# Rapport end of semester project

# Chatter application

**Subject : Flutter cross platform**

**Produced by :**

**Med Saad Jlassi.**

## Chatter application documentation

# Introduction

This application represents the final project for the subject: Flutter cross platform, which puts into effect all the knowledge and skills we've worked on gaining during the entire semester: the creation of the chatter mobile application.

# 1. General Presentation of the application

Chatter is a free Flutter instant messaging system that allows you to chat with your contacts and provide other multiple services to the user.

## 1.1 Requirement specification

This phase consists of understanding the context of the system. It consists in determining the most relevant functionalities and actors.

★ **Actor's identification**

We have a single actor which is the user :
Whoever has an account and wants to send or receive messages through our application.

★ **Functional requirements**

The functional requirements of a system describe the purpose of the system to be developed.
Chatter application allows the user to :

- Authenticate
- Send and receive messages from other accounts
- Create new conversations
- Check your profile informations
- Check list of contacts and notifications
- Check list of stories
- Log out

★ **Non-functional requirements**

Non-functional requirements are either optional requirements or implementation requirements.The main non-functional requirements of our application are:

- Reliability

The application must run consistently, error-free and satisfactorily. Most importantly, it must be stable, as it will be handling a large amount of data .

- Scalability

the application must be able to respond to future enhancements without causing problems or bugs, the extension must be served seamlessly and without interruption.

- Ergonomics

Ergonomics is based on the ease of use, on the way of navigating on it, on its portability through the various supports and digital terminals and finally, on its accessibility.

## 1.2 Technological choices

### ★ Software Tools

Github

GitHub est un service d'hébergement Internet pour le développement de logiciels et le contrôle de version à l'aide de Git. Il fournit le contrôle de version distribué de Git ainsi que le contrôle d'accès, le suivi des bogues, les demandes de fonctionnalités logicielles, la gestion des tâches, l'intégration continue et les wikis pour chaque projet.

Visual Studio Code

Un éditeur de code source réalisé par Microsoft avec le Framework Electron, pour Windows, Linux et macOS. Les fonctionnalités comprennent la prise en charge du débogage, la coloration syntaxique, la complétion de code intelligent, les snippets, le remaniement du code et Git intégré.

StarUML

Un outil d'ingénierie logicielle pour la modélisation de systèmes utilisant le langage de modélisation unifié, ainsi que le langage de modélisation des systèmes et les notations de modélisation classiques.

### ★ Frameworks

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross-platform applications for Android, iOS, Linux, macOS, …

### ★ Programming languages

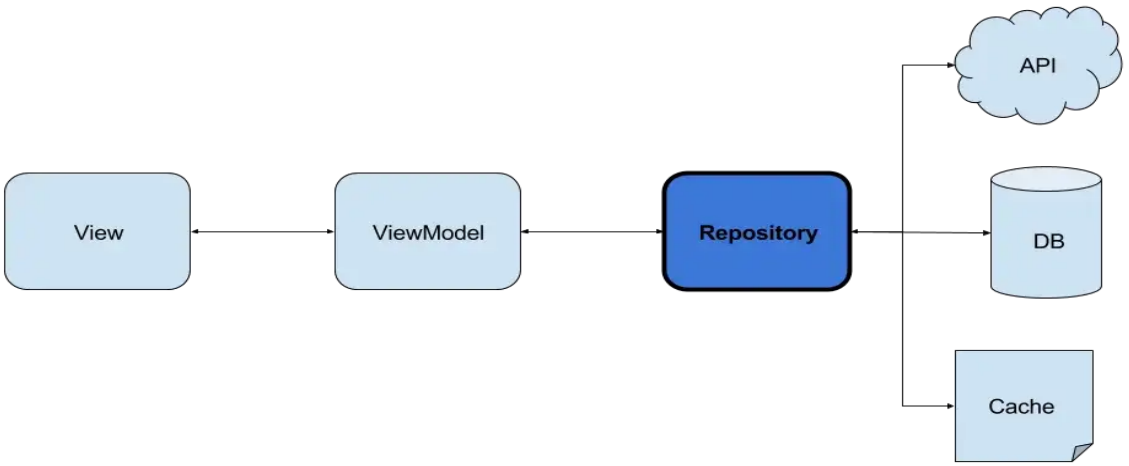
Dart

Dart is a programming language optimized for applications on multiple platforms. It is developed by Google and is used to create mobile, desktop, server and web applications. Dart is an object-oriented, garbage-collected language with a C++-like syntax.

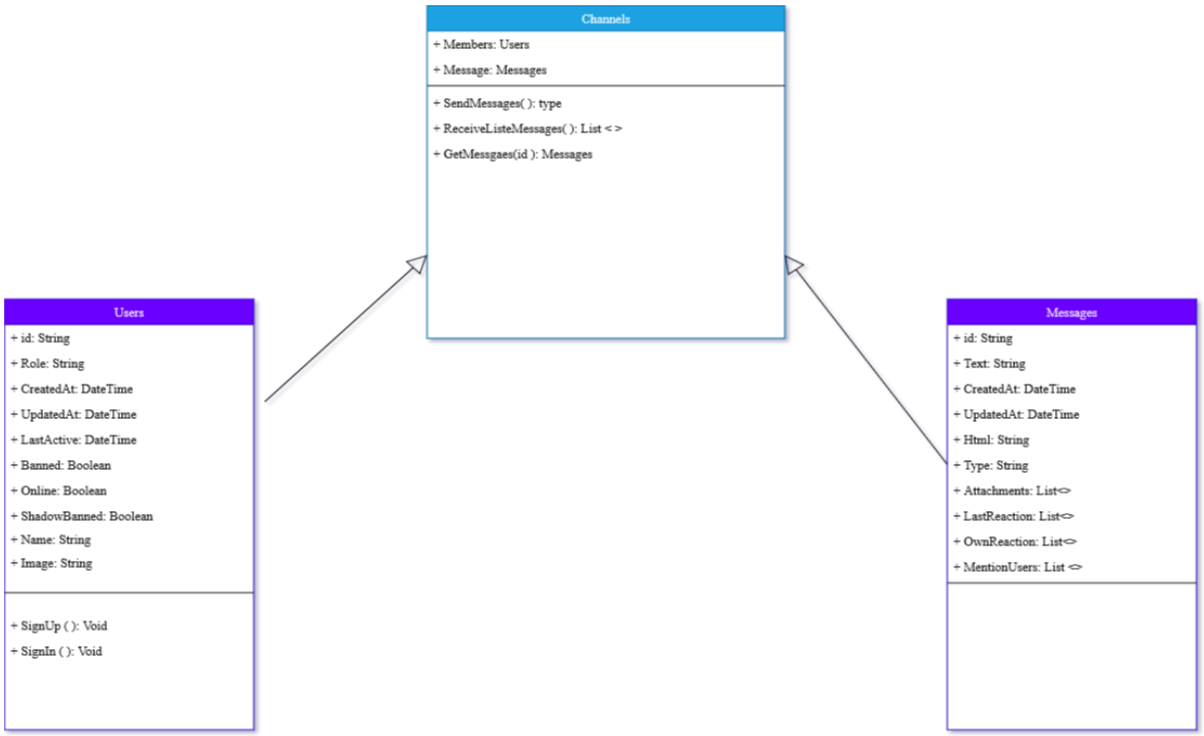
## 1.3 Architecture

For the creation of our chat application we needed an architecture to communicate between the UI & business logic, so **MVVM** is proving this in an easy way as you can hold all your business logic inside the ViewModel class and UI separately.
The key benefit of using MVVM are:

1. Business logic is separate from the UI
2. The view is independent from the ViewModel class and only reading the state from ViewModel
3. Code will be easy to maintain and update in terms of logic & UI
4. Easy to Write the test cases for the project.




## 1.4 Class diagram

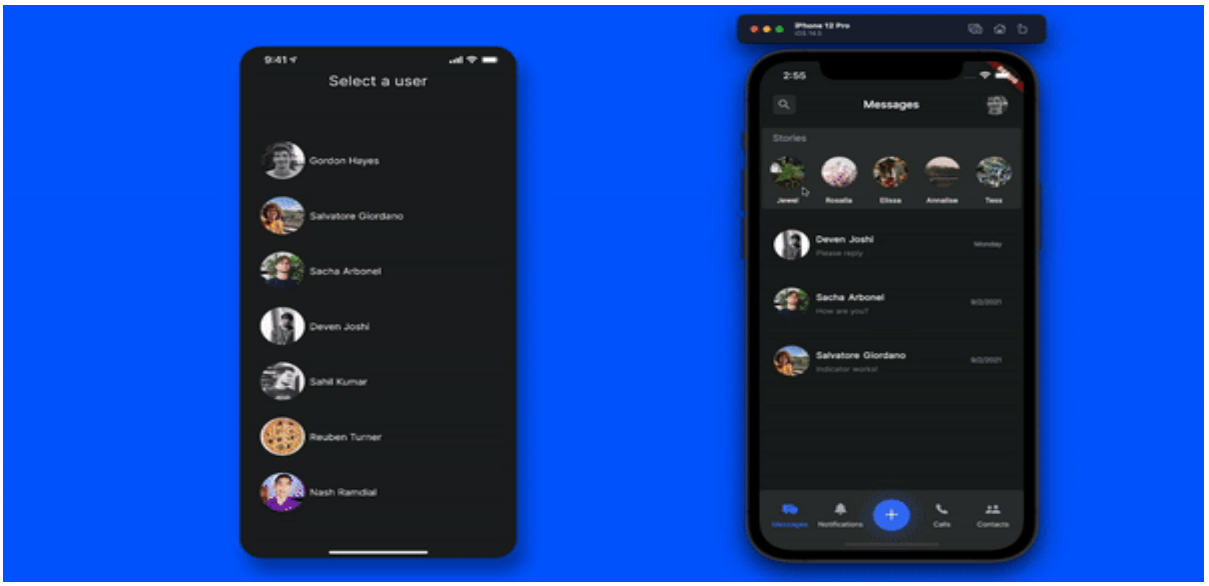



# 2. Project creation

## 2.1 Main steps

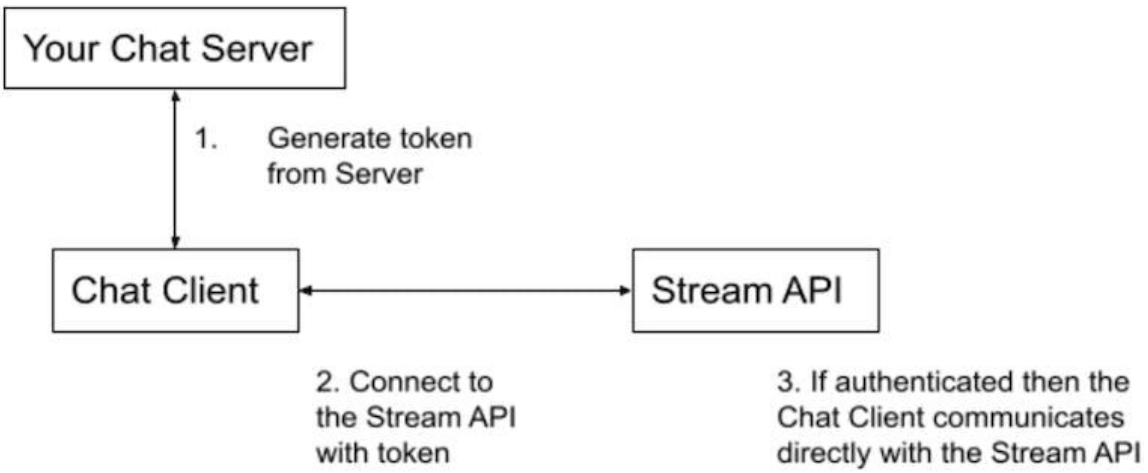
★ **Design/UI**

UI is the abbreviation of user interface. The UI design refers to the graphic environment in which the user of a software, a website or an application evolves. We created a pleasant and practical interface as the first step of the creation process.



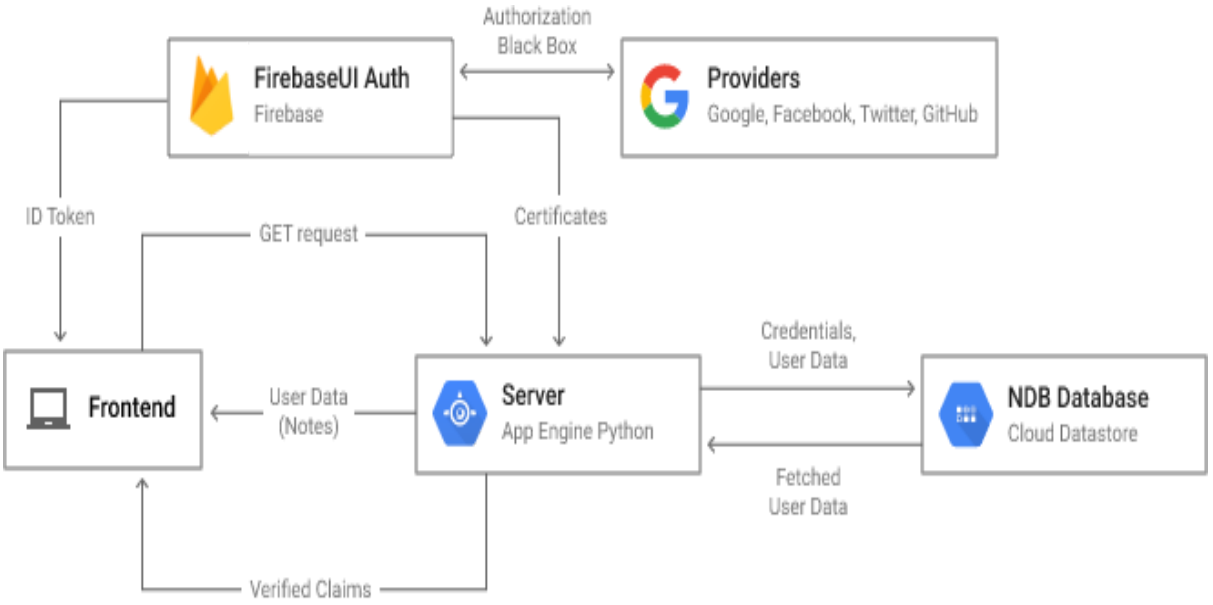

★ **Stream API implementation**

The Stream API is a powerful, but simple to understand set of tools for processing the sequence of elements.




★ **Firebase Auth and cloud functions**

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app.
Cloud Functions lets you treat all Google and third-party cloud services as building blocks.




## 2.2 Dependencies

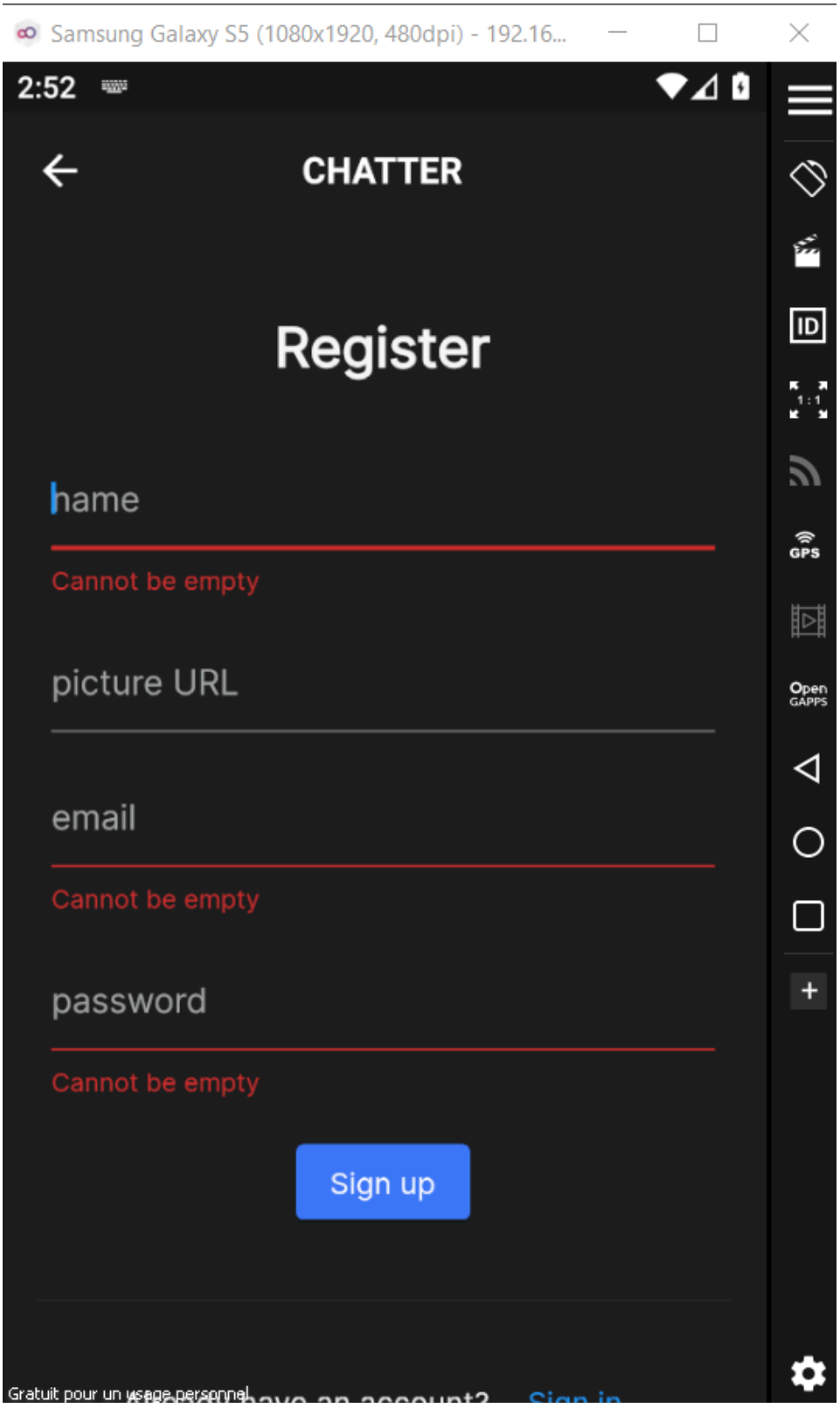
We added the following dependencies:

- **Cupertino Icons** Gallery | An Open-Source Home of over 1335 Flutter Cupertino Icons.
- **Google Fonts** is a library of 1474 open source font families and APIs for convenient use via CSS and Android.
- **Cached network image** package allows you to use any widget as a placeholder.
- **Faker** is a library for Dart that generates fake data.
- **Jiffy** is a Dart date time package for parsing, manipulating, querying and formatting dates.
- **Logger** is a small, easy to use and extensible logger which prints beautiful logs.
- **Stream chat flutter core** is a package that provides UI components required for integrating Stream Chat into your application.
- **Firebase core** plugin is responsible for connecting your Flutter app to your Firebase project.
- **Firebase Auth** provides many methods and utilities for enabling you to integrate secure authentication into your new or existing Flutter application.

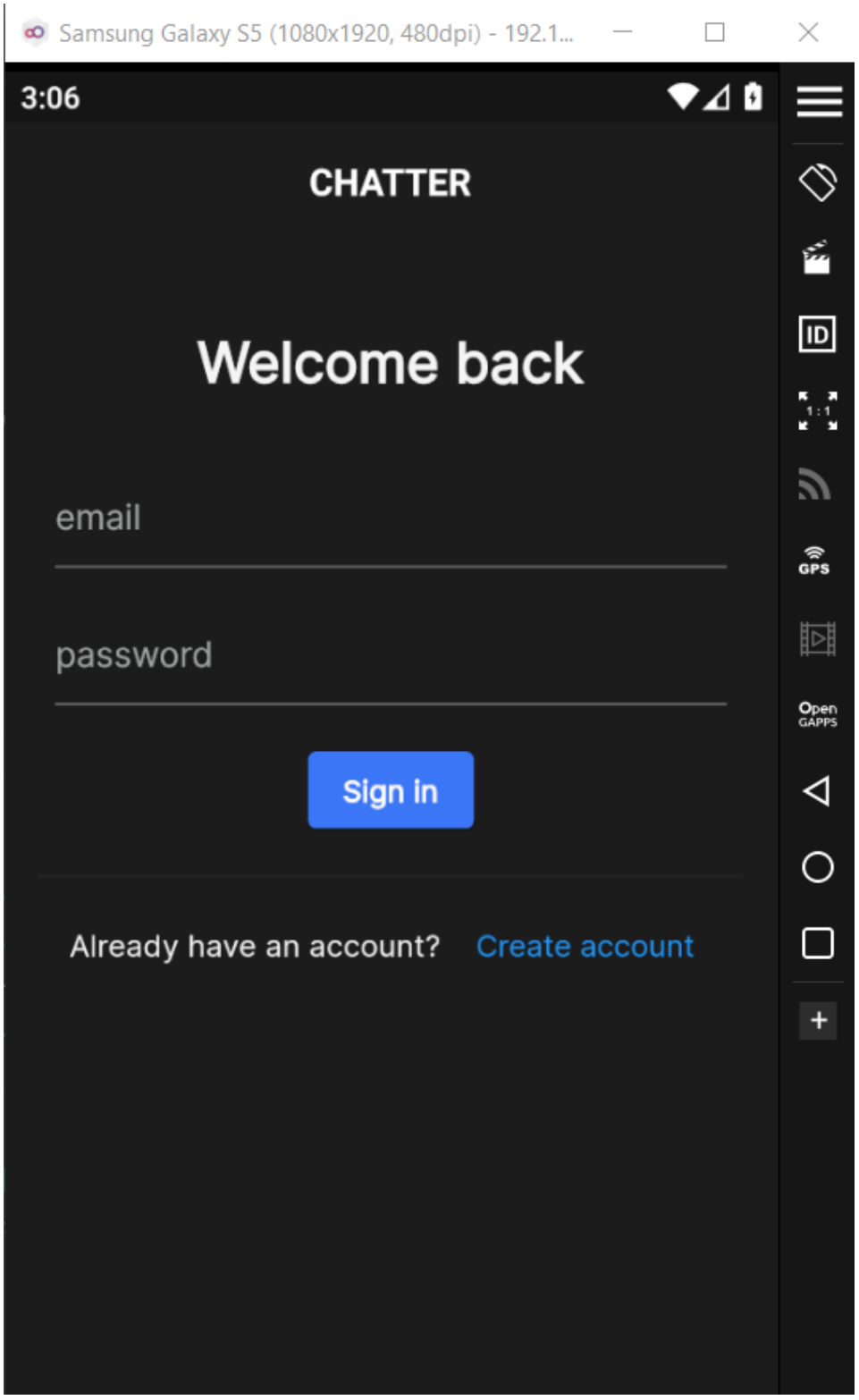
```yaml
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  google_fonts: ^3.0.1
  cached_network_image: ^3.2.1
  faker: ^2.0.0
  jiffy: ^5.0.0
  logger: ^1.1.0
  stream_chat_flutter_core: ^3.2.0
  firebase_core: ^2.4.0
  firebase_auth: ^4.2.0
  cloud_functions: ^4.0.6
```

# 3. Implementation

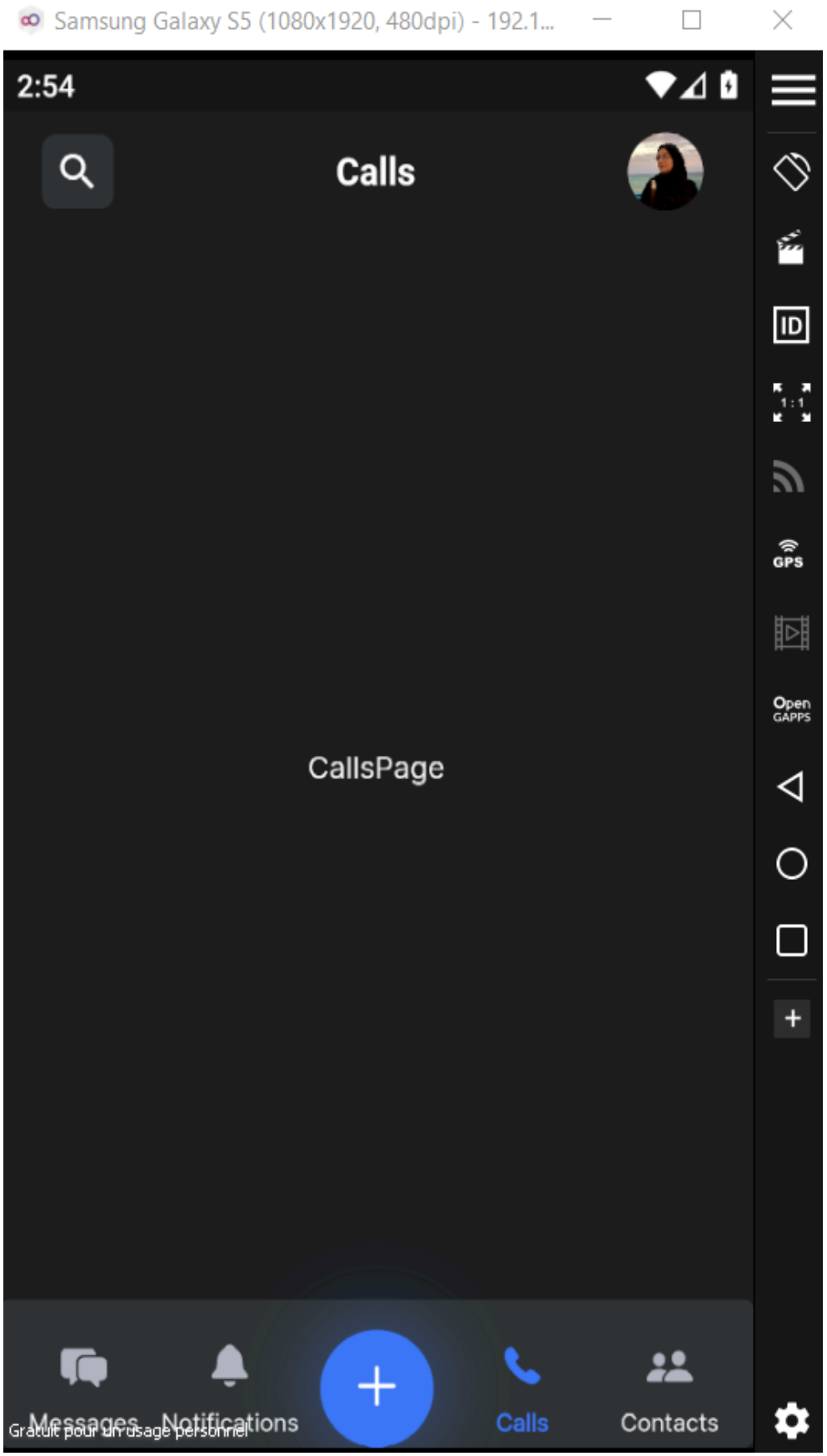
★ **Register**



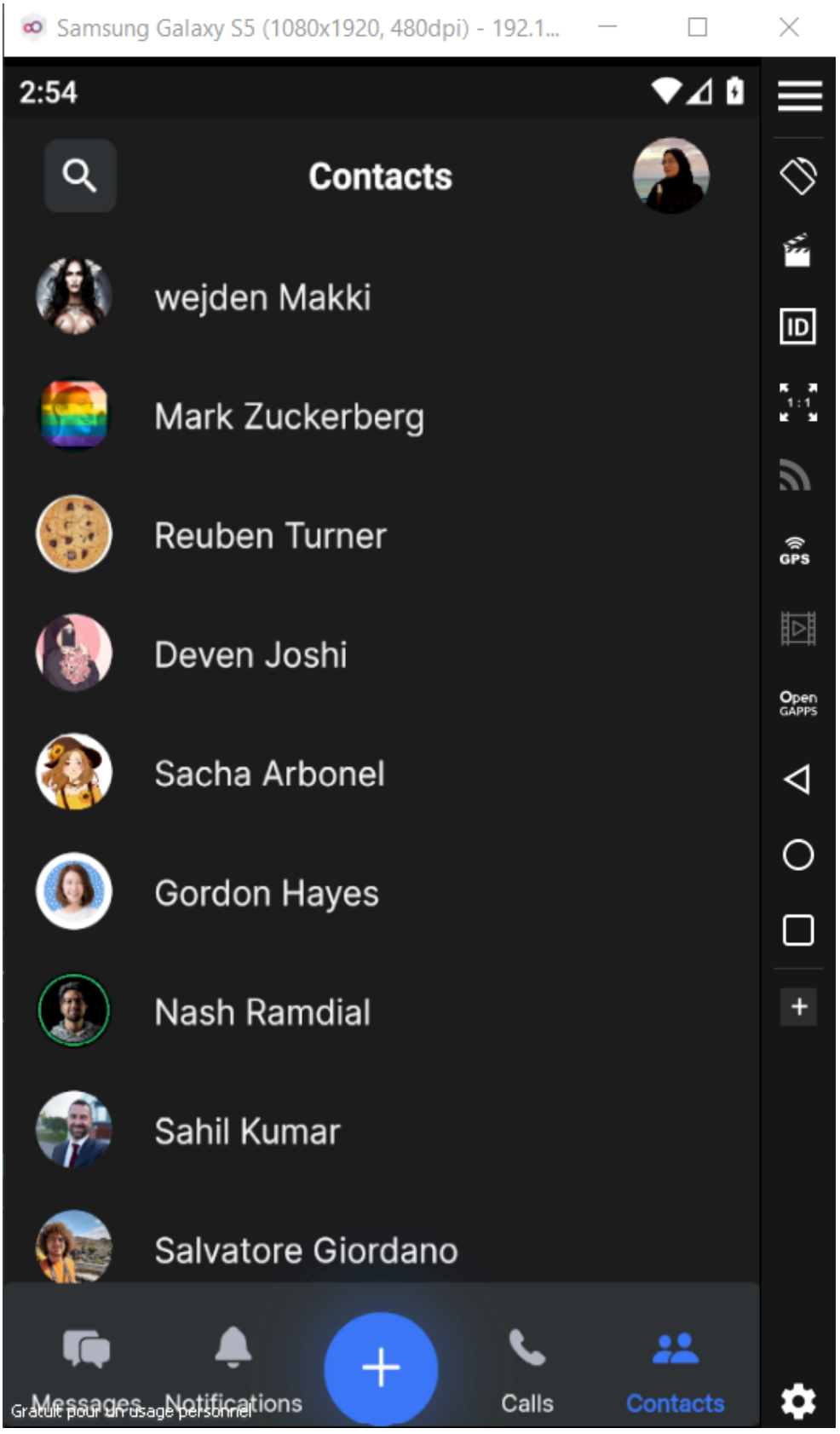
★ **Sign-In**


Samsung Galaxy S5 (1080x1920, 480dpi) - 192.1...
3:06
CHATTER
Welcome back
email
password
Sign in
Already have an account?
Create account

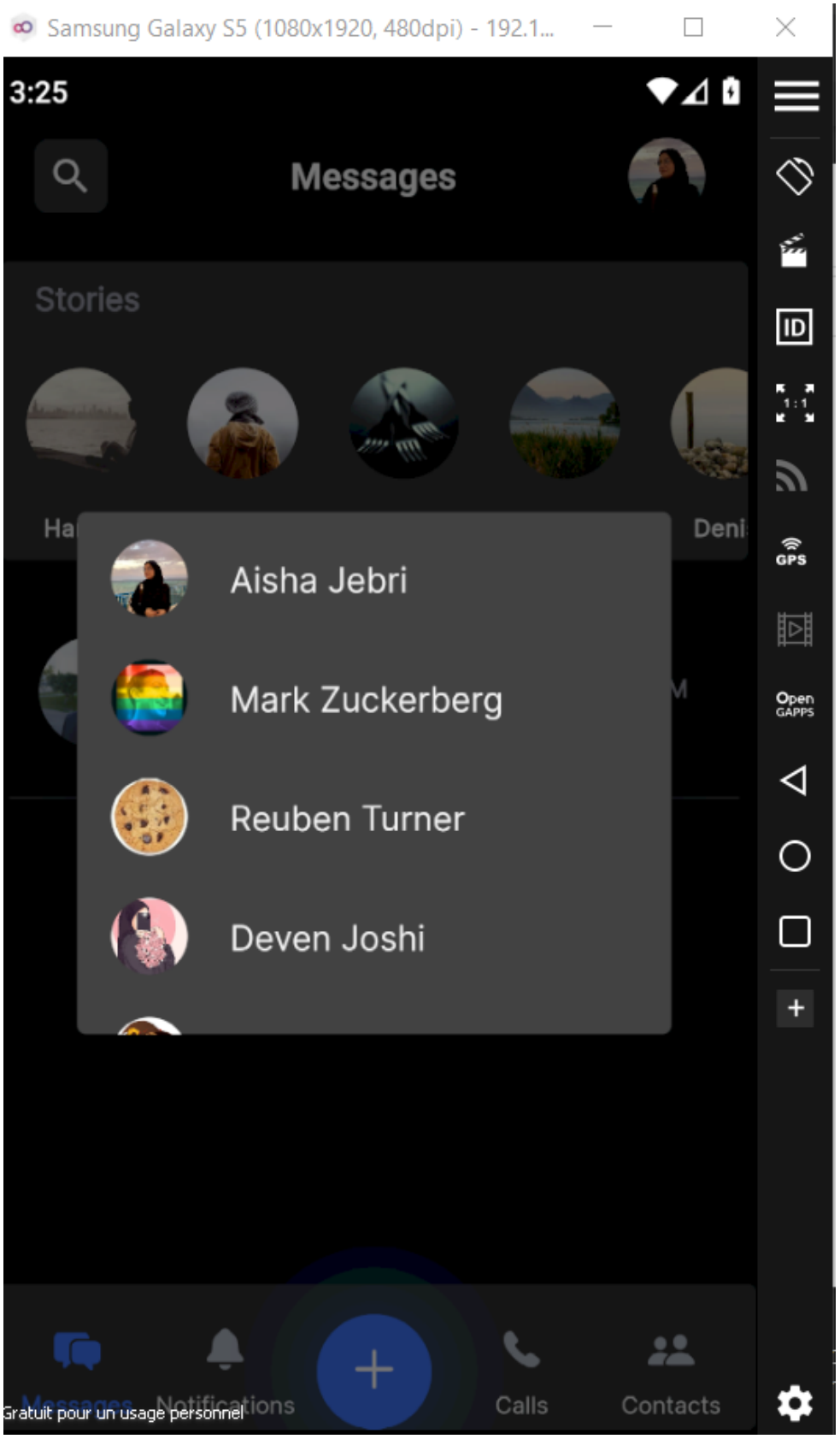
★ **Calls page**


Samsung Galaxy S5 (1080x1920, 480dpi) - 192.1...
2:54
Calls
CallsPage
Messages
Notifications
Calls
Contacts
Gratuit pour un usage personnel

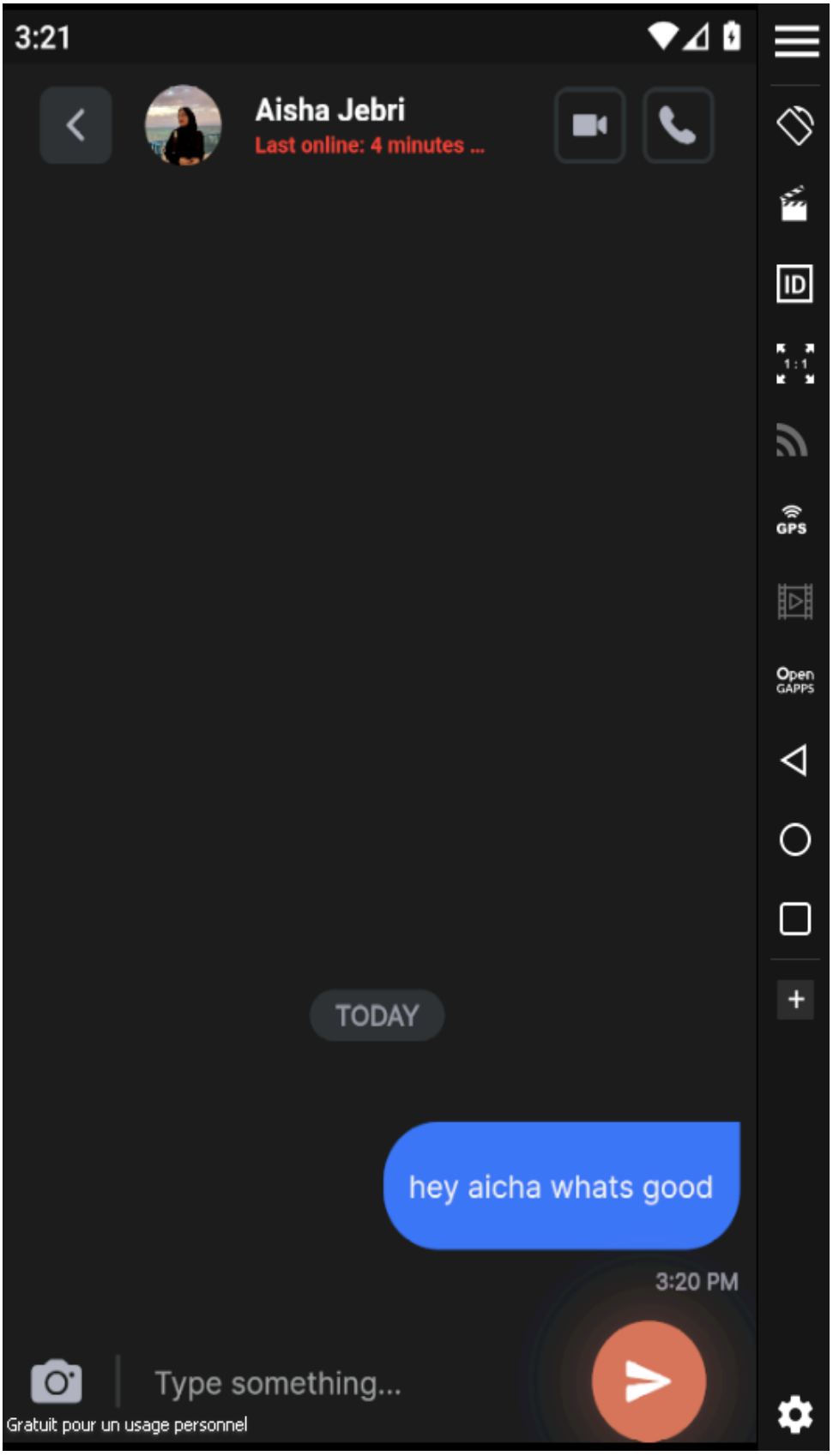
★ **Contacts**


Samsung Galaxy S5 (1080x1920, 480dpi) - 192.1...
2:54
Contacts
wejden Makki
Mark Zuckerberg
Reuben Turner
Deven Joshi
Sacha Arbonel
Gordon Hayes
Nash Ramdial
Sahil Kumar
Salvatore Giordano
Messages
Notifications
Calls
Contacts
Gratuit pour un usage personnel

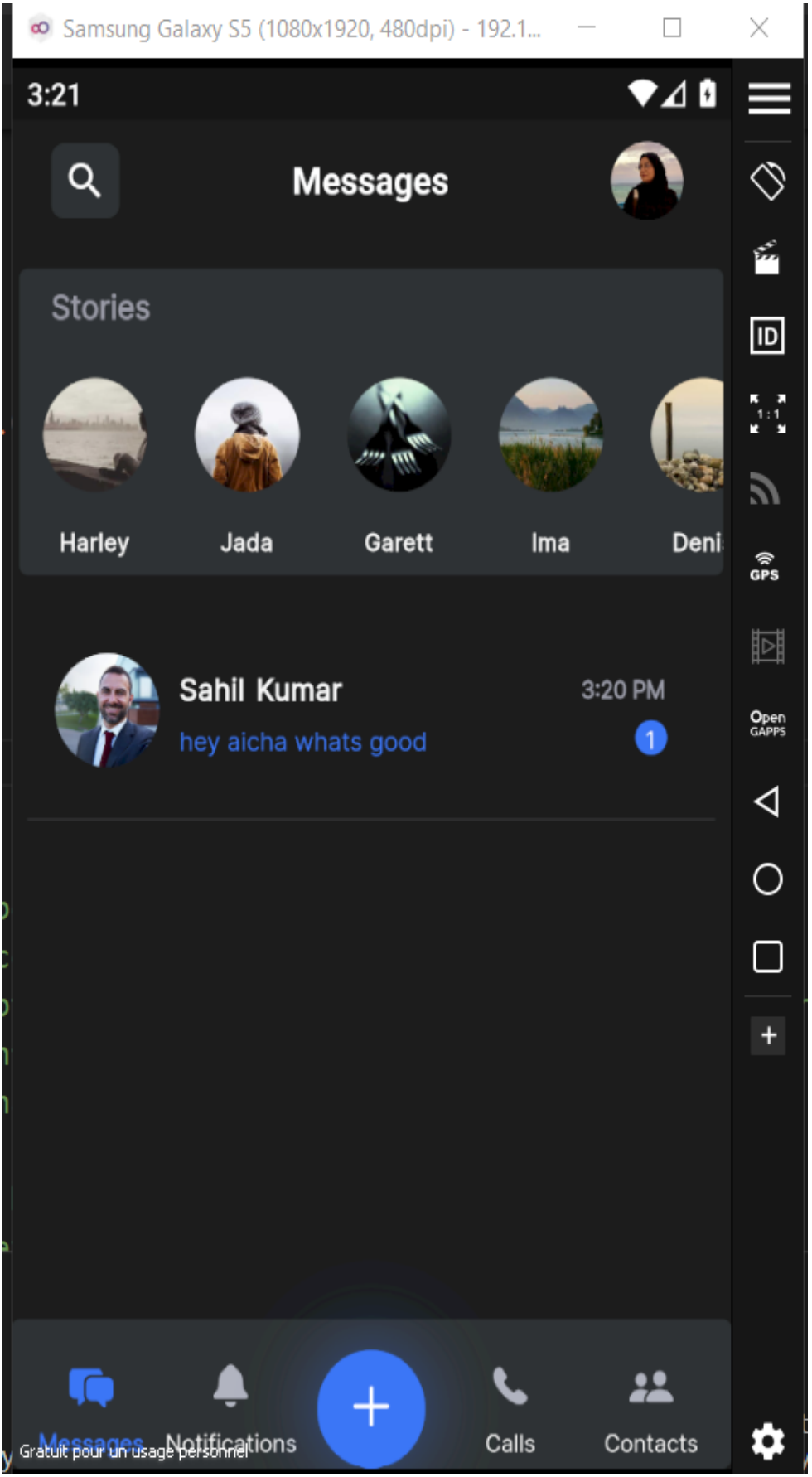
★ **Create a new conversation**



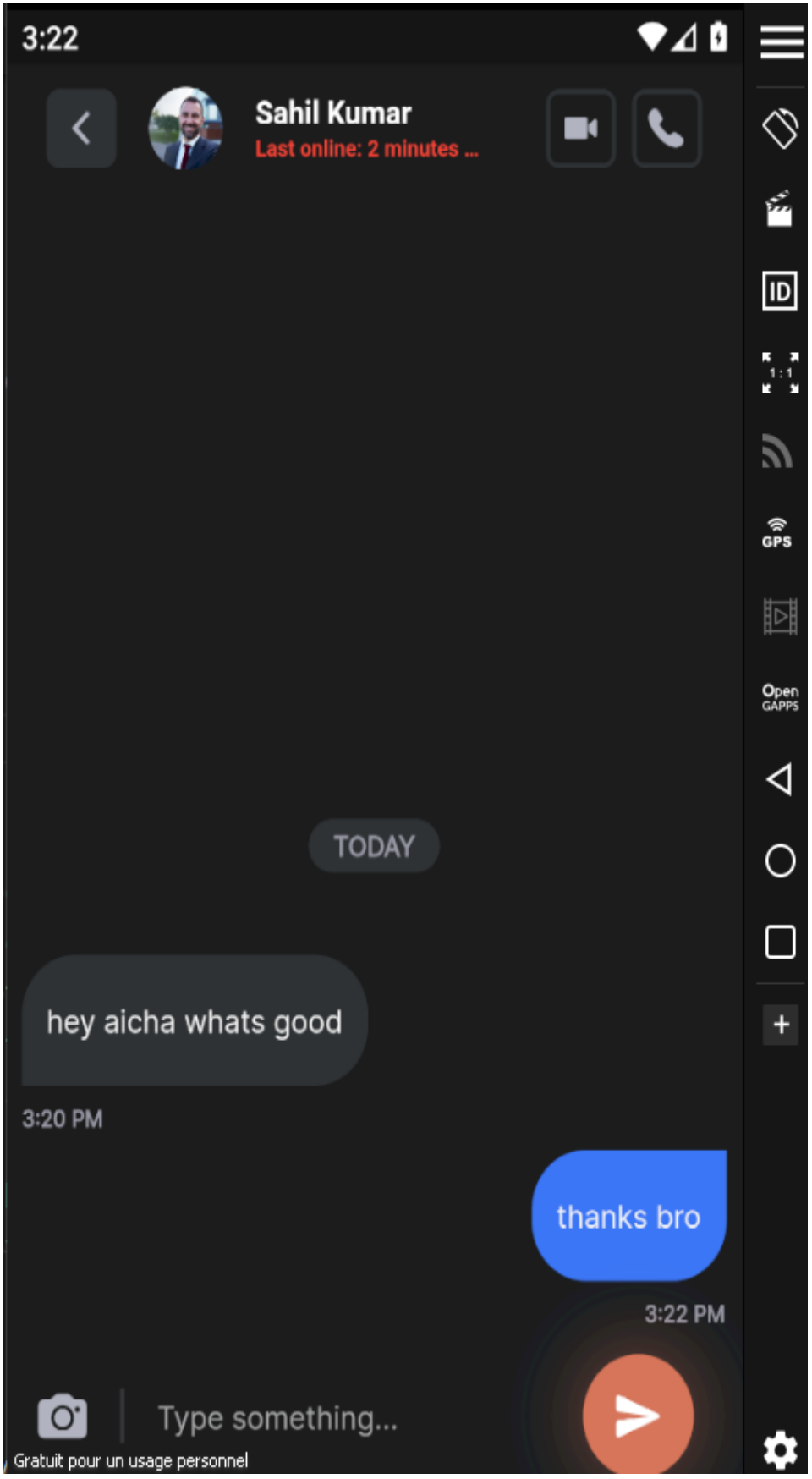
★ **Sending a message to user Aisha**



★ **Aisha’s perspective (receiving the text)**



★ **Aisha answering the text**



# How to run the project

Code source on : https://github.com/desktop69/ChatterFlutterApp.git
How to run the code :
Download the code source open it on VsCode and run the following commandes in the terminal :

1. flutter create .
2. flutter pub get
3. firebase init
4. dart pub global activate flutterfire_cli
5. flutterfire configure --project=/* your firebase project name */
6. flutter run

# Bibliography


- Stream api official documentation .
- Stream Developers official youtube channel .
- The official package repository for Dart and Flutter apps.
- Firebase console .