.

- We can connect our app's existing sign-in system to the Firebase Authentication SDK and gain access to Firebase Real-time database and other Firebase services.
- We can create a temporary anonymous account to use Firebase features, which requires authentication
- without requiring users to sign-in first.

There are the following steps to use Firebase SDK Authentication:

- Set up sign-in methods:
  - We have to enable the authentication method in the Firebase console for an email address and password or phone number sign-in and any identity providers.
  - We have to complete the configuration if anyone is required for identity providers.
    - Setting our OAuth redirect URL.
- Implementing UI flows for our sign-in methods:
  - For email signing, add screens which prompt the user to type their email addresses
  - For phone number sign-in, add screens which prompt users to type their phone number, and after that, for the code from the SMS message they receive.
  - For identity sign-in, implement the flow required by each provider.
- Passing the user's credentials to the Firebase Authentication SDK:
  - Pass the user's email address and password.
  - Pass the OAuth token, which was acquired from the identity provider.

How Authentication Works?

- We first get authentication credentials from the user to sign a user into our app.
  - Credentials can be the user's email address and password.
  - The credential can be an OAuth token from an identity provider.
- We then pass these credentials to the Firebase Authentication SDK. Backend services will then verify those credentials and return a response to the client.
- After a successful sign in:
  - We can access the user's access to data stored in other Firebase products.
  - We can access the user's basic profile information.
  - We can use the provided authentication token to verify the identity of users in our own backend services.
- An authenticated user can read and write data to the Firebase Real-time Database and Cloud Storage.
  - We can control the access of those authenticated users by modifying the Firebase Database Rules and Storage Security Rules.

User

- A Firebase User object represents the account of a user who has signed up to the app in Firebase project. Apps have many registered users, and every app in a Firebase project shares a user data base.
- AFirebase User instance is independent of a Firebase Auth instance. It means we can have several references to different users within the same context and still call any of their method

- A Firebase User has a fixed set of basic properties such as Unique ID, Primary email address, Name, and a photo URL.
- Firstly, a user signs up to the app. The user's profile data is populated with the primary email address if using email/password auth, account information available by the provider if using identity auth, and anything we want if using custom auth.
- The user becomes the current user of the Auth instance when a user signs up or signs in.
- The Auth instance stops to keep a reference to the User object. And no longer persists it states when a user signs out:
  - No current user
  - The user instance continues to be completely functional
  - If we keep a reference to it, we can still access and update the user's data.
- Using listeners is the recommended way to track the current state of the Auth instance.
  - An Auth listener gets notified any time when something relevant happens to the Auth object.

User Lifecycle

- An Auth listener gets notified in the following situation
- The Auth object finishes initializing, and a user was already signed in from a previous session or has been redirected from an identity provider's sign-in flow
- A user signs in.
- A user signs out.
- The current user's access token is refreshed:
  - The access token expires.
  - The user changes their password.
  - The user re-authenticates

Firebase Cloud Messaging - Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that reliably sends the message at no cost. It is formally known as Google Cloud Messaging, which is used for Android, iOS, and web applications.

The service is provided by Google's subsidiary Firebase and on 21 October 2014, Firebase announced that it had been acquired by Google for an undisclosed amount. The official Google Cloud Messaging website demonstrates the Firebase Cloud Messaging (FCM) as the new version of GCM.

If we are using Google Cloud Messaging (GCM) server and client APIs, then there is some bad news, which is that this service has already been removed, and Google plans to discontinue "most" GCM services in April 2019. If we are still using GCM, then we should start migrating our projects to FCM now and complete our migration by April 2019.

FCM is a free, cross-platform messaging solution which allows us to send push notifications to our audience without worrying about the server code. Using FCM with Firebase's Notification Composer (as seen in the following screenshot), the user can create notifications that target very specific sections of the user base, generally without writing any specific code. **Using FCM:**

- We can send data messages and notification messages.
- We can distribute messages for a single device, groups of the device, or for devices subscribed to some topic.

- We can send chats, acknowledgments, and other messages from devices back to the server over FCM's battery-efficient connection and reliable channel.

How is FCM differ from GCM?

- In FCM, there is no need to write our own registration or subscription retry logic.
- There is no need to declare "Receiver" in FCM explicitly.
- There is no need to initialize the registration token because the generation of registration token is handled by the library itself.
- FCM gives server-less notification solution with web console-Firebase notification. This web console will remind us about the Parse Push console.

How does it work?

FCM implementation contains two main components for sending and receiving. The first one is a trusted environment such as Cloud Function for Firebase or an app server on which to build, target, and send messages, essentially the server-side, and another one is an android client app which receives messages. If we implement our own server code in Cloud Function or Java code, then we can send messages via Firebase Admin SDK or the FCM server protocols.

We can also use the Notification composer for testing or for sending marketing or engagement messages with powerful built-in targeting and analytics.

Implementation path

Set up the FCM SDK - Set up Firebase and FCM on our app according to the setup instruction for our platform.

Develop our client app - In our client app, we have to add message handling, topic subscription logic, or other optional features. During development, we can easily send text messages from the Notification composer.

Develop our app server - We need to decide whether we want to use the Firebase Admin SDK or one of the server protocols to create our sending logic, i.e., logic to authenticate, buildsend requests, handle responses, etc., and build out the logic in our trusted environment.

Concerning the development of our own app server - It will give us the basics of the Server environment, but we will not write any code.

Types of Message - Using Firebase Cloud Messaging, we can send three types of messages, i.e., Notification Message, Data Message, and the message with both Notification & Data Payload.

Notification Message - Firebase SDK has handled notification messages itself. Typically, the notification message includes the title, icon, message, etc. These messages can also be sent from the Firebase console UI. By sending this type of message, we will not have much control over the information. The notification will appear automatically when the app is in the background.

A notification is a message which Android shows outside of our app's UI for providing users with reminders, communication with other people, or other timely information about our app. Users can tap on the notification to open our app or take action directly from the notification.

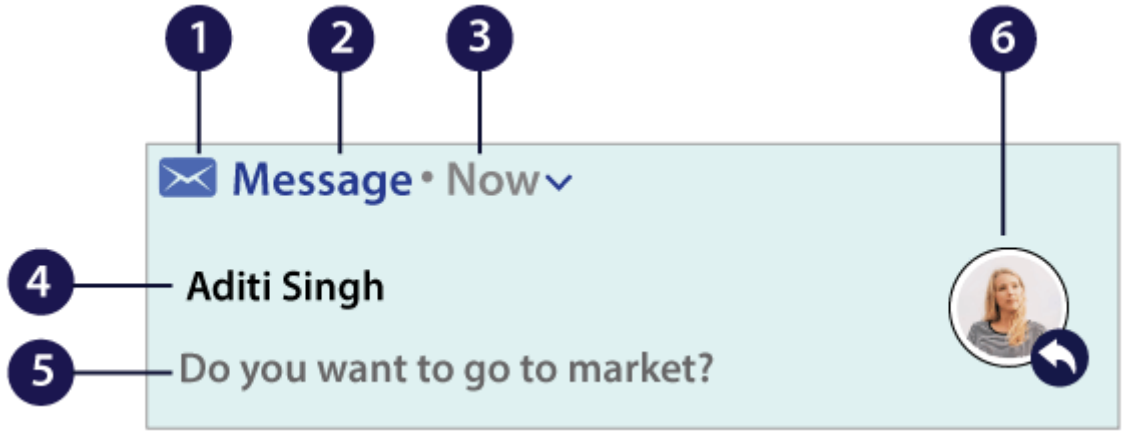


Fig 6. FCM Structure

The notification design is determined by the system template - the content for each part of the template is defined by our app. Some information on the notification only appears in the expanded view.

Data Message - Data messages are handled by the Android app. If we want to send some additional data along with the information, then we can add such messages. But it is not possible to send these messages through the Firebase console. To send notifications using the Firebase API, we must have server-side logic. We must use the data key while sending this message. We can use data messages to send custom data elements to a client application. However, FCM places a 4KB limit on these data messages, so if our payload is greater than 4KB, we must obtain additional data using the Work Manager or Job-Scheduler API.

Messages with both Notification and Data payload - Both Notification and Data payload can also be contained in a message. Sending of these types of messages is handled in two scenarios depending upon the app state, i.e., background and foreground. We can use both the notification and data keys for these messages.

When the app state is in the background, the apps receive the notification payload when the user taps on the notification, and when in the foreground, the app receives a message object with both payloads available.

FCM Console - Now, we will create an Android project and add it with the Firebase either from Google Assistant or manually. After that, we will move to the FCM console (Firebase->Grow->Cloud Messaging).

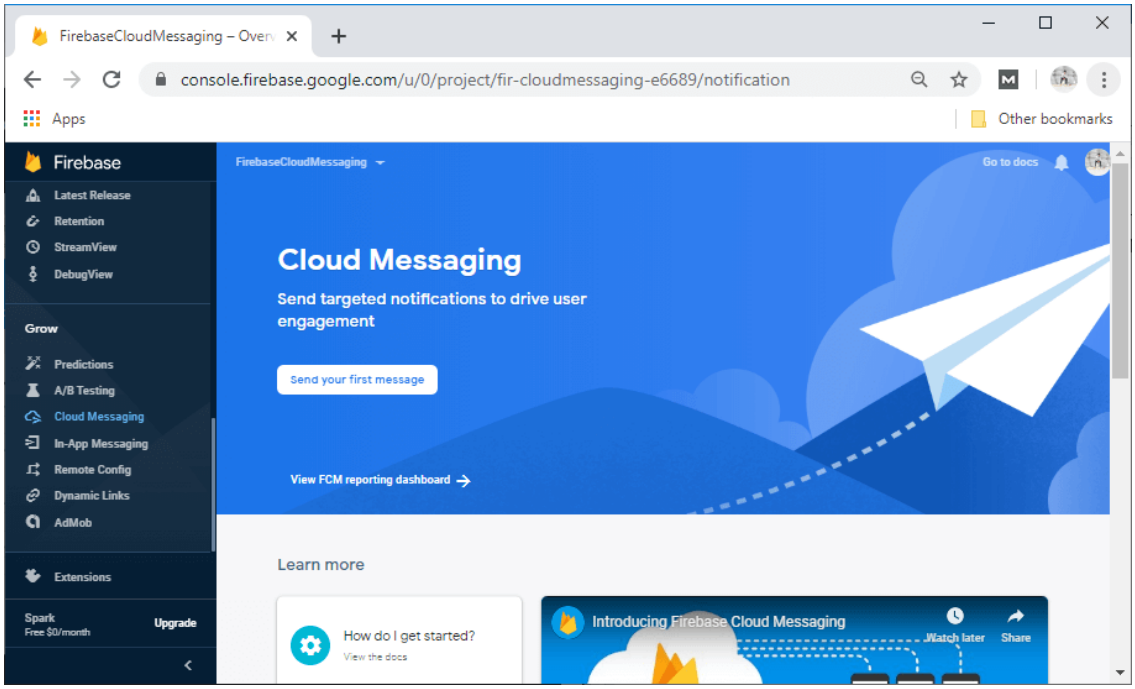


Fig 7. FCM Console

Click on **Send your first message**.

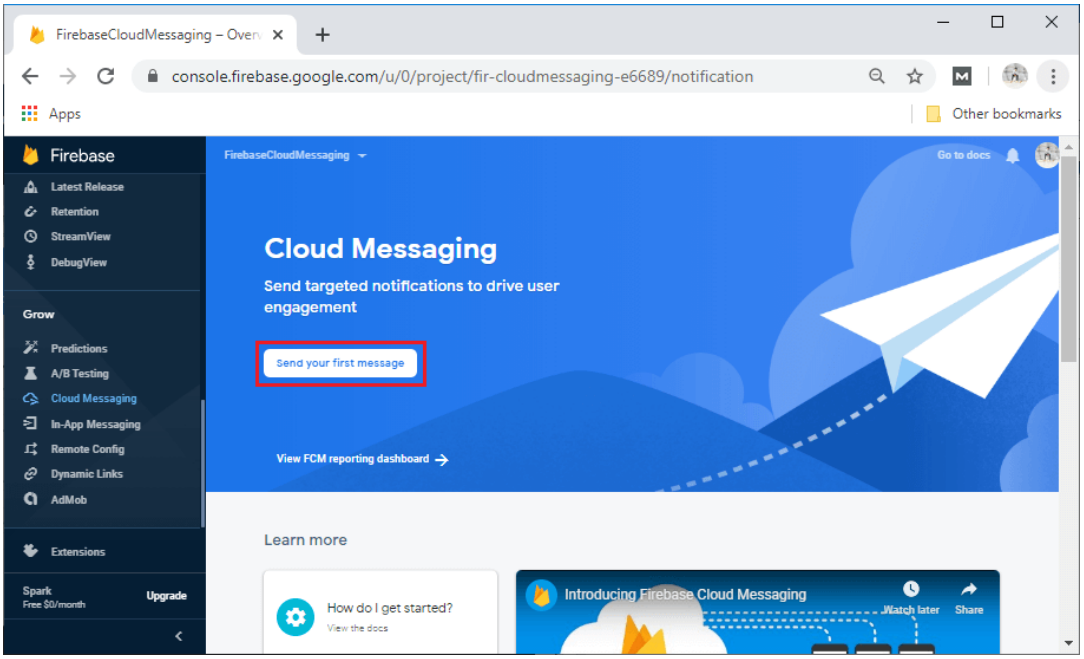


Fig 8. FCM Message

After clicking on **Send your first message**, it will ask to fill a few of the fields such as notification title, notification text, notification image that is optional, and notification name, which is also optional.

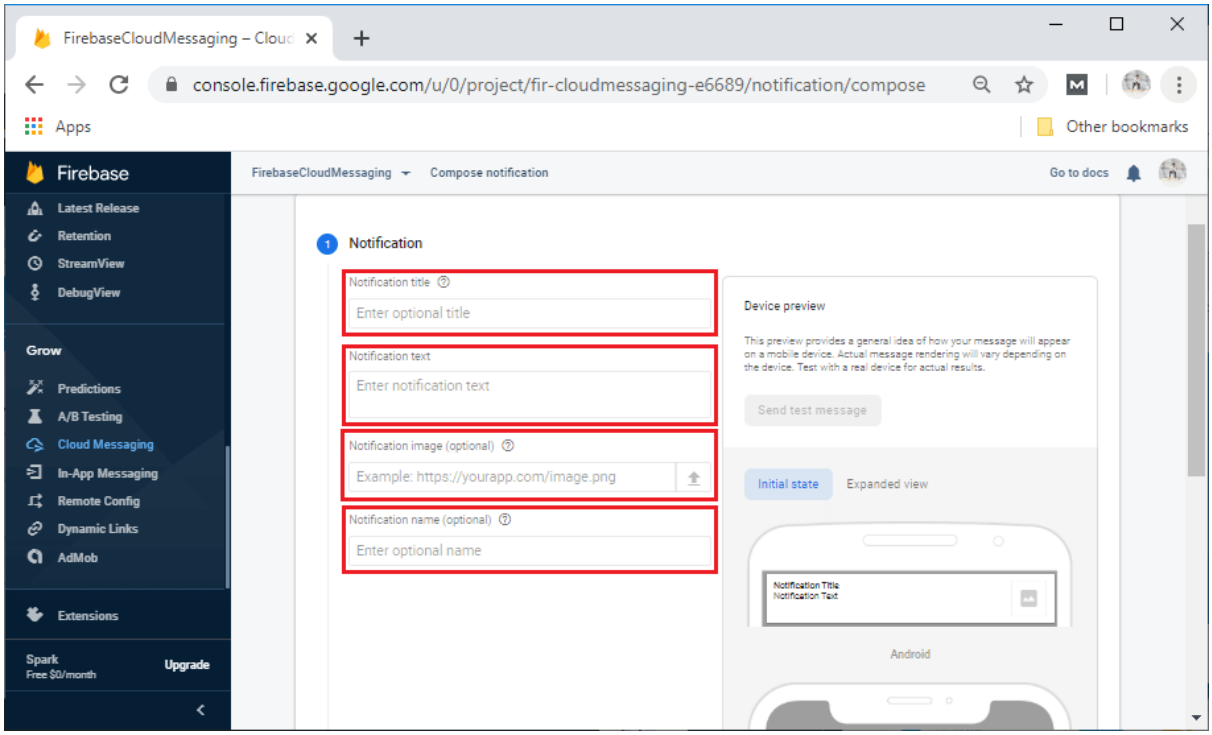


Fig 9. FCM Message Menu

After filling all the fields, click on **Next**.
After clicking on **Next**, it will ask for the **Target** either **User segment** or **Topic**. We will select the app for the target user and click on **Next**.

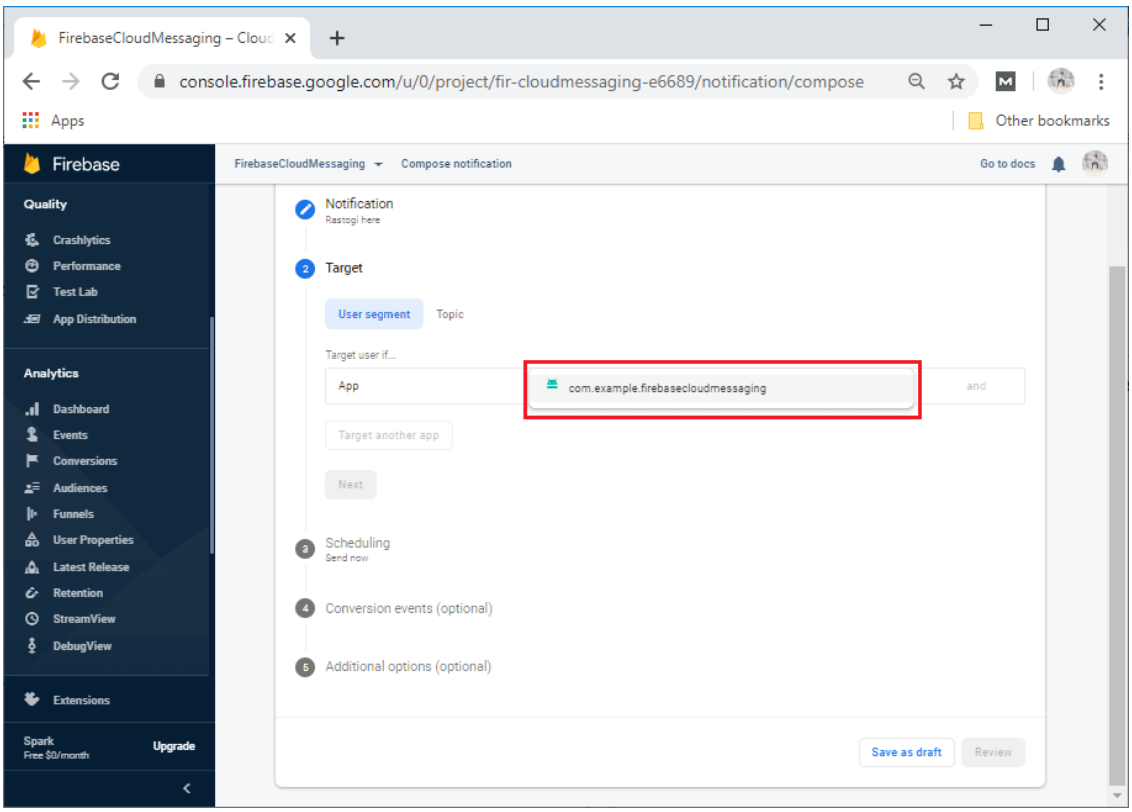


Fig. 10. FCM Target set

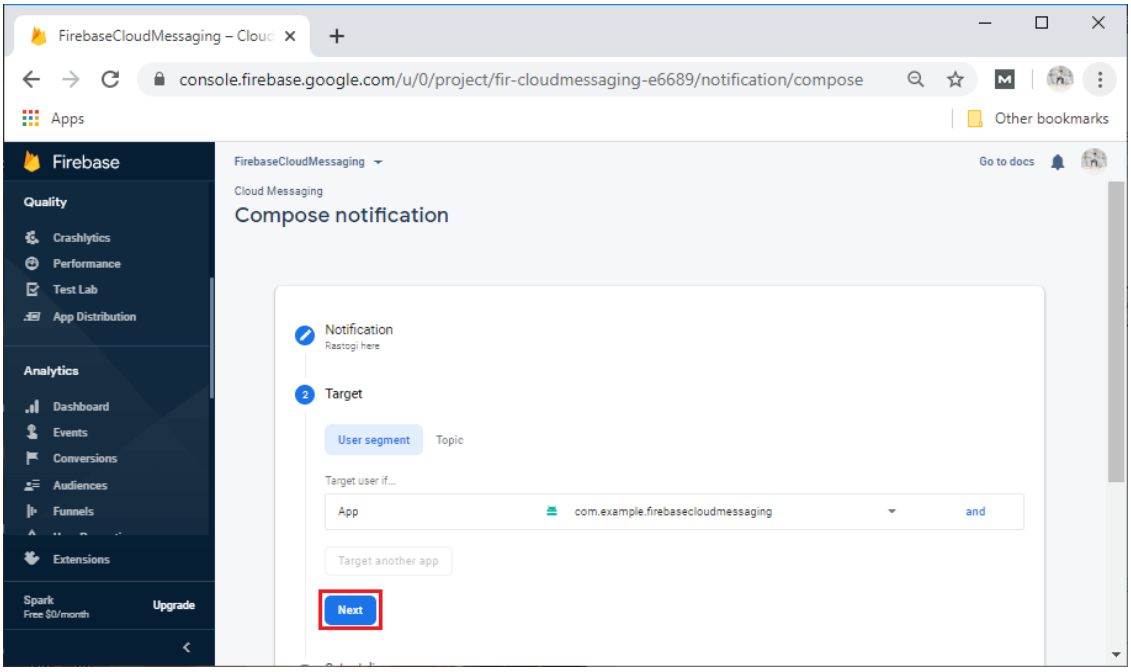


Fig 11. FCM Compose Notification

When we click on the **Next**, it will ask for the **Scheduling** and click on **Next again**.

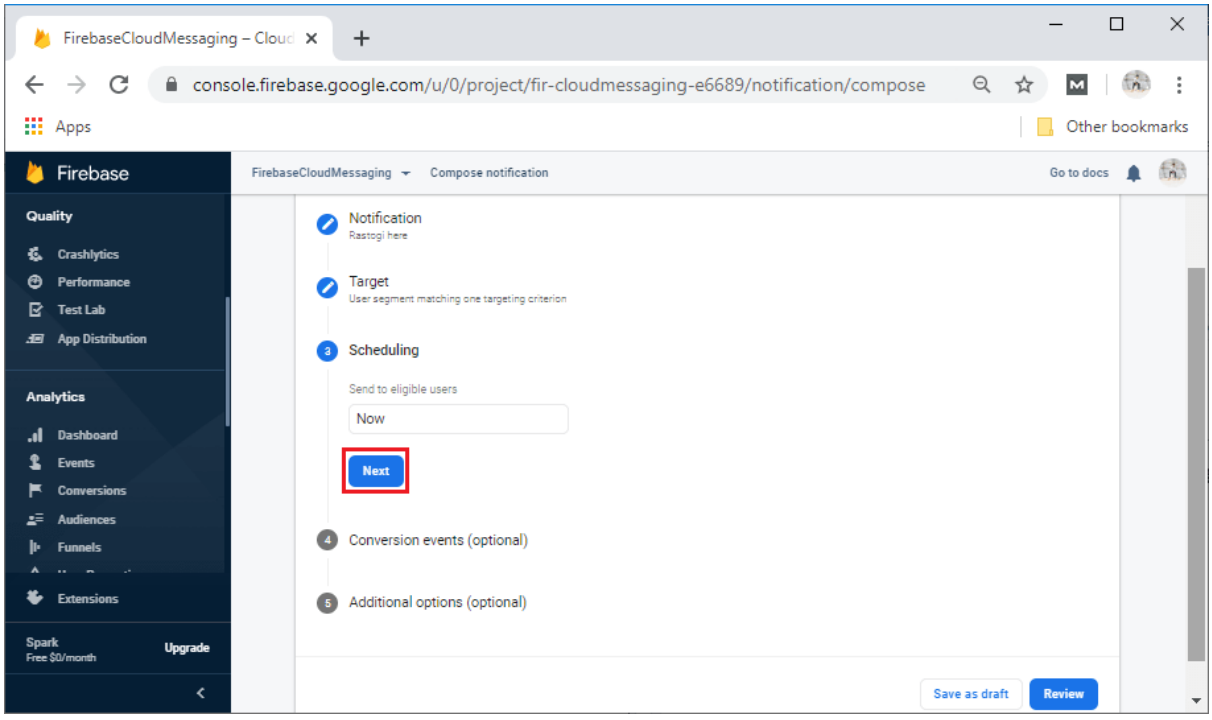


Fig 12. FCM Select Target

After that, it will ask for the **Conversion events**. We can do this, but it is optional. So, we can ignore it and proceed further.

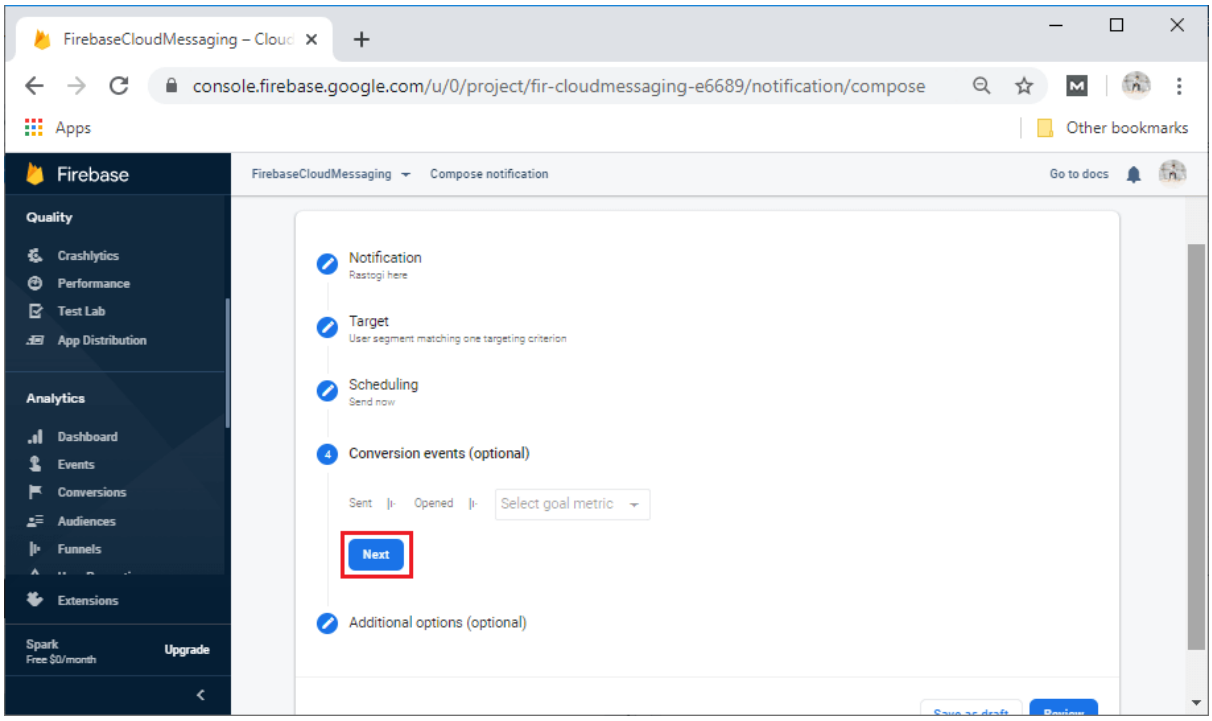


Fig 13. FCM Set Range

After that, it will ask for some additional options, which are also optional. These are notification channels, custom data, sound, and expires, etc. After that, we will click on **the Review**.

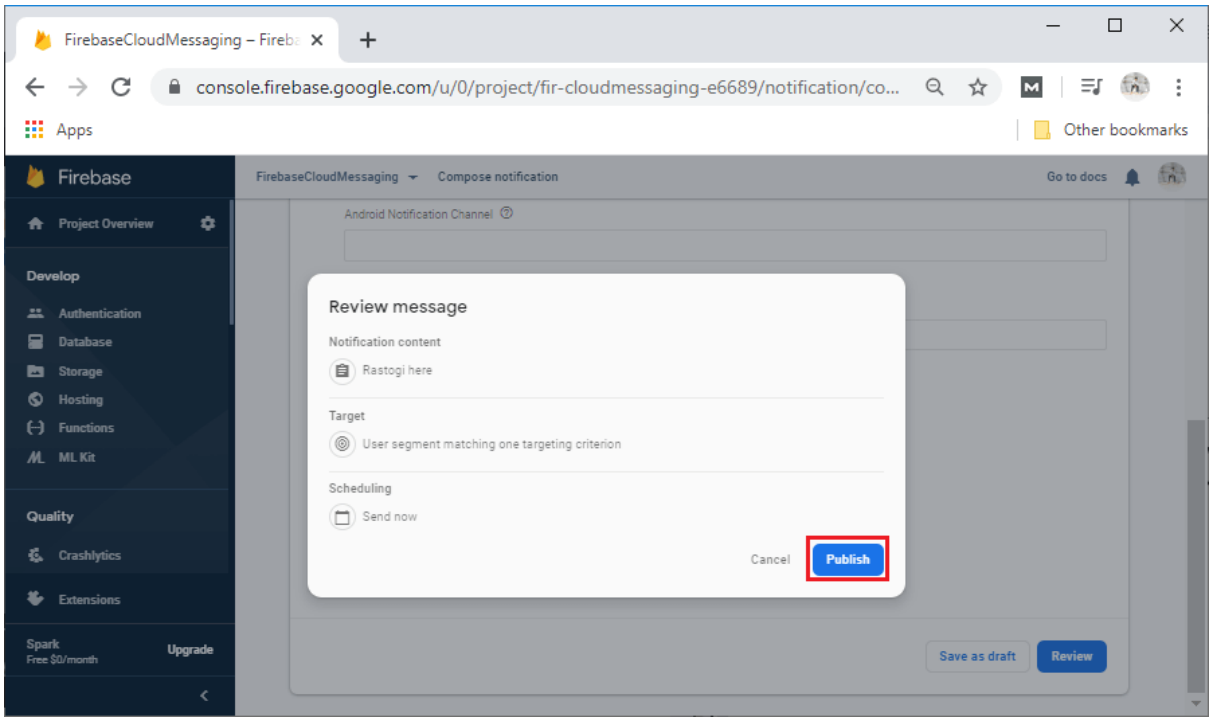


Fig 14. FCM Publish

After publishing, we can see the notification in the console.

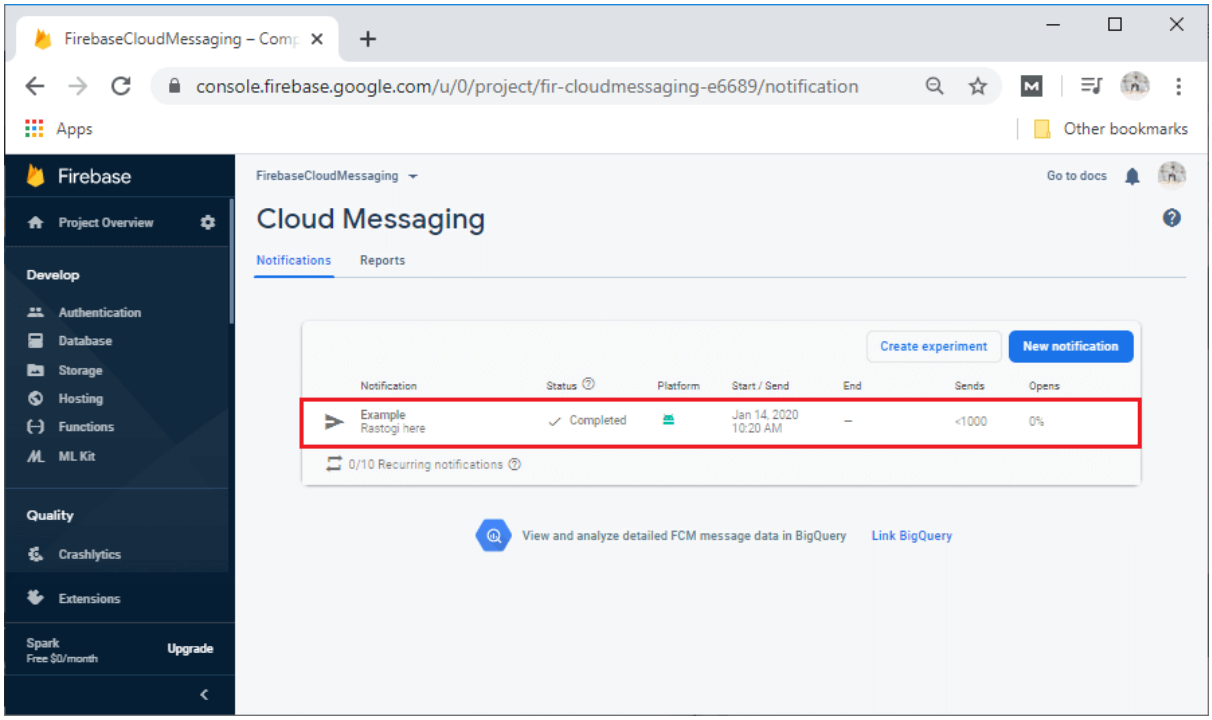


Fig 15. FCM Final Screen

3. FEASIBILITY STUDY

3.1 Requirements Gathering & Analysis

A study was conducted with Integral University students to find out the main issues they run into when searching for anything. The main issue facing the pupils was that they required numerous websites to get different types of material. Following my observation of this issue, I contacted my project guide, who assisted me in obtaining data and links from various Integral University websites, including the Main University website (iul.ac.in), the Student Management System (IUSMS), and the Student LMS (ILI). My project guide gave me the pertinent information for the project after talking with him. All requirements regarding this project are listed below:

3.2 System Analysis and Design:

3.2.1 Functional Requirements:

Registration Page:

Capture student information:

- **Fields:** Full Name, Email, Enrollment Number, Course, Year, Profile Picture (optional), Passwor d.
- **Data Types:** String, Email, Integer, String, Integer, Image (URL), Password (hashed).
- **Validation:** Required fields.
- Email format validation.

- Enrollment number format and uniqueness.
- Password strength (length, complexity).

Storage:

- Store data securely in Firebase Firestore.
- Use appropriate security rules to restrict unauthorized access.
- Hash passwords using a strong hashing algorithm like bcrypt.

Login Page:

Authentication:

- **Users:** Students and Admins.
- **Credentials:** Email and password.
- **Firebase Authentication:** Utilize Firebase Authentication for user management.
- **Authorization:**Rolebased access control: Define roles (Student, Admin) and grant permissions a ccordingly.

Restrict access:

- Admin pages should only be accessible to Admin users.
- Student-specific data should only be accessible to the respective student and admins.

Security:

- Usage of HTTPS to encrypt communication between client and server.

Admin Page:

- **Send notification:** Admin will be able to send notifications to the students which will be notified in the form of banner.
- **See no. of users:** Admin will be able to view the number of users who are currently using the app in a realtime.

Content Management:

- **Announcements:** Create announcements.
- **Other content:** Manage additional content relevant to the system (e.g., Uploading Coordinator or of other useful announcements).

Home Page:

Personalized Information:

- **Dashboard**: Display information of the user who is currently logged in into the application.
- **Student View:** Can see their profile and can be able to fetch different URL buttons listed on a homepage.

- **Admin View:** Admin can be able to view number of students logged in into the app and can send messages to every student who logged in into the application.

Resource Access:

- **Navigation:** Clear navigation to different system sections (e.g., Attendance, Timetable, Resource s).
- **Direct links:** Providing easy access to frequently used resources.

User-friendly Interface:

- **Responsive design:** Ensuring compatibility across various devices.
- **Clean layout:** Organizing information for optimal readability.
- **Intuitive interactions:** Making actions straightforward to understand.

3.2.2 Non-Functional Requirements:

Security:

- **Data Encryption:** Encrypting sensitive data at rest and in transit.
- **Access Control**: Enforcing strict authorization policies.
- **Regular Security Audits:** Identifying and addressing potential vulnerabilities.

Performance:

- **Optimized Database Queries:** Use efficient queries to minimize data retrieval time.
- **Caching:** Cache frequently accessed data to reduce server load.
- **Scalability:** Design the system to handle increasing data volumes and user traffic.

Usability:

- **User-centered design:** Prioritizing user needs and preferences in the interface design.
- **Accessibility:** Ensuring the system is usable by individuals with disabilities.
- **Help and Documentation:** Providing clear instructions, FAQs, and tutorials.

Availability:

- **Redundancy:** Implementation of redundant systems to prevent single points of failure
- **Disaster Recovery:** Develop a plan for data recovery and system restoration.
- **Monitoring:** Continuously monitor system health and performance.

3.3 System Design

3.3.1 System Architecture:

Client-Server Architecture:

- **Clients:** Web browsers (using Dart/Flutter web), Android devices (using Kotlin).
- **Server:** Cloud-based, utilizing Firebase services (Authentication, Firestore).
- **Communication:** Data is exchanged between client and server via HTTPS.

Technology Stack:

- **Frontend:**
- **Web:** Dart with Flutter framework for building a web application.
- **Android:** Kotlin with XML for UI design and Android framework for functionality, Dart and flutter framework.
- **Backend:** Firebase (Authentication, Firestore).
- **Database:** Firebase Firestore for storing user data, announcements, and other relevant informatio n.

3.3.2 Database Design (Firestore):

Collections:

- **Users:** Stores user data (full name, email, enrollment number, course, year, password).
- **Notifications:** Stores announcement data (title, body, timestamp).

3.3.3 User Interface Design:

- **Wireframing:** Create wireframes to define the structure and layout of each screen.
- **Mock-ups:** Develops high-fidelity mock-ups to visualize the final design with colours, fonts, and imagery.
- **Prototyping:** Build interactive prototypes to test user flows and interactions.

3.3.4 Database Design:

Tables:

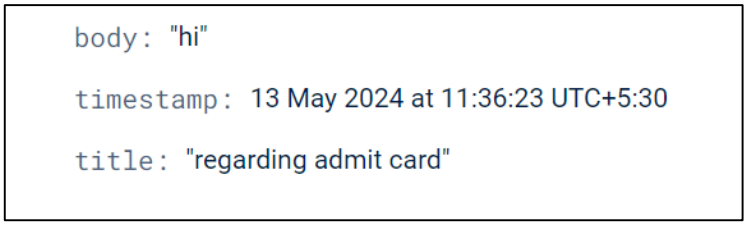
Users:

- Full name: “String”
- Email: “String”
- Course: “String”
- Year: “Numeric”
- No: “Numeric”
- Password: “String”

```
Course : "BCA"
enrollment no : 2100100695
full name : "shoaib"
gmail: "shoaibahmed12222234@gmail.com"
year: 3
```

Notifications

- Title: “String”
- Body: “String”
- Timestamp: “timestamp”



Relationships:

- Users have one Role
- Roles have many Users

User Interface Design:

- Wireframes and Mockups: Develop visual representations of each page to define layout a nd user interaction.

Design Principles:

- Focus on clarity, consistency, and ease of use.

Security Considerations:

- Authentication: Use strong password hashing algorithms and implement multi-factor authentication (if needed).
- Authorization: Implement rolebased access control to restrict actions based on user roles.
- Data Encryption: Secure sensitive data both at rest and in transit.

3.4 Implementation

- Development Methodology: Agile development with iterative sprints is recommended for flexibil ity and adaptability.
- Version Control System: Utilize tools like Git for managing code changes and collaboration.
- Testing: Implementing unit testing, integration testing, and user acceptance testing to ensure qual ity and functionality.

3.5 Deployment & Maintenance

- Deployment Environment: Choose a suitable hosting platform based on needs (cloud hosting, on-premise server).

- Monitoring: Implement system monitoring tools to track performance and identify potential issue s.

4. DETAILED LIFE CYCLE OF PROJECT

4.1 Entity Relationship (ER) Diagram

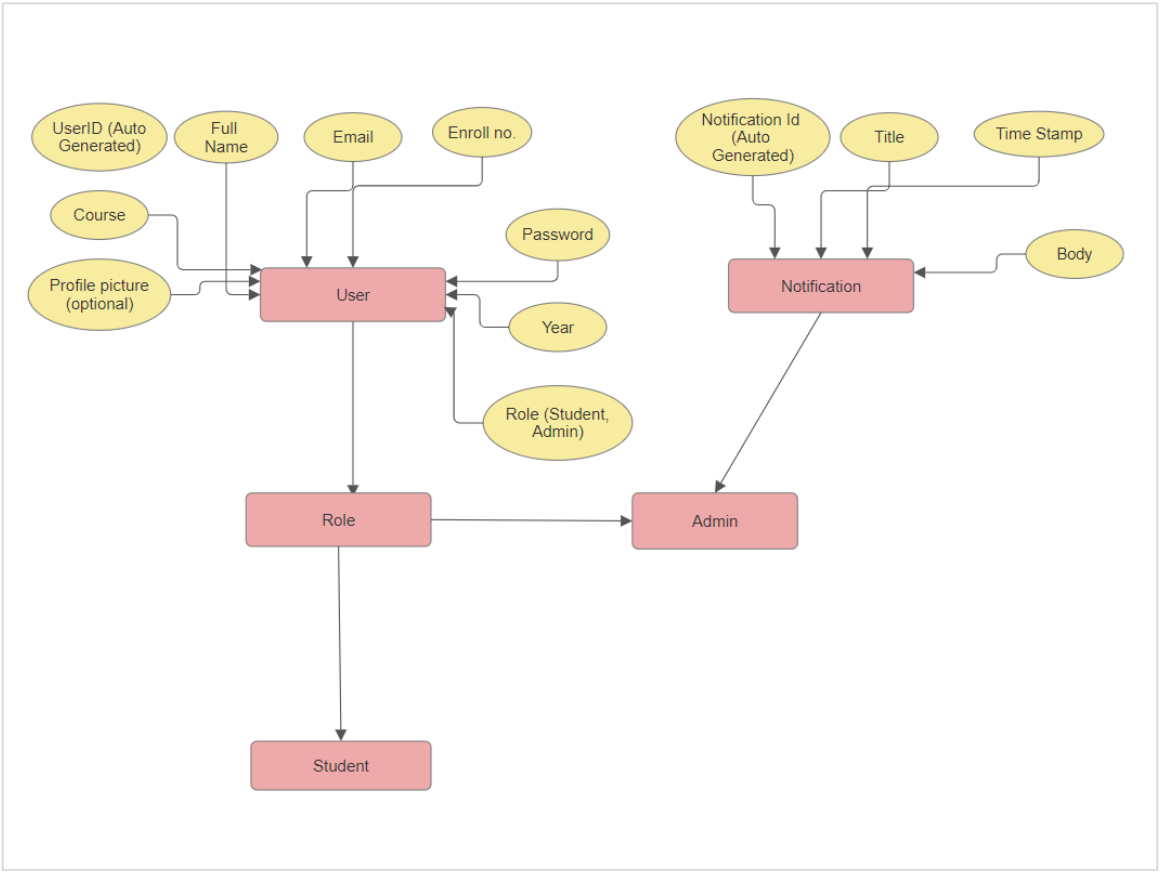


Fig 16. ER Diagram

Entities:

- **User:** Represents a user in the system.
- **Role:** Represents the role assigned to a user (e.g., Student, Admin).
- **Student:** Represents a student user, inheriting from the User entity.
- **Admin:** Represents an administrator user, inheriting from the User entity.
- **Notification:** Represents a notification sent within the system.

Attributes:

- **UserID (Auto Generated):** Unique identifier for each user, generated automatically.
- **Full Name:** User's full name.
- **Email:** User's email address.
- **Enroll no.:** User's enrollment number.

- **Password:** User's password.
- **Year:** The year the user is enrolled in.
- **Profile picture (optional):** Optional profile picture for the user.
- **Role (Student, Admin):** Specifies whether a role is for students or administrators.
- **Notification Id (Auto Generated):** Unique identifier for each notification, generated automatica lly.
- **Title:** Title of the notification.
- **Time Stamp:** Date and time when the notification was created.
- **Body:** The main content or message of the notification.

Relationships:

- **User -- Has a -- Role:** A user belongs to a specific role (Student or Admin). This is a one-to-one relationship.
- **Admin -- Sends -- Notification:** An admin can send notifications to users. This is a one-to-many relationship.
- **User -- Receives -- Notification:** A user can receive notifications from the admin. This is a many-to-many relationship, as a notification can be sent to multiple users, and a user can receive multiple notifications.

4.2 Data Flow Diagram

4.2.1 0 Level DFD

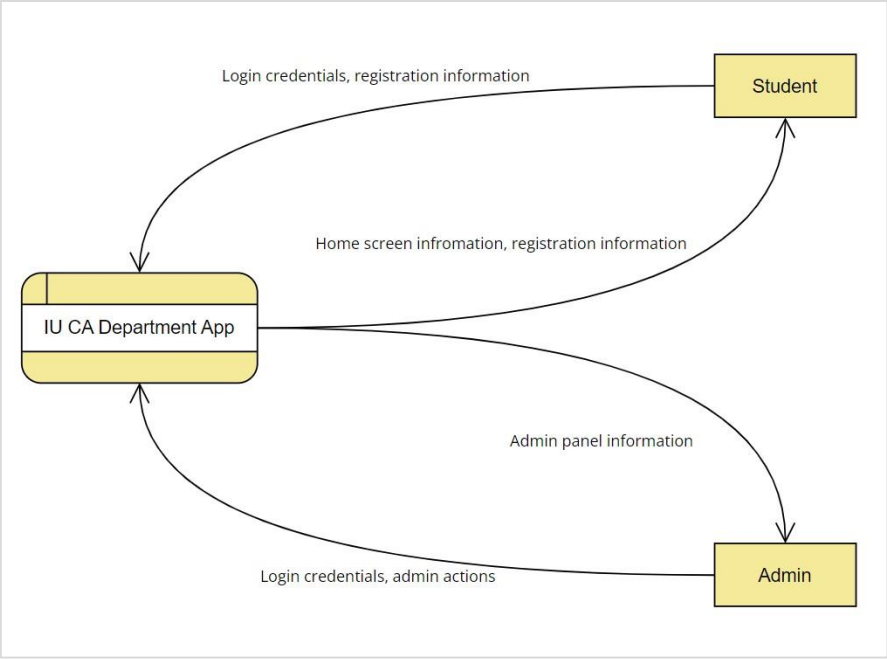


Fig 17. 0 Level DFD

Student: This box represents the student user.

Information sent to the app:

- Login credentials
- Registration information

Information received from the app:

- Home screen information
- Registration information (confirmation, etc.)

Admin: This box represents the administrator user.

Information sent to the app:

- Login credentials
- Admin actions (creating announcements, managing content, etc.)

Information received from the app:

Admin panel information (user data, system status, etc.)

4.2.2 1st Level DFD

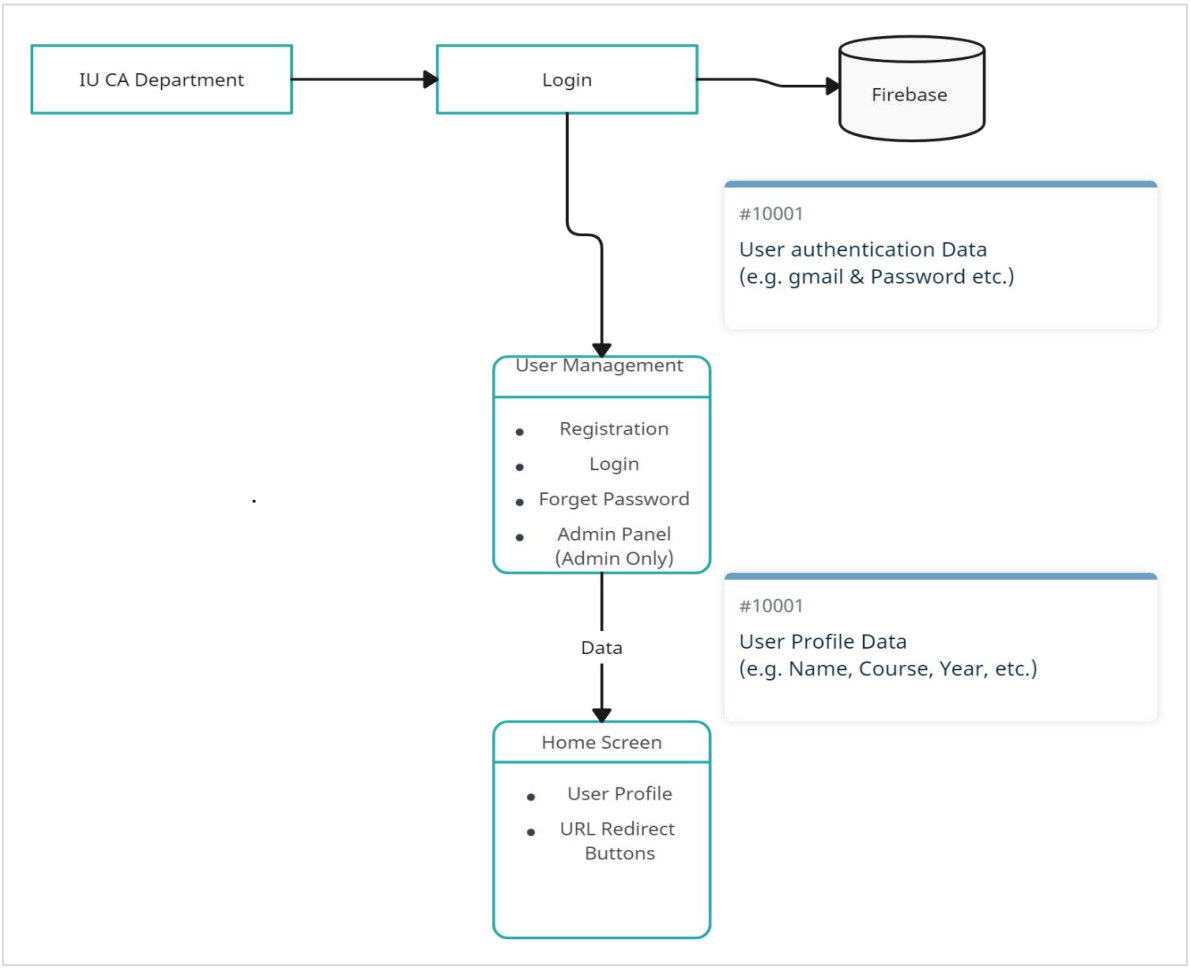


Fig 18. 1st Level DFD

IU CA Department: This likely represents the user interface (UI) of the application, where users interac t with the system.

Login: This indicates the login process where users provide their credentials.

Firebase: The cylinder represents the Firebase backend system, which includes authentication services a nd a database.

User Authentication Data: This box, linked to Firebase, illustrates the storage of user authentication da ta (email, password, etc.) within Firebase.

User Management: This block outlines the user management functionalities:

Registration: New users can create accounts.

Login: Existing users can log in with their credentials.

Forget Password: Functionality for password recovery.

Admin Panel (Admin Only): A specific section for administrative functions, accessible only to authoriz ed administrators.

User Profile Data: This box, also linked to Firebase, shows the storage of user profile data, such as nam e, course, year, etc. within Firebase.

Home Screen: This block represents the application's home screen, where users land after successful log in. Its functionalities include:

User Profile: Displays user-specific information.

URL Redirect: Redirects users to various university resources via URLs.

Buttons: Buttons for navigating different sections of the application.

Data Flow:

- A user interacts with the IU CA Department application UI.
- For login or registration, the app communicates with the Firebase backend.
- Firebase authenticates the user against the stored User Authentication Data (#10001).
- If authenticated, user profile data (#10001) is retrieved from Firebase.
- The Home Screen is displayed, populated with user profile information and relevant buttons for navigation and accessing resources.

4.3 Use Case Diagram

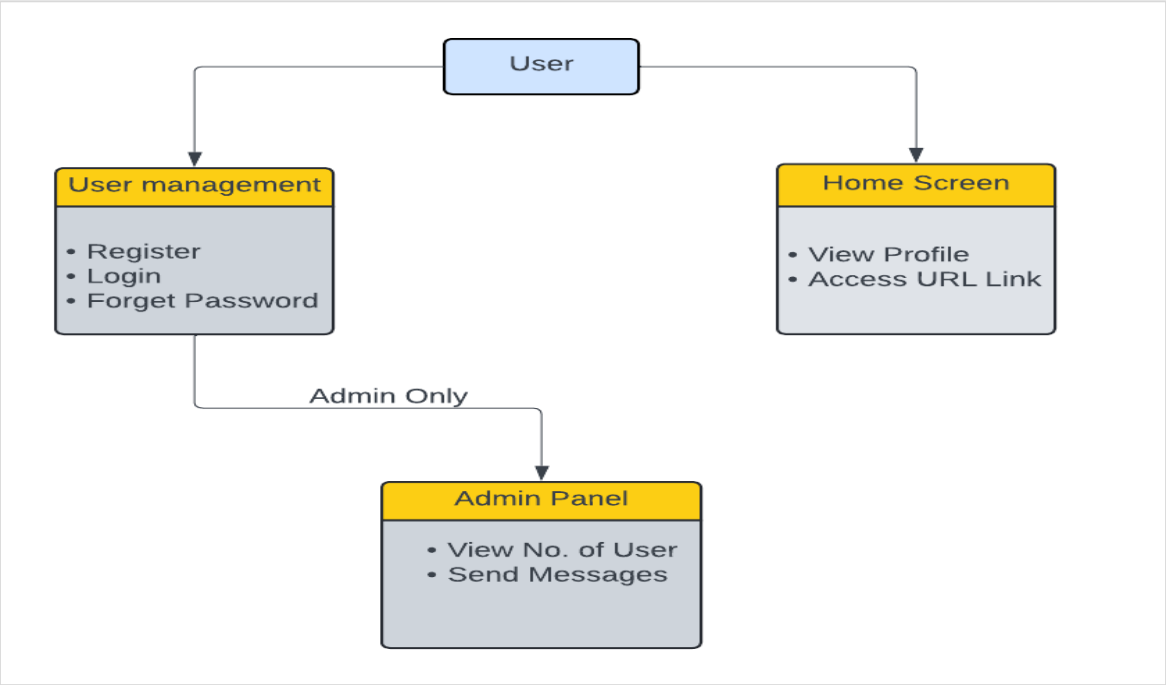


Fig 19. Use case Diagram

User Interaction:

1. **The Starting Point:** The "User" box signifies the user's initial entry into the system. This could be through a login screen, registration page, or even simply visiting the website's homepage.

2. **User Management:** This section deals with user accounts.

- Register: This option enables new users to create accounts, likely by providing details like email, username, password, etc.
- Login: This allows existing users to access their accounts using their registered credentials.
- Forget Password: This is crucial for security and user experience. Users can initiate a password r ecovery process, often through email verification or security questions**.**

3. **Home Screen:** This is the core user interface.

- View Profile: This function allows users to view and potentially modify their account informatio n like profile picture, contact details, and other settings.
- Access URL Link: This offers users quick access to specific resources or features within the syst em. It could be links to particular sections of the website, external links, or downloadable content.

Admin Privileges:

1. **Admin Only:** This label implies a separate access level for administrators. It highlights that the " User Management" section has advanced features that are restricted to administrators.

2. Admin Panel: This is the control center for administrators.

- View No. of User: This provides administrators with an overview of the user base, allowing them to track growth and engagement.
- Send Messages: This empowers administrators to communicate with users, for instance, sending updates, announcements, or resolving issues.

4.4 Activity Diagram

4.4.1 Admin Activity Diagram

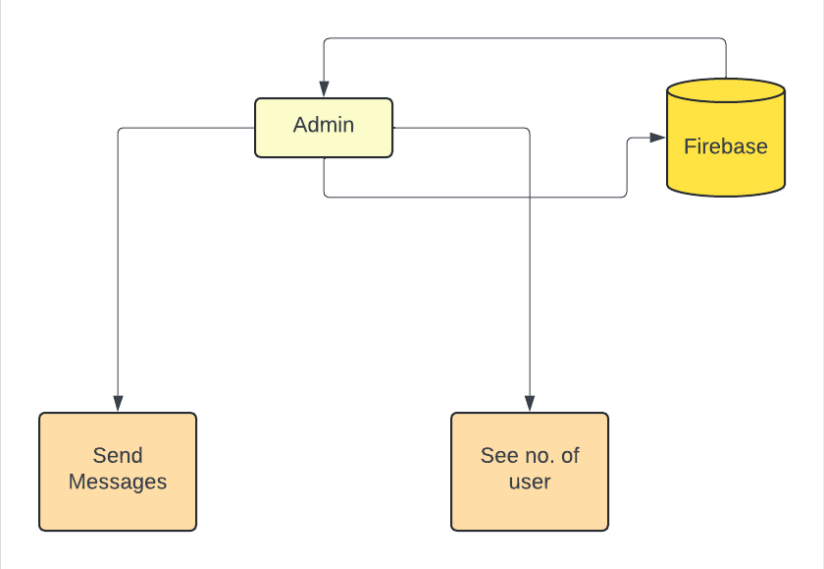


Fig 20. Admin Activity Diagram

Admin: This box represents the starting point, likely a user interface (UI) or an administrative panel wh ere an administrator interacts with the system.

Firebase: The cylinder represents the Firebase backend, likely a database or a cloud function, responsibl e for storing and processing data.

See no. of user: This box represents an action performed by the admin: retrieving the total number of us ers within the system. The arrow points towards the Admin box, indicating that the retrieved information is displayed to the administrator.

Send Messages: This box represents another action initiated by the admin: sending messages to users. T he arrow points towards Firebase, indicating that the admin sends the messages, and Firebase handles th e delivery or storage of these messages.

Data Flow:

- The admin interacts with the system.

- The admin initiates a request to Firebase to view the number of users.
- Firebase retrieves the user count data and sends it back to the admin to be displayed.
- To send messages, the admin inputs the messages and sends them to Firebase.

4.4.2 Student Activity Diagram

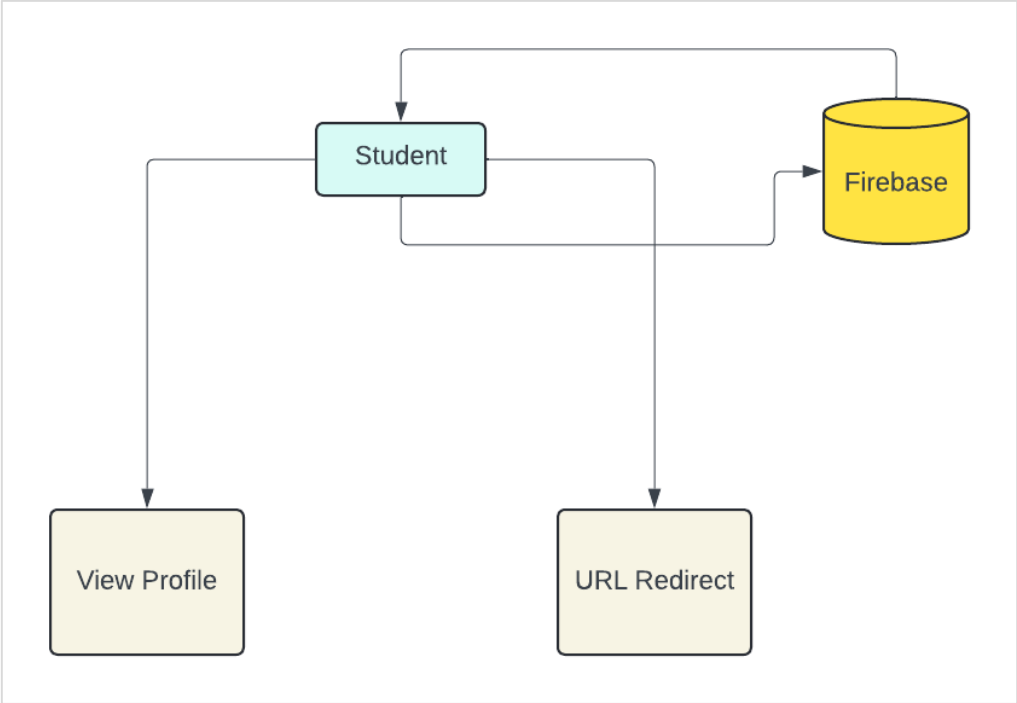


Fig 21. Student Activity Diagram

Roles:

1. **Student:** This represents the primary user of the system, a student accessing the application.

- **Actions:** The student interacts with the system by logging in, accessing their profile, and potentia lly engaging with other features.

2. **Firebase:** This is a cloud-based, NoSQL database, widely used for real-time data and mobile application development.

- **Purpose:** Firebase acts as the central repository for storing and managing student data.
- **Capabilities:** It enables efficient storage, retrieval, and synchronization of data across multiple d evices.

The Flow of Information:

1. **Student Entry:** The "Student" box represents the entry point for the student into the system. Thi s a login screen where someone authenticate using their credentials (username/password or a single sign-on method).

2. **Data Interaction:** The arrow connecting "Student" to "Firebase" signifies that the student interacts with the database. This involves reading (retrieving) and potentially updating (writing) information stored within Firebase.

3. **"View Profile" Function:** This function enables students to access their personalized information stored in the Firebase database.

- **Content:** This include details like their name, student ID, academic records, program of study, contact information, or even personalized course details.
- **Interface:** The actual "View Profile" screen would likely display this information in an organized and user-friendly format.

4. **"URL Redirect" Function:** This function allows the system to dynamically direct the student to different pages or external resources based on specific conditions.

- **Triggering Conditions: This could be based on the role (e.g., student vs. Admin),** accessing URL redirect functions and viewing their own profile.
- **Example:** The system redirect students to their Attendance, Profile, Time-table, IUSMS, ILI, etc.

5. **Circular Interaction:** The arrows from "Student" to itself and to "Firebase" indicate a continuous exchange of information. This suggests that students can repeatedly access their profile information, interact with the system, and trigger further actions that involve reading or updating data stored within Firebase.

4.5 Snapshots

Fig 22. Splash Screen

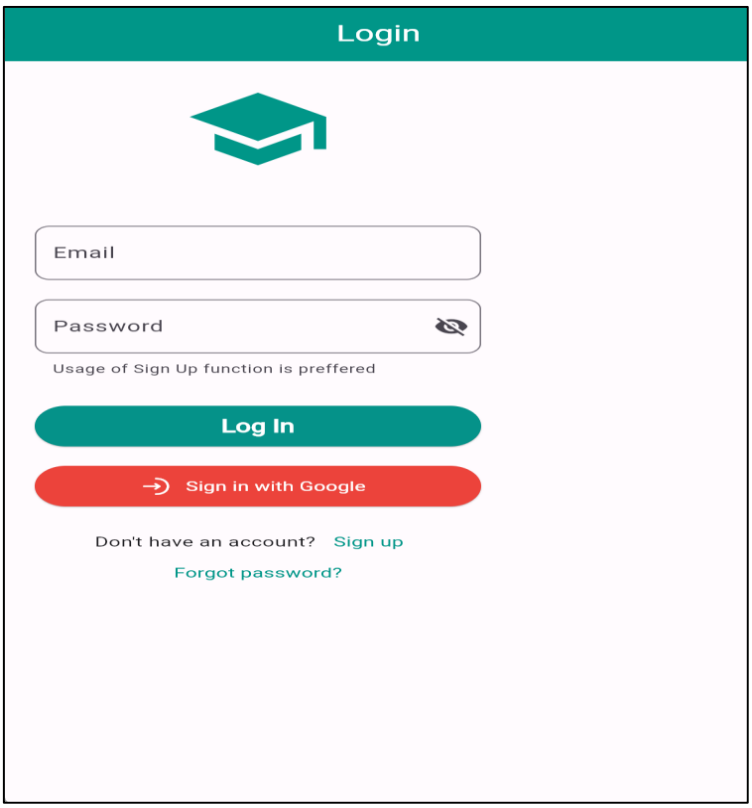


Fig 23. Login Page

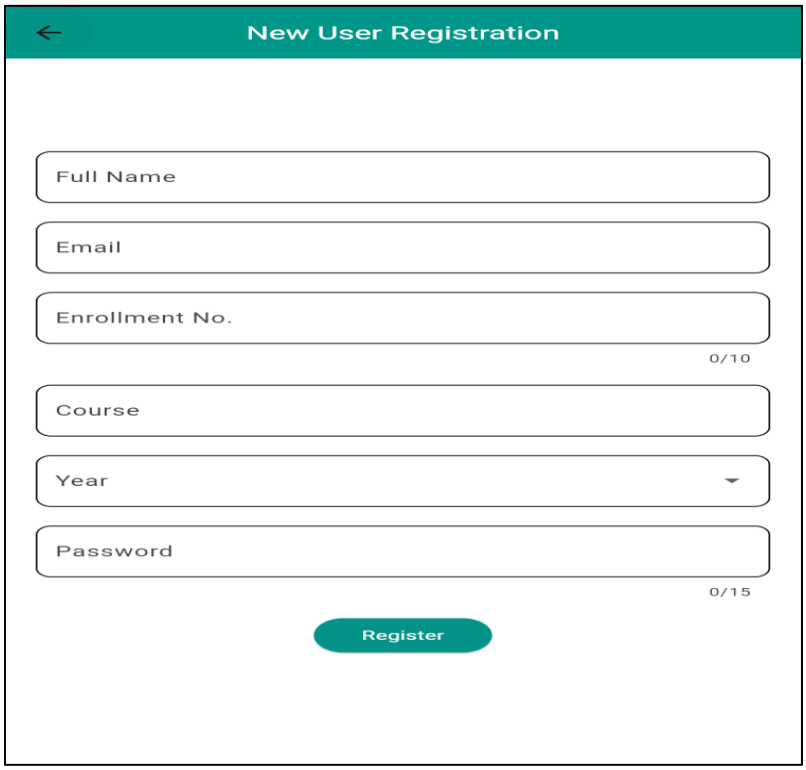


Fig 24. Registration Page

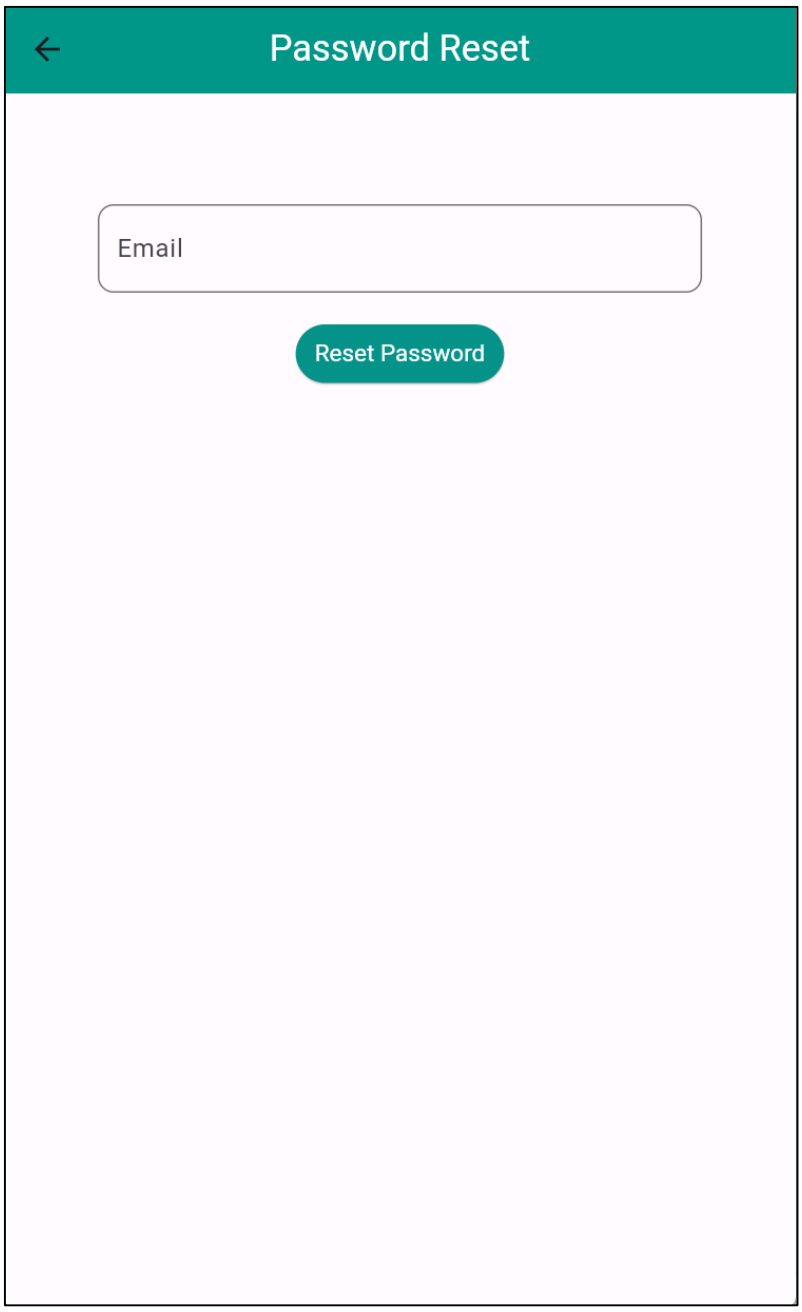


Fig 25. Password reset

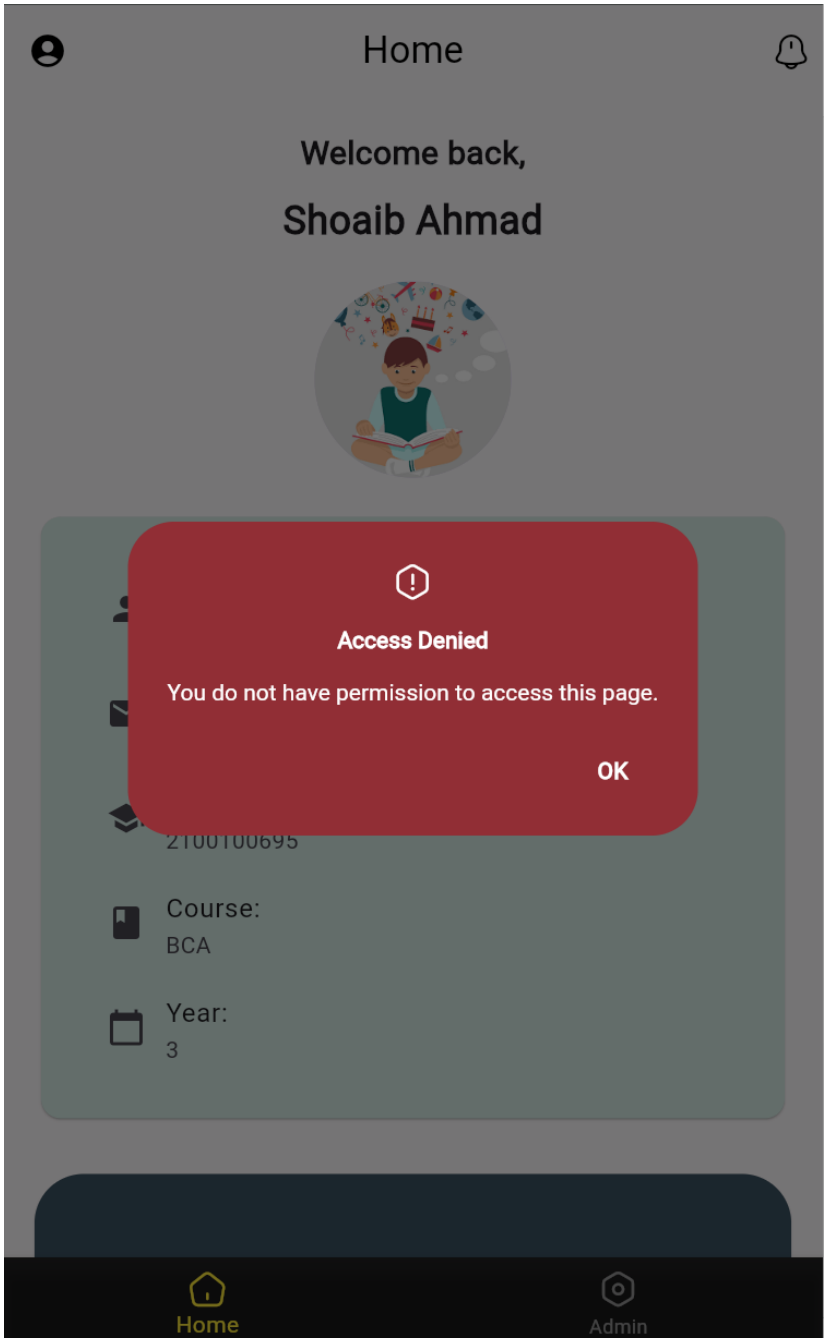


Fig 26. Alert Dialog

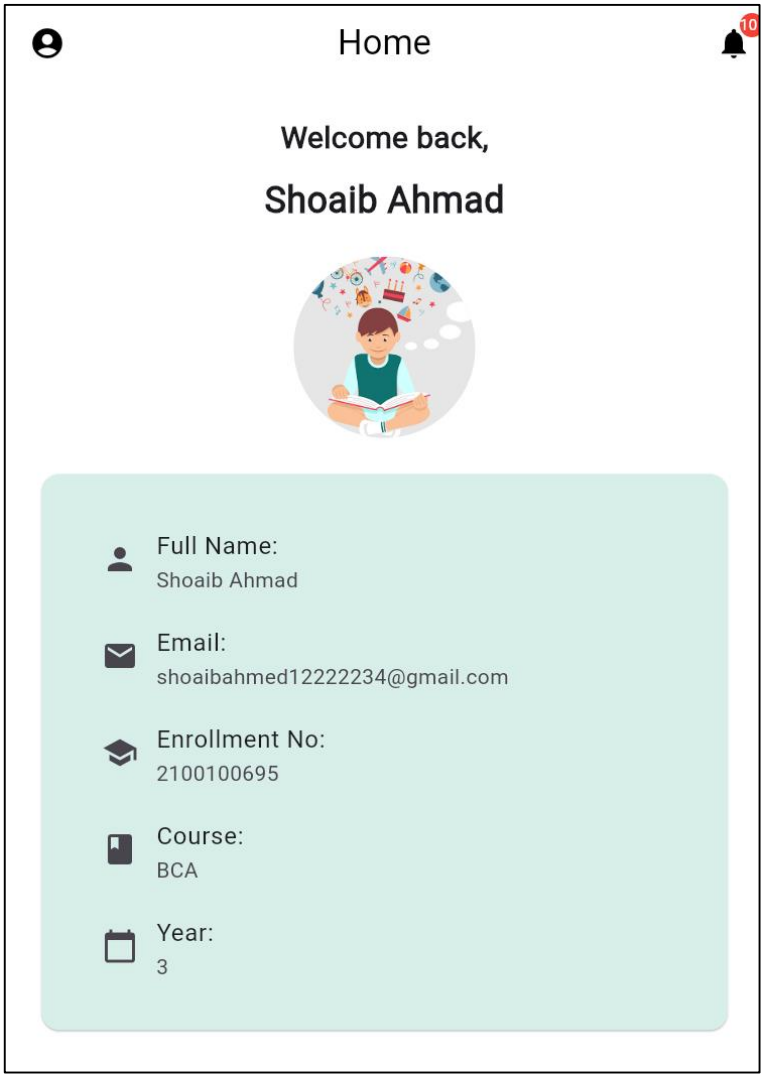


Fig 27. User Profile

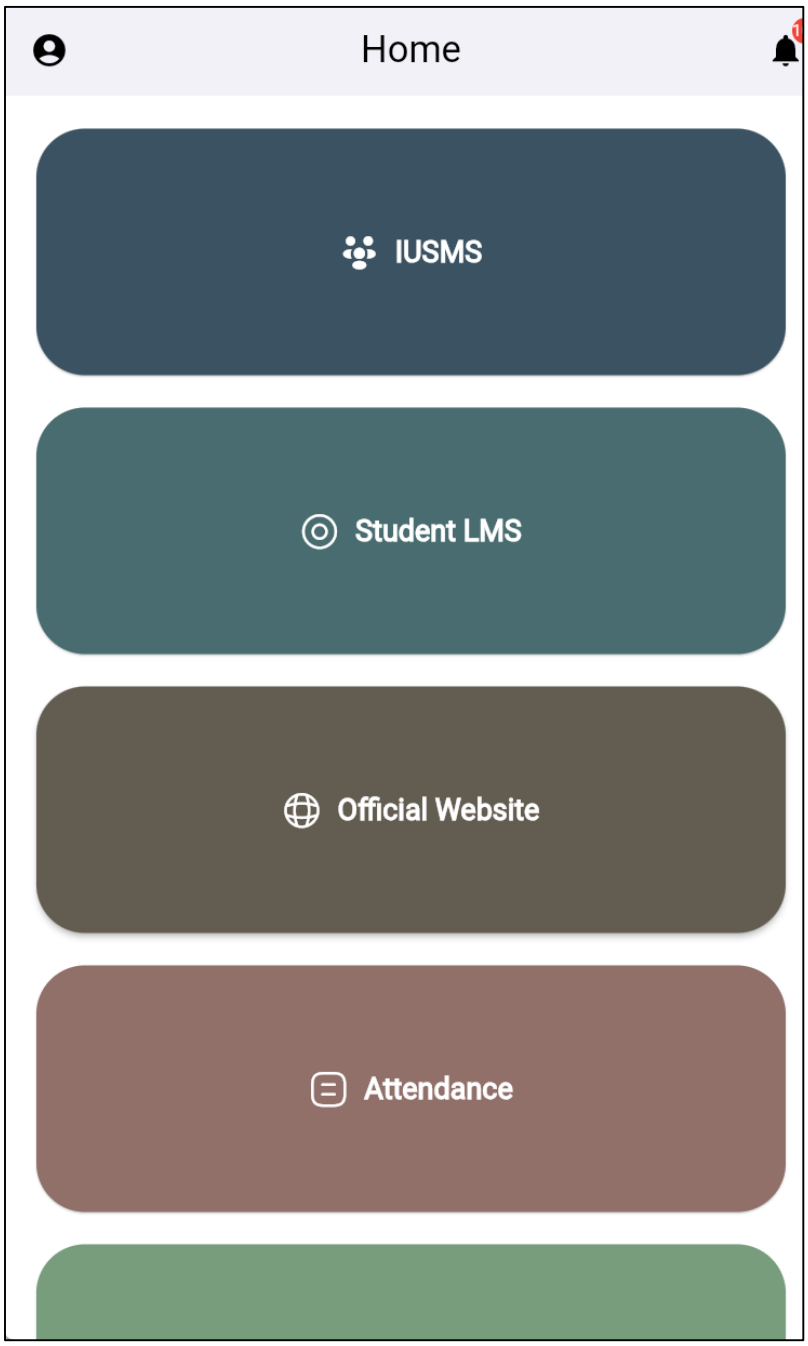


Fig 28. URL Redirect buttons

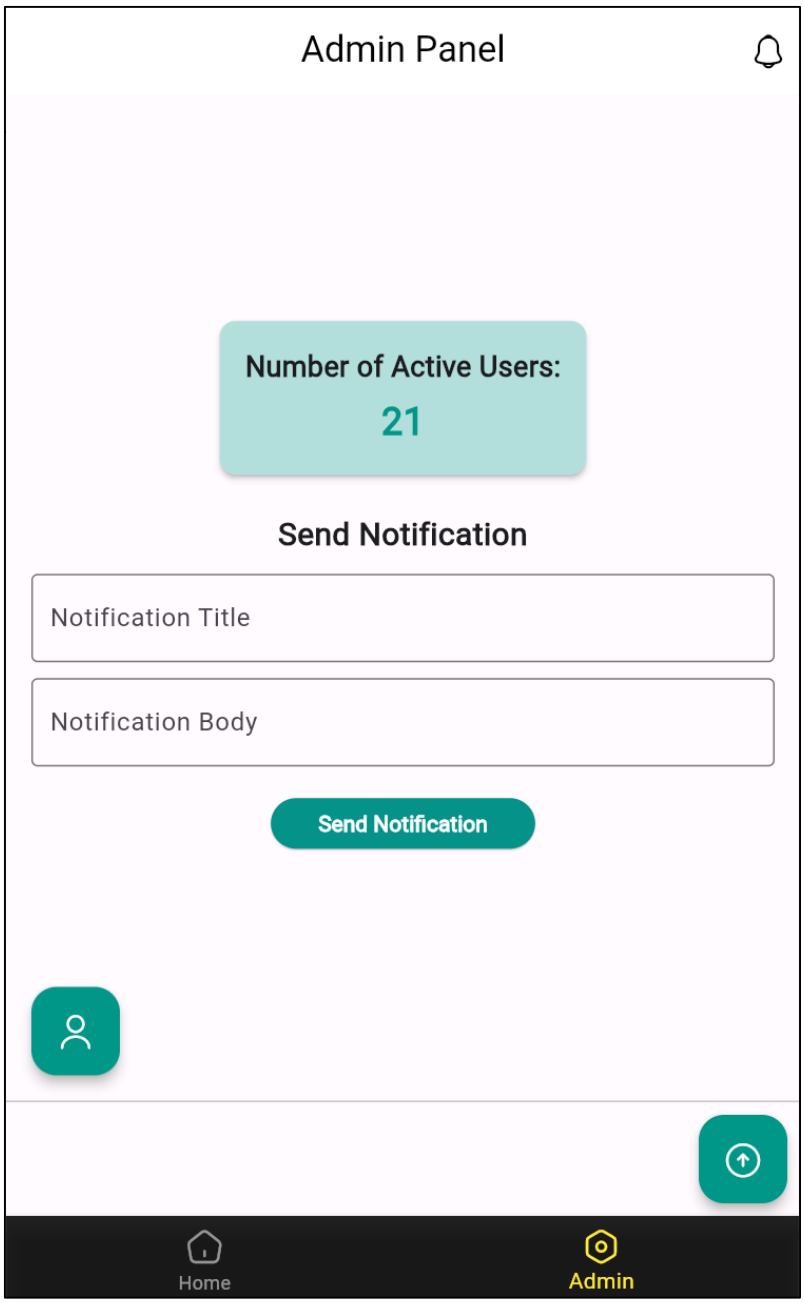


Fig 29. Admin Panel

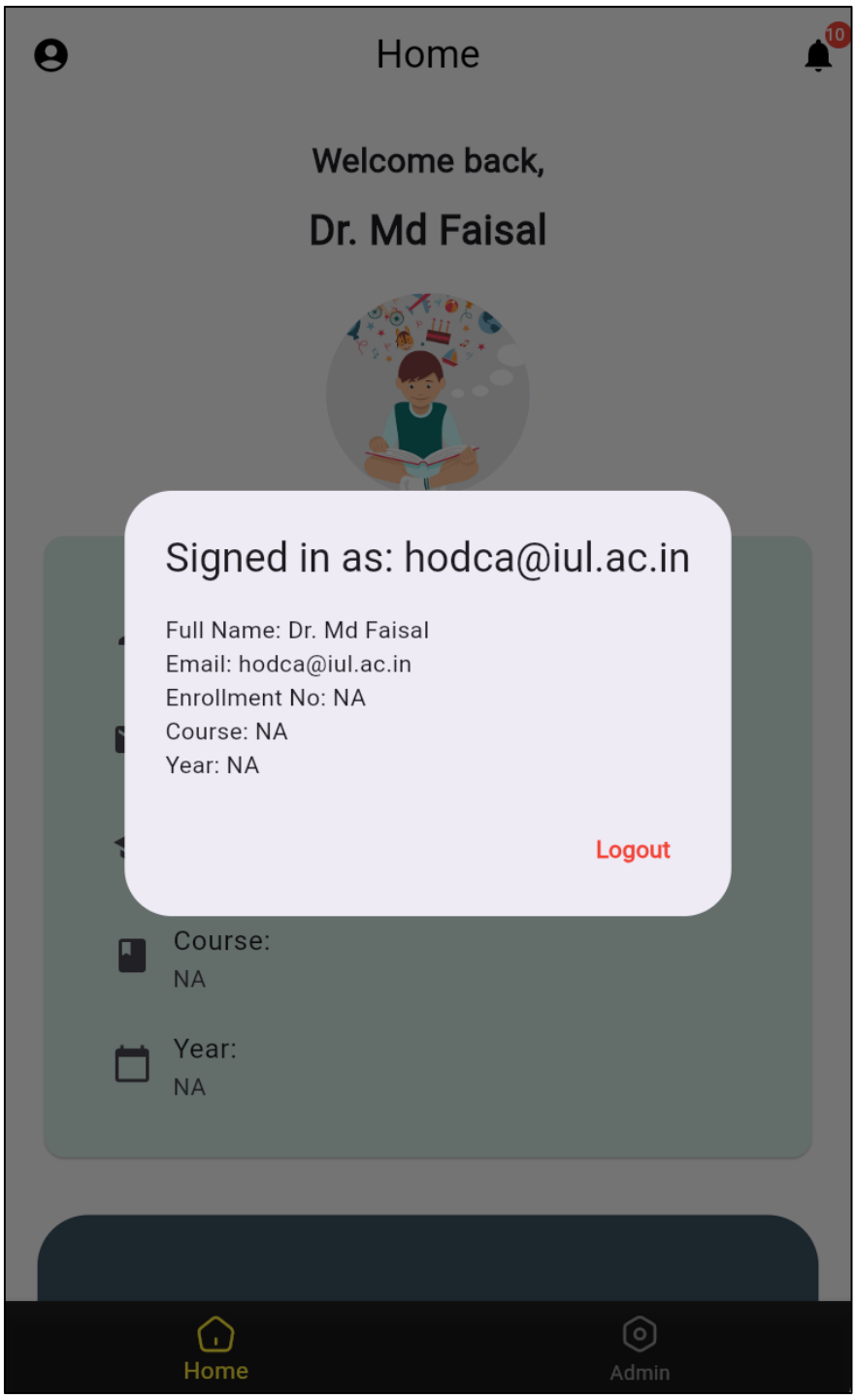


Fig 30. Logout Screen

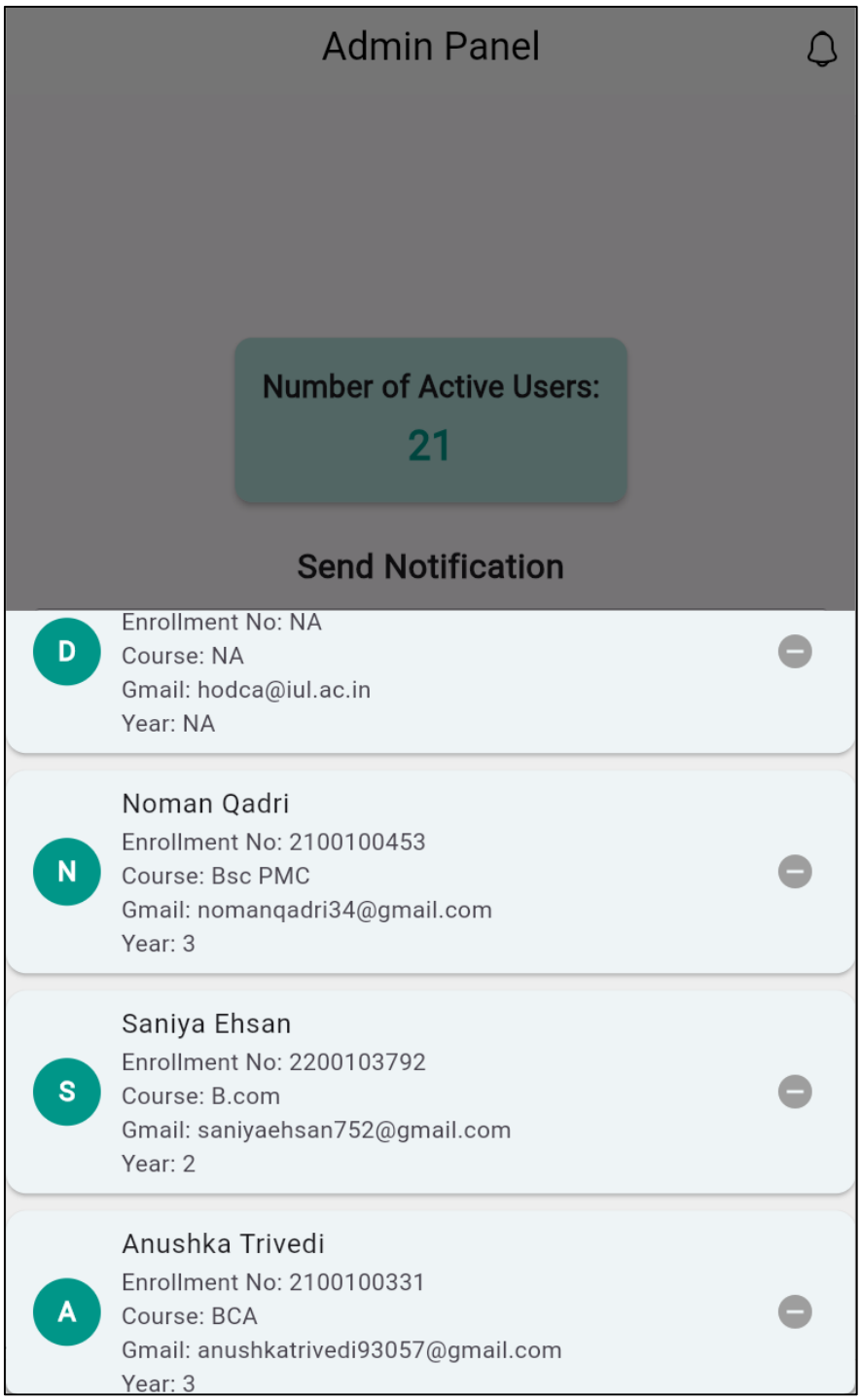


Fig 31. User Data

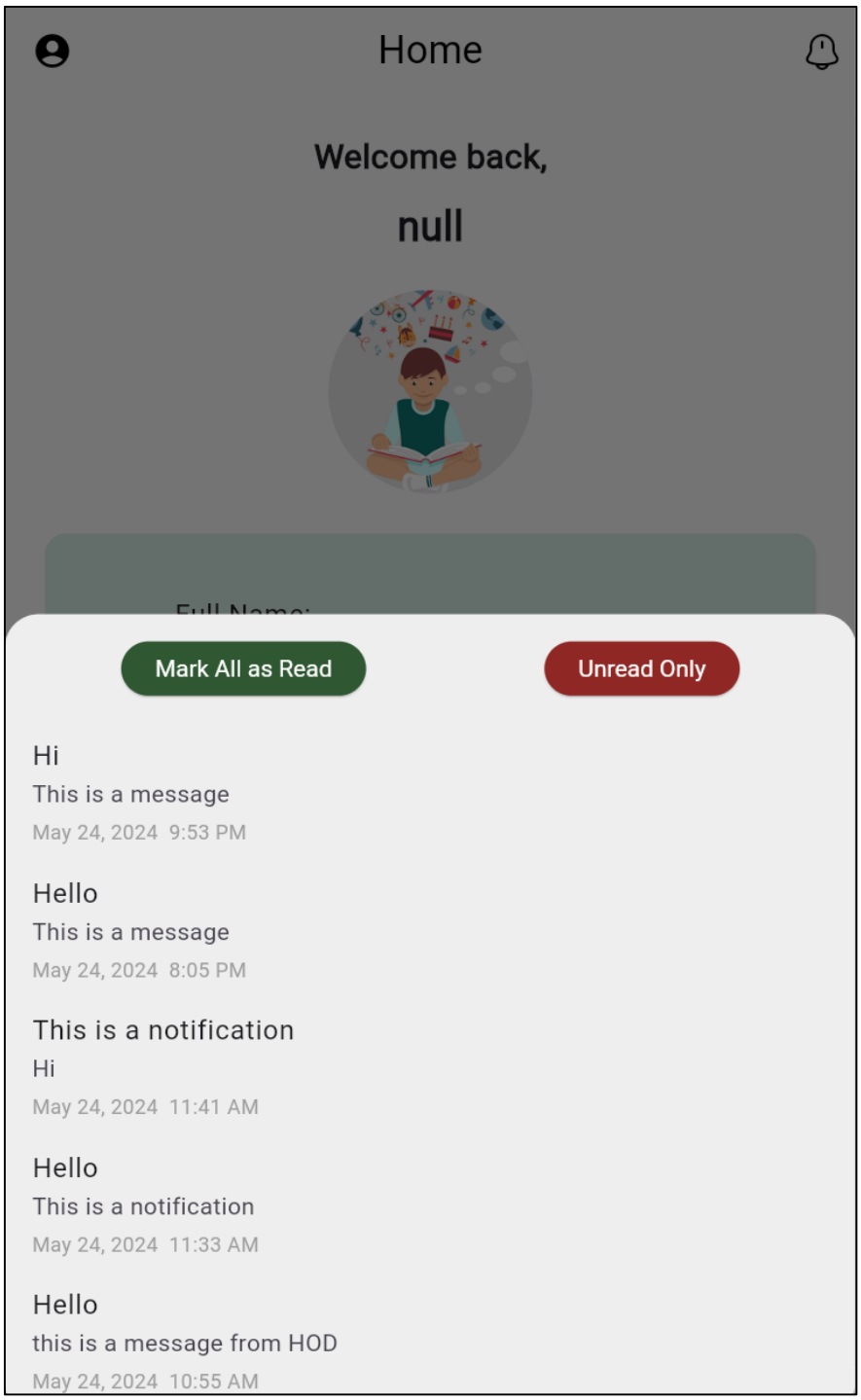


Fig 32. Received Notification Screen

4.6 Database Tables

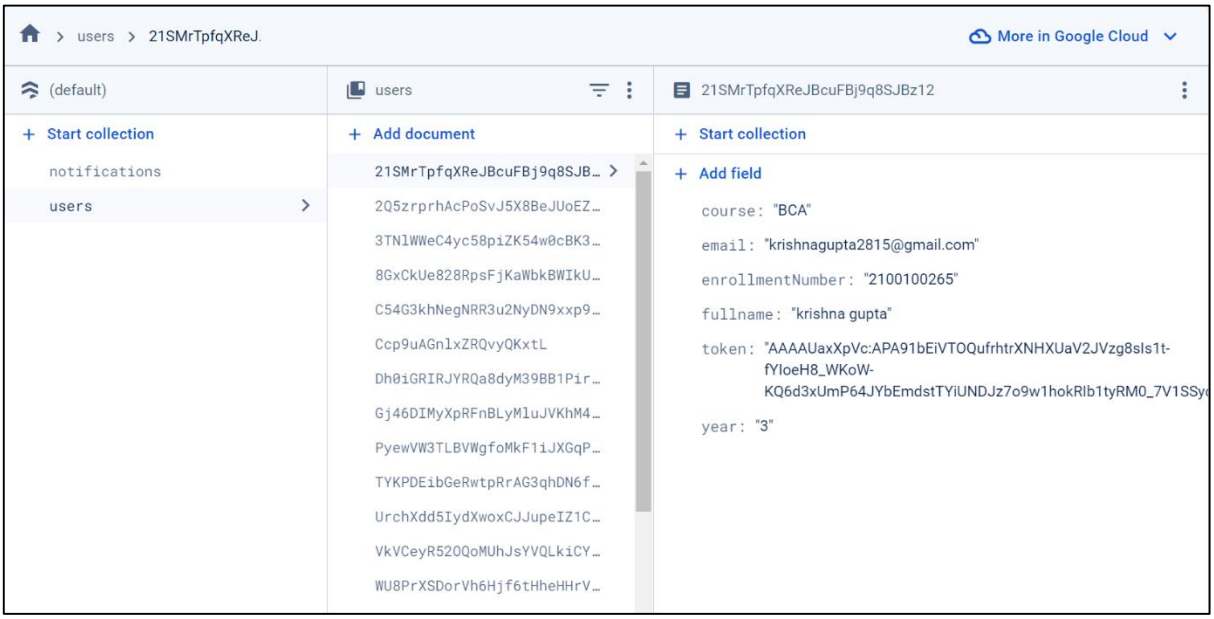


Fig 33. User Database Table

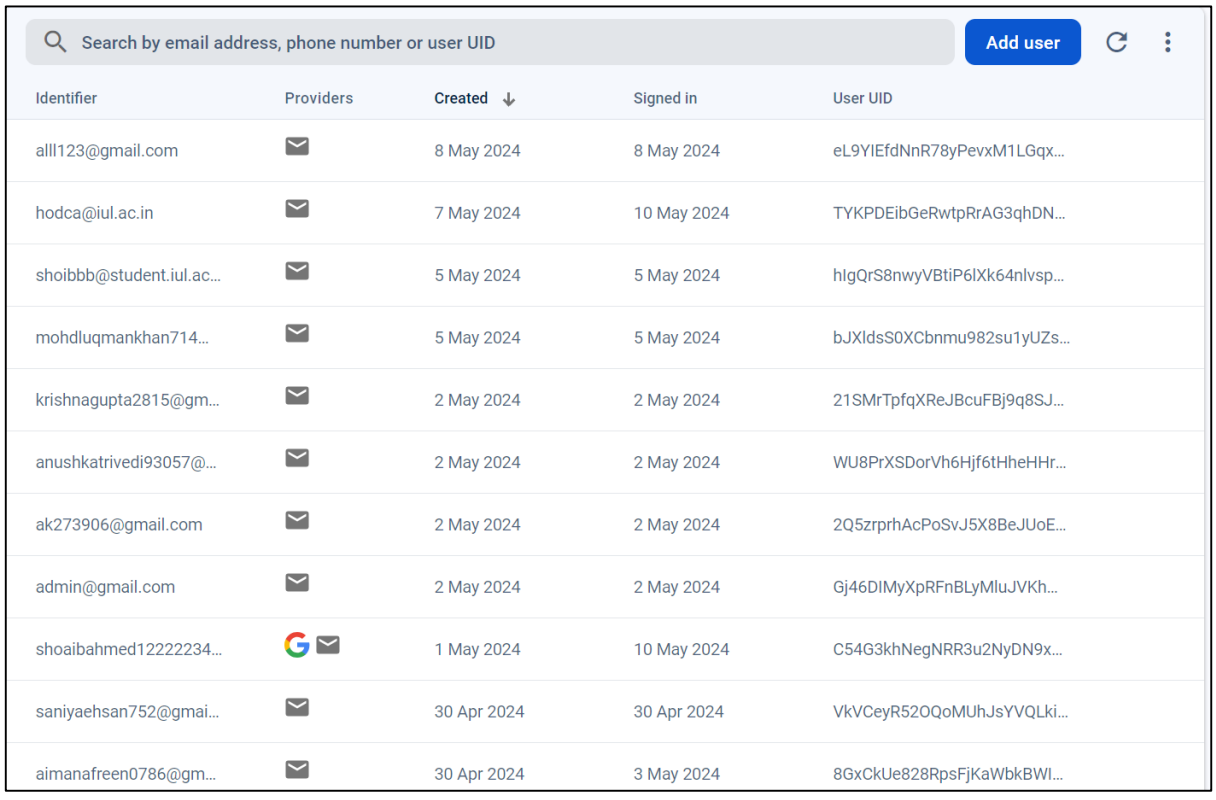


Fig. 34. User Authentication Table

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

The goal of this project was to increase productivity and communication at Integral University by creating a comprehensive IU CA department administration application using Flutter and Dart. Key features of the application include announcement and messaging systems, tracking of attendance, and maintenance of student profiles. The development team maintained a user-friendly design and effectively integrated robust features despite initial obstacles. All things considered, this project shows how well Flutter and Dart work together to create user-friendly and productive mobile applications for educational organizations.

5.2 Future Work

Important considerations are taken into account when creating the IU CA Department so that it can grow in the future. This can be expanded based on the administrator's needs. In the future, the administrator will have the option to personally alert each student. Students will also be able to view the real-time attendance tracking and receive automated warnings if their attendance is below expectations. This feature will allow notifications to be sent to both parents and students. Additionally, a new module called "staff and teachers" may be introduced. Teachers and staff can also utilize this software, and it can be scaled to a higher level with the addition of these resources.

6. REFERENCES & APPENDICES

6.1 References

1. Dart overview. (n.d.). Dart. https://dart.dev/overview
2. Flutter architectural overview. (n.d.). Flutter. https://docs.flutter.dev/resources/architectural-overview
3. FlutterFire Overview | FlutterFire. (n.d.). https://firebase.flutter.dev/docs/overview/
4. GeeksforGeeks. (2021, July 15). Firebase Introduction. GeeksforGeeks. https://www.geeksforgeeks.org/firebase-introduction/
5. Wikipedia contributors. (2024, February 2). Android SDK. Wikipedia. https://en.wikipedia.org/wiki/Android_SDK#:~:text=The%20Android%20SDK%20is%20a,%2C%20sample%20code%2C%20and%20tutorials.
6. SDK Platform Tools release notes. (n.d.). Android Developers. https://developer.android.com/tools/releases/platform-tools

6.2 Appendices

A: Project Overview

The "IU CA Department Application" has been developed to address the challenges associated with the manual systems currently in use. This application aims to streamline processes and enhance efficiency by providing a user-friendly interface tailored to the specific requirements of Integral University's CA Department.

B: Problem Definition

The application integrates content from key university websites, namely IUSMS, IUL, and ILI, to provide a centralized platform for students. It eliminates the need to navigate multiple websites, simplifying access to essential information.

C: User Panel

Home: Provides access to the application's main homepage and user profile.

Registration Form: Allows new users to register and log in securely.

Password Reset: Enables users to retrieve forgotten passwords.

Logout Alert Dialogue: Allows users to log out securely and confirms their identity.

Profile Section: Displays the user's complete profile.

URL Redirect Section: Provides access to relevant websites for students.

D: Admin Panel

Accessible only to administrators, this panel allows for actions such as viewing logged-in users and sending notifications to students.

E: Hardware & Software Requirements

Outlines the hardware and software configurations necessary for running the application, including operating systems, IDEs, and development tools.

F: Software Features

Details the features of the IDE, Dart language, Flutter framework, Android SDK tools, and Google Firebase, which form the technological foundation of the application.

G: System Analysis and Design

Includes requirements gathering, functional and non-functional requirements, system architecture, database design, user interface design, and security considerations.

H: Implementation

Describes the development methodology, version control system, testing procedures, and deployment strategies for the application.

I: Conclusion and Future Work

Summarizes the project's objectives, achievements, and plans for future enhancements, including personalized student alerts, real-time attendance tracking, and expansion to include staff and teachers.