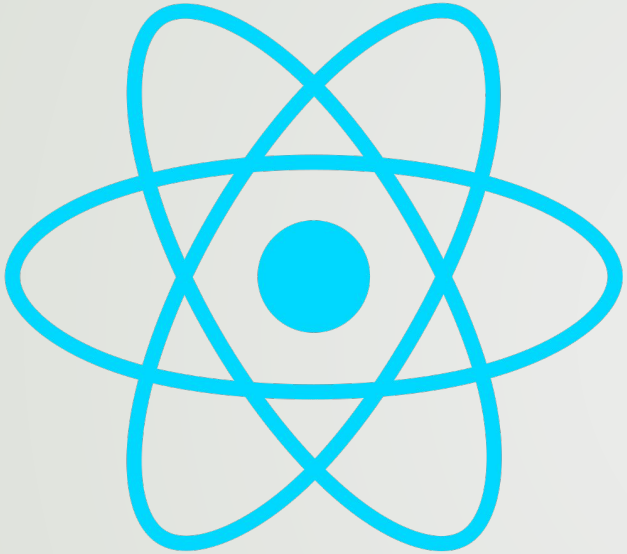
Chat app

Development process

September	2023

TABLE OF CONTENTS

TABLE OF CONTENTS



01

Setting up the development environment and **creating** the project structure, layout and styling

Skills used

Research
Problem solving
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

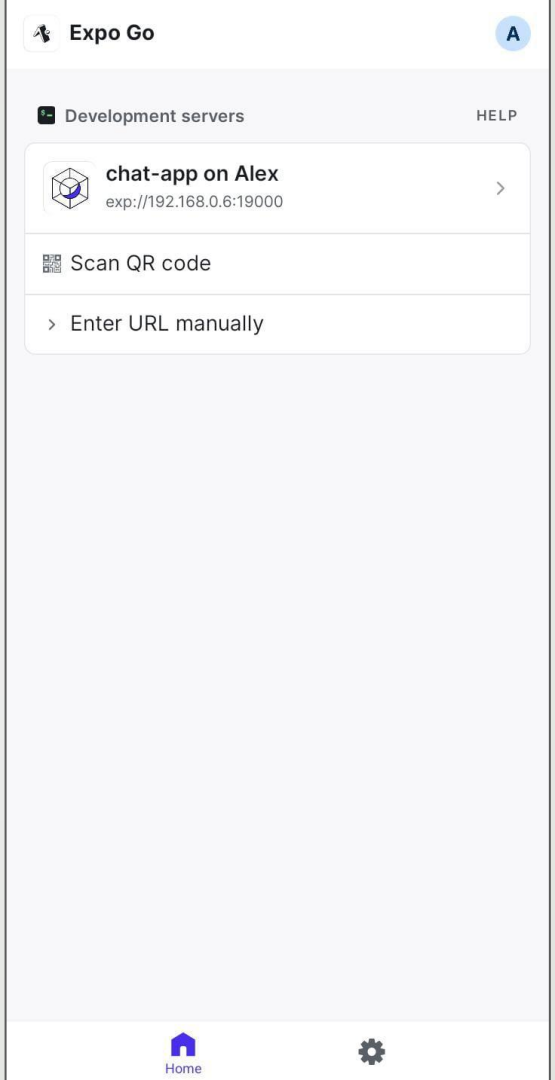
Organizing the development and the coding environment for efficient and well-oriented work from the beginning is one of the keys to ensure smooth workflow and limit avoidable time loss due to inefficient project initialization.

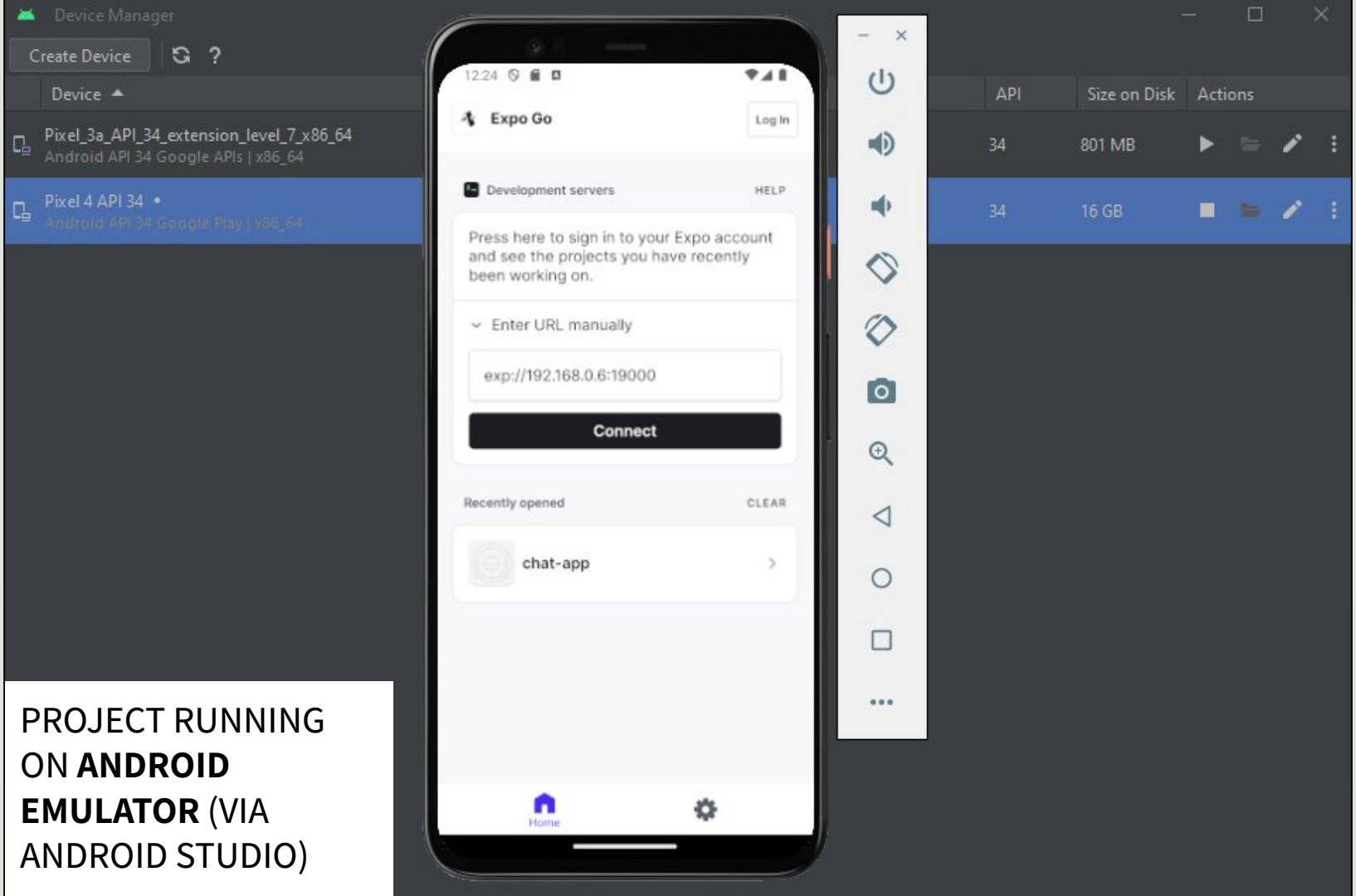
WHAT WAS THE GOAL

Setting up the development environment to ensure the project is properly initialized. More precisely, I:

- Updated my Node.js to a version supported by Expo;
- Set up Expo and Expo CLI on my machine, as this is the platform used to build the app;
- Set up Expo Go app on my phone, so that it’s possible to test the app on a physical mobile device;
- Set up Android Emulator (Android's virtual device simulation tool) on my machine using Android Studio to see how the app looks and behaves on different devices.

PROJECT RUNNING ON **EXPO GO** ON A PHYSICAL ANDROID DEVICE





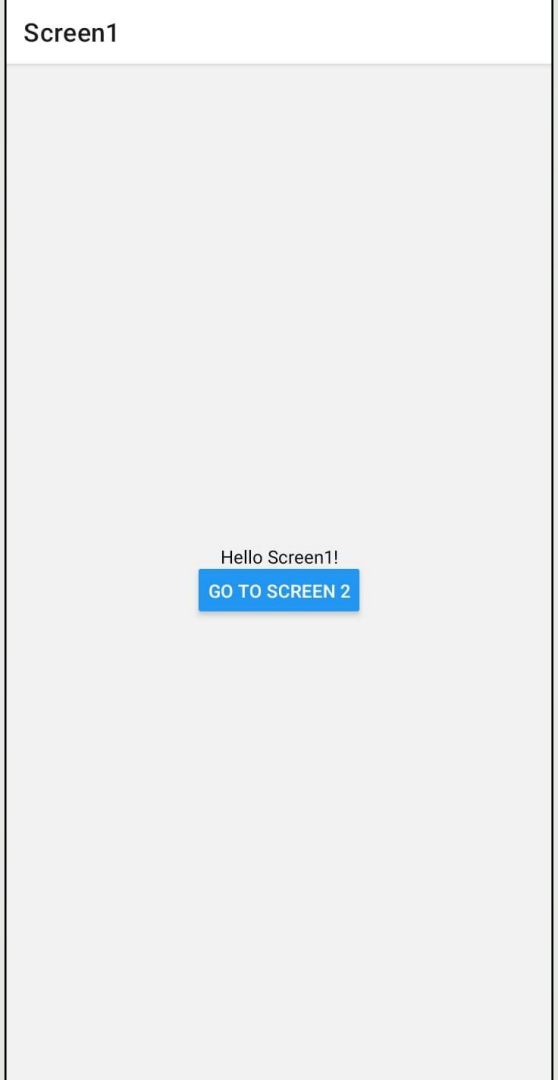
PROJECT RUNNING ON **ANDROID EMULATOR** (VIA ANDROID STUDIO)

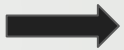
WHAT WAS THE GOAL (SUITE)

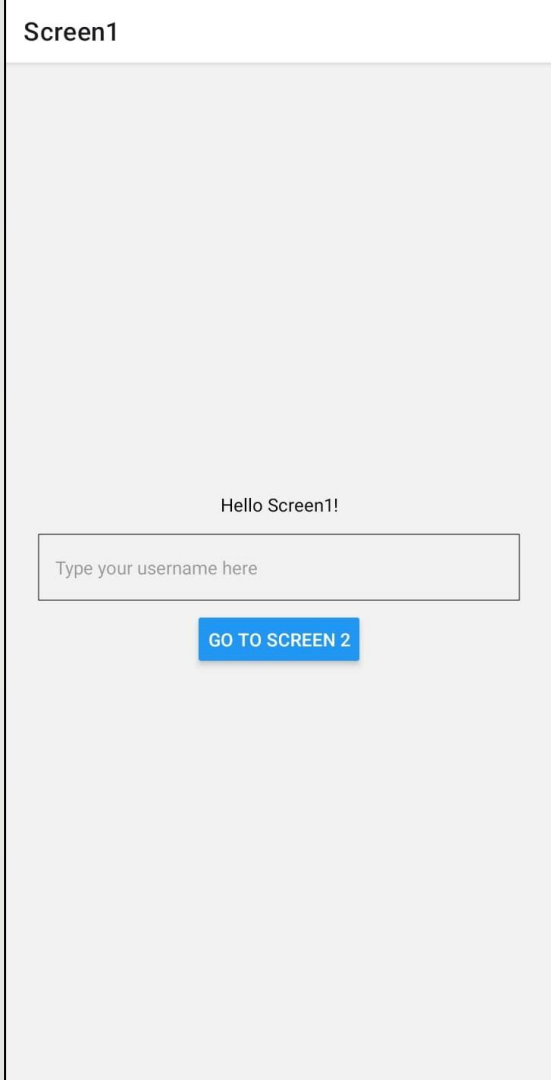
Implementing navigation logic between the initial screen and the chat screen (see next images).

Creating the app layout / styling for the initial screen and testing it in my own Android device as well as with Android Emulator from Android Studio (see next images).

CHAT APP **INITIAL STARTING SCREEN** DEVELOPMENT

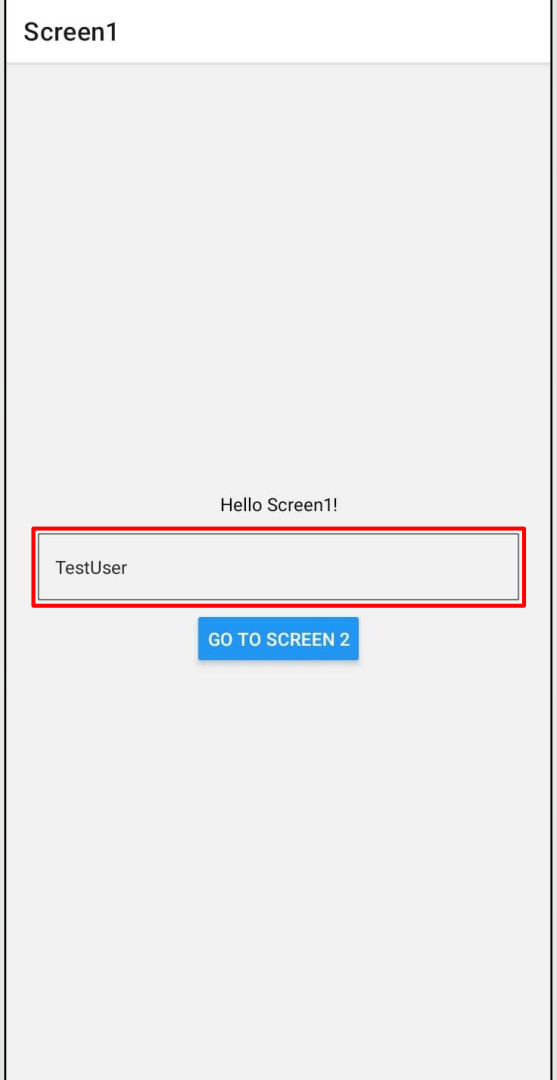


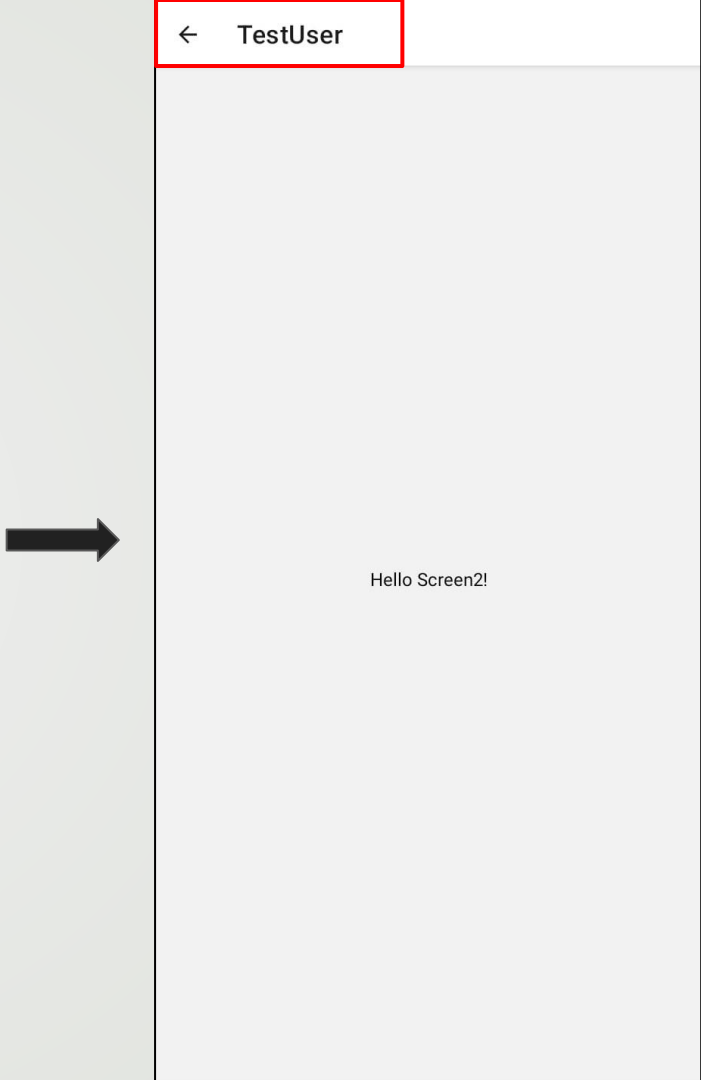




CHAT APP **INITIAL STARTING SCREEN** DEVELOPMENT

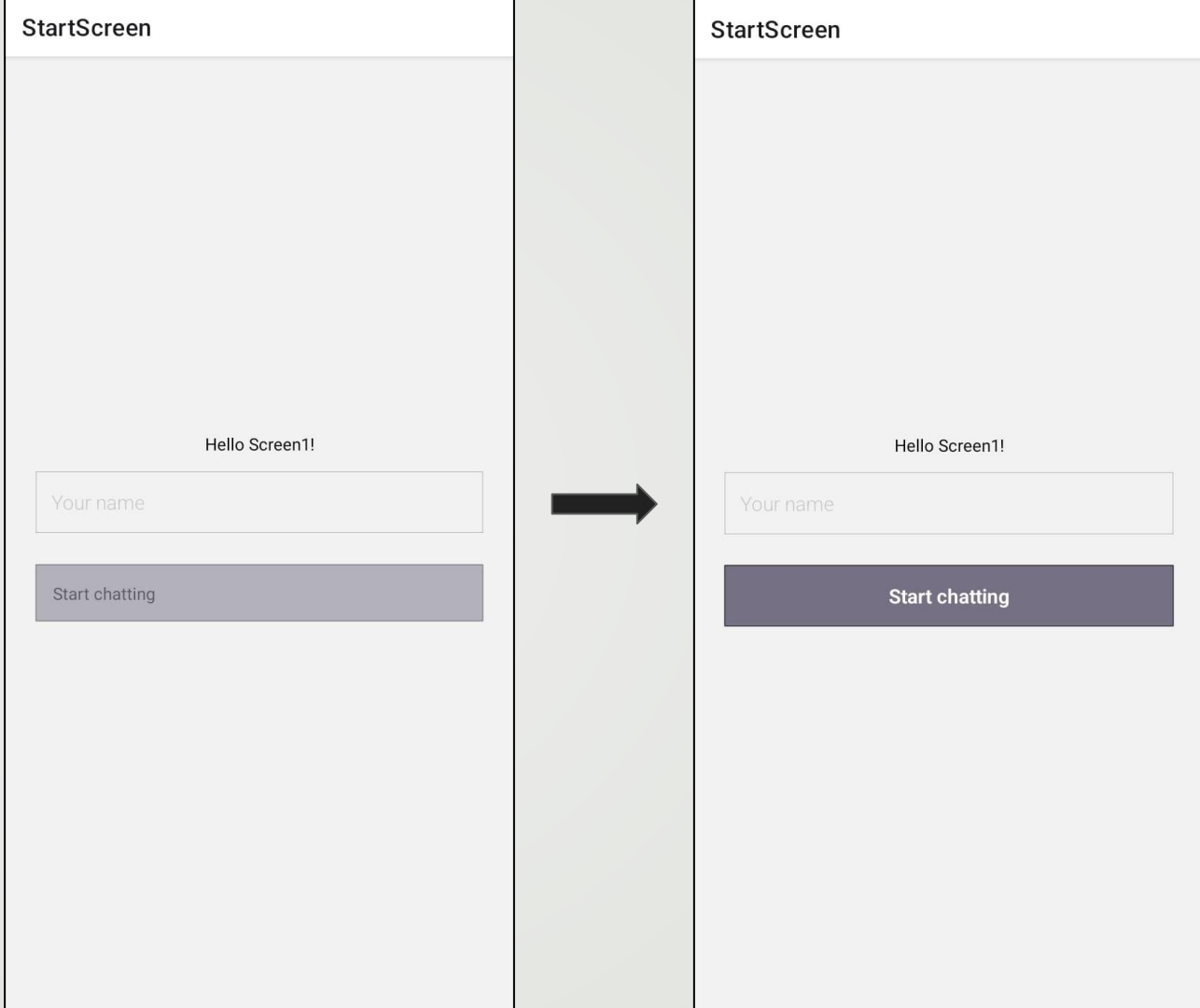
*NAVIGATE FROM INITIAL STARTING SCREEN TO THE CHAT SCREEN WHEN THE BLUE BUTTON ON STARTING SCREEN IS PRESSED





CHAT APP **INITIAL STARTING SCREEN** DEVELOPMENT

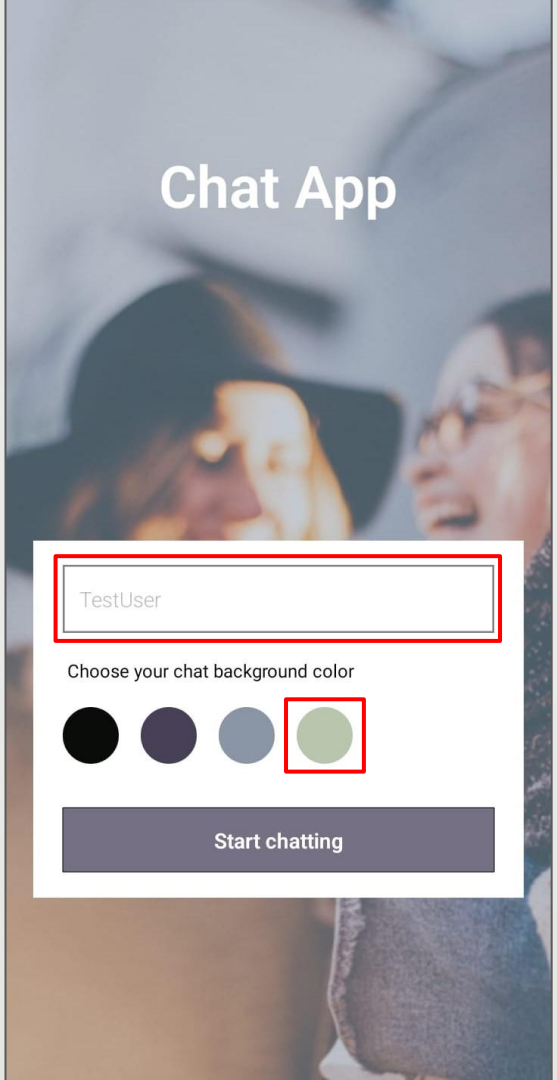
*STYLING INITIAL STARTING SCREEN ELEMENTS

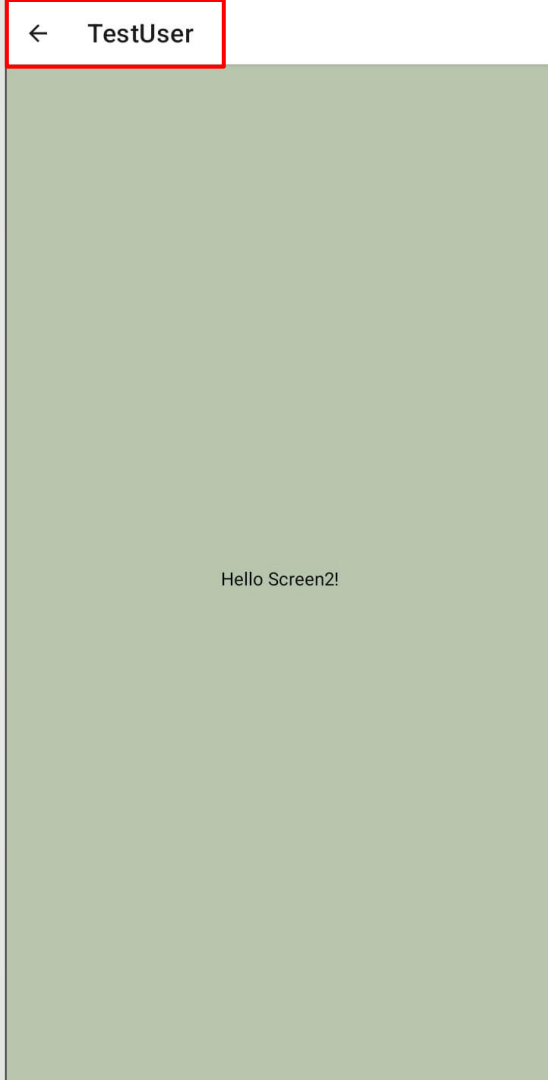


CHAT APP **INITIAL STARTING SCREEN** FINAL LAYOUT

*ALLOW USERS TO ENTER THEIR NAME AND CHOOSE A BACKGROUND COLOR FOR THEIR CHAT SCREEN

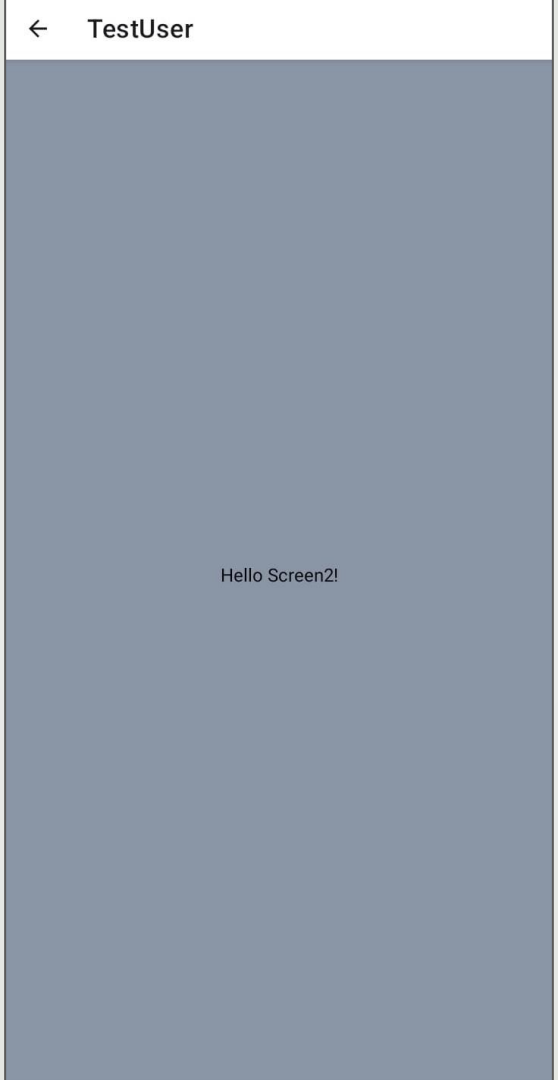
*SECOND IMAGE IS THE CHAT SCREEN, AFTER USERS PRESSED ‘START CHATTING’ ON INITIAL STARTING SCREEN

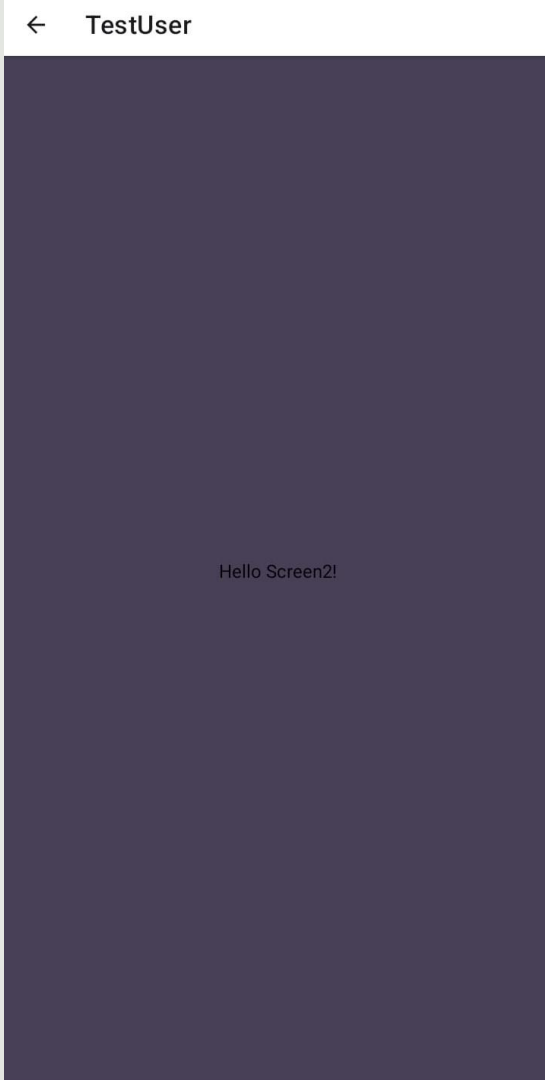




NAVIGATION TO CHAT SCREEN AFTER USERS PRESSED ‘START CHATTING’ ON INITIAL STARTING SCREEN

*DIFFERENT BACKGROUND COLORS (MESSAGE COLORS HAVE BEEN ADJUSTED LATER TO ENHANCE VISIBILITY)





02

Developing the chat screen using Gifted Chat library

Skills used

Research
Problem solving
Code writing

WHY WAS THIS STEP IMPORTANT

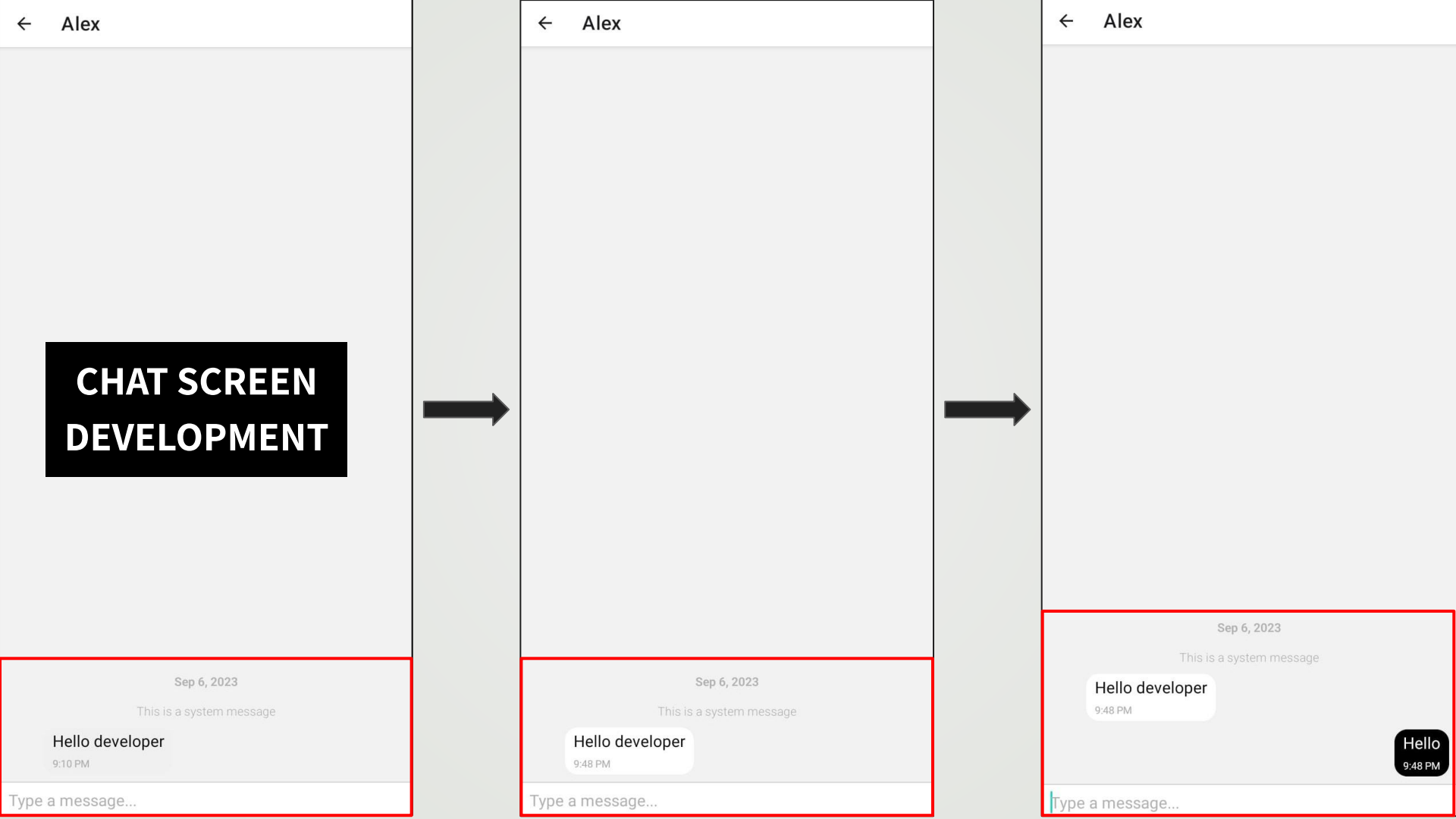
Until this step, the chat screen had no functionalities to send messages. It was therefore important to implement the logic in the chat UI so that users, once on this screen, can send and receive messages in a design that is familiar to them.

WHAT WAS THE GOAL

Developing the chat screen by using Gifted Chat library to ensure users can send and receive messages (implementation of a text input field for typing messages, speech bubbles for sent and received messages, automatic system messages, etc.).

Styling some aspects of the chat screen (ex: message bubbles) to improve the visual.

Testing everything is working as expected, using both my own Android device and an Android Emulator.


Alex
CHAT SCREEN DEVELOPMENT
Sep 6, 2023
This is a system message
Hello developer
9:10 PM
Type a message...
Alex
Sep 6, 2023
This is a system message
Hello developer
9:48 PM
Type a message...
Alex
Sep 6, 2023
This is a system message
Hello developer
9:48 PM
Hello
9:48 PM
Type a message...

03

Connecting the app to a Cloud database

Skills used

Research
Problem solving
Coding
Debugging

WHY WAS THIS STEP IMPORTANT

This step was important to ensure that the messages sent by users would be stored somewhere and be accessible in real-time. This is needed for a chat app since users expect the messages sent and received to be shown on their screen right away.

WHAT WAS THE GOAL

Setting up a Firestore database (NoSQL) to save / store the messages sent on Google Cloud.

- Setting up the Firestore database was done on the official Firebase website.
- The connection between the Firestore database and the application was carried on in the app code files, after having installed Firebase as one of the project dependencies and imported all the necessary elements in the app's files.

Implementing codes in the app files so that each message sent is being saved automatically in the Firestore database, and rendered in real-time in users chat UI.

WHAT WAS THE GOAL (SUITE)

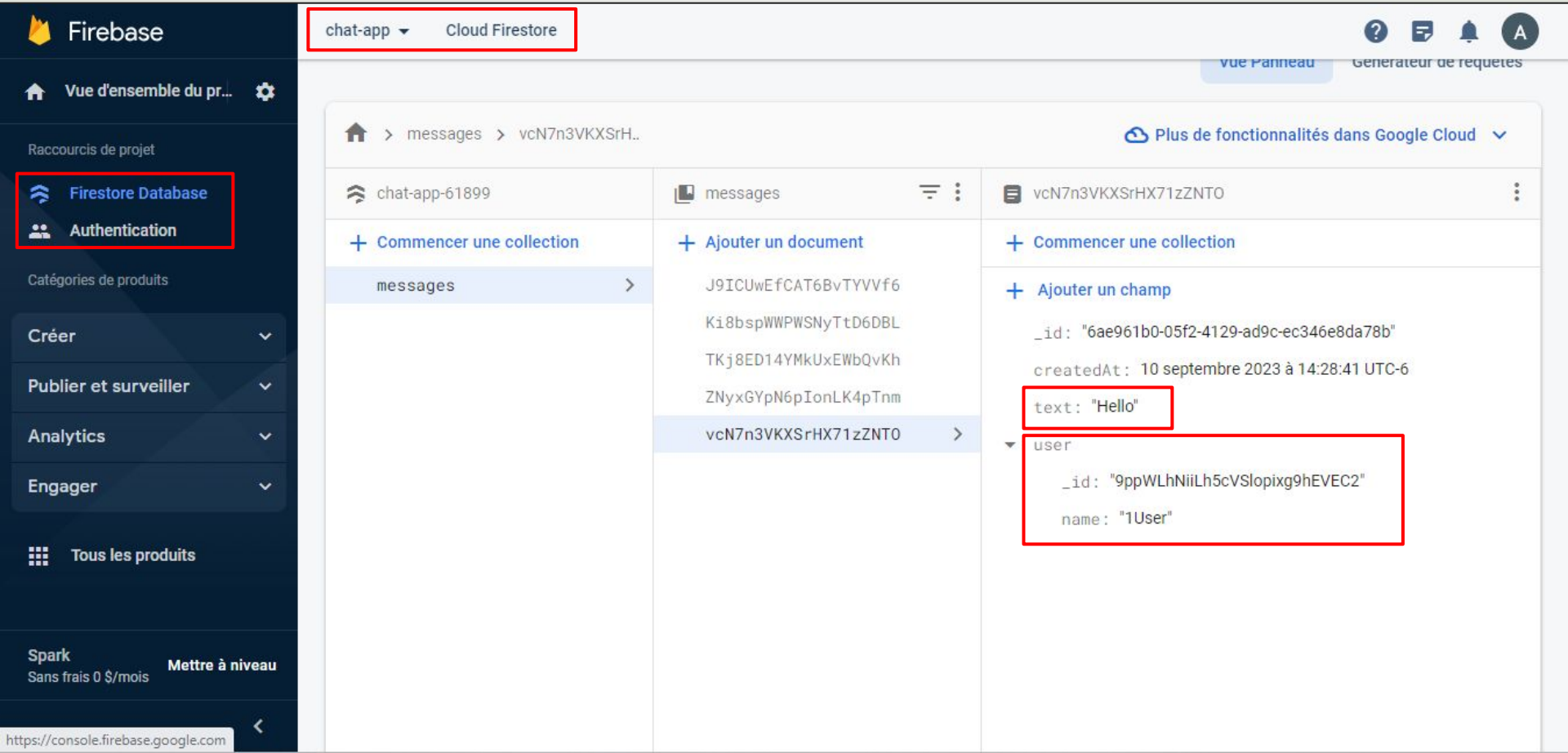
Implementing an anonymous authentication process for users using Firebase Authentication feature to ensure that each user only has access to their messages - and that those messages are not accessible to other users (in other words, ensuring the user experience is personalized).

- Anonymous authentication doesn’t require users to verify their identity by way of an email address (or otherwise).
- Instead, when users sign in in the app for the first time, they are assigned a unique user ID, and this unique user ID is stored both in Firestore database and locally on their device. Everytime they go back on the app, this user ID is retrieved and users get connected with this.

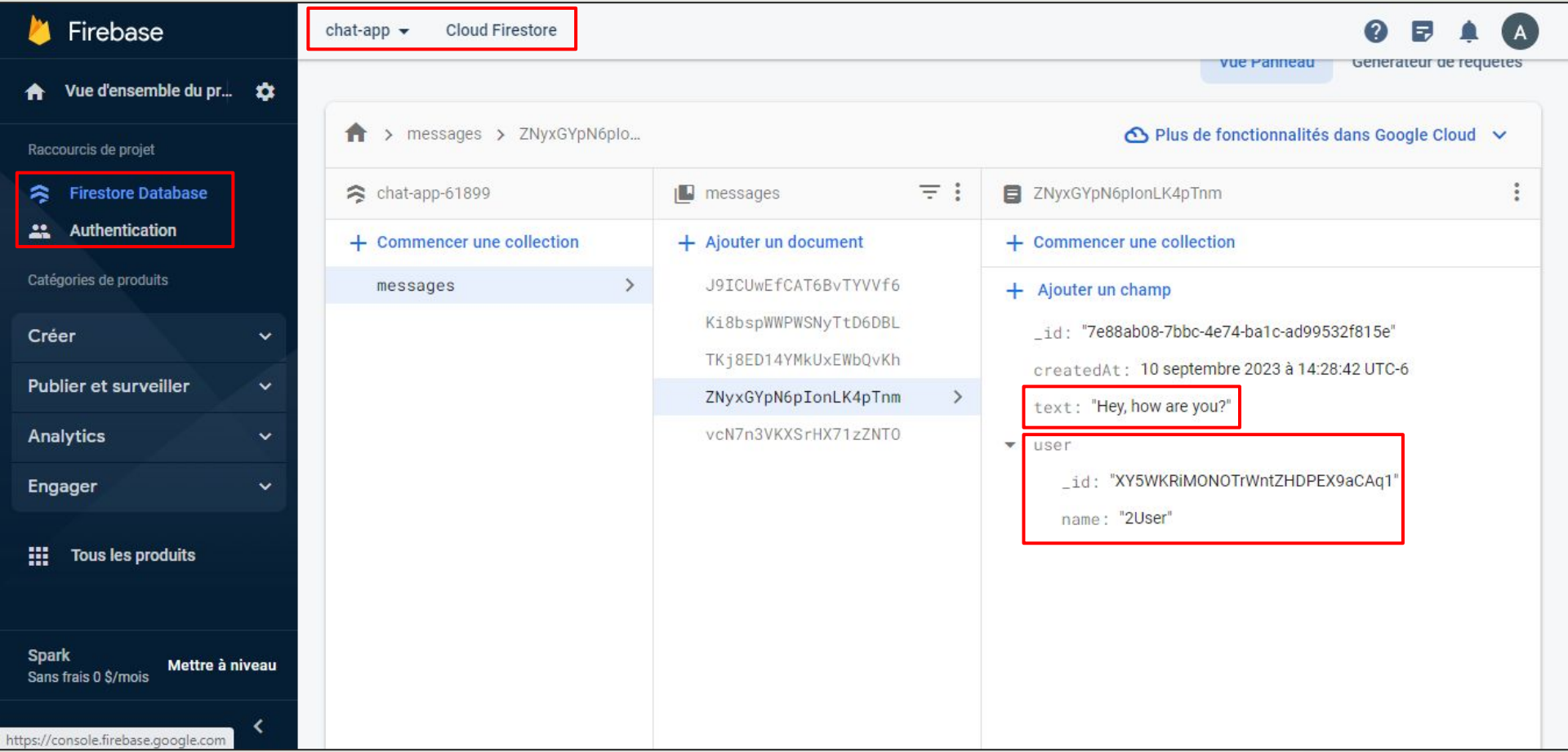
Testing the app to ensure everything works as expected, using both Expo Go on my physical Android device and an Android Emulator.

SAMPLE OF MESSAGES SAVED / STORED **IN THE FIRESTORE DATABASE** AFTER BEING SENT ON CHAT APP BY 1USER AND 2USER

*THANKS TO ANONYMOUS AUTHENTICATION METHOD IMPLEMENTED, EACH USER HAVE THEIR OWN ID



*THANKS TO ANONYMOUS AUTHENTICATION METHOD IMPLEMENTED, EACH USER HAVE THEIR OWN ID



CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

At first, I tried to initialize my Firestore database in my code by importing and using the *getFirestore* function from firebase/firestore. However, this caused incompatibility/network issues. After many failed attempts to resolve this, I looked for alternatives on Github, and eventually found out that many other developers faced this same problem. Following discussions on the subject, I eventually spot a solution and implement it in my code in replacement of the *getFirestore*, which resolves the issue.

04

AsyncStorage

Implementing offline logic and client-side data storage

Skills used

Research
Problem solving
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

Users don’t always have an internet connection, so it’s generally very useful to implement some offline logic to make some features available even when there’s no network. This can greatly enhance user experience, especially in a chat type of app.

WHAT WAS THE GOAL

Implementing client-side storage in chat app to allow users to read their messages even when offline. To do so, the package *@react-native-async-storage/async-storage* (*AsyncStorage*) have been used and implemented in the app.

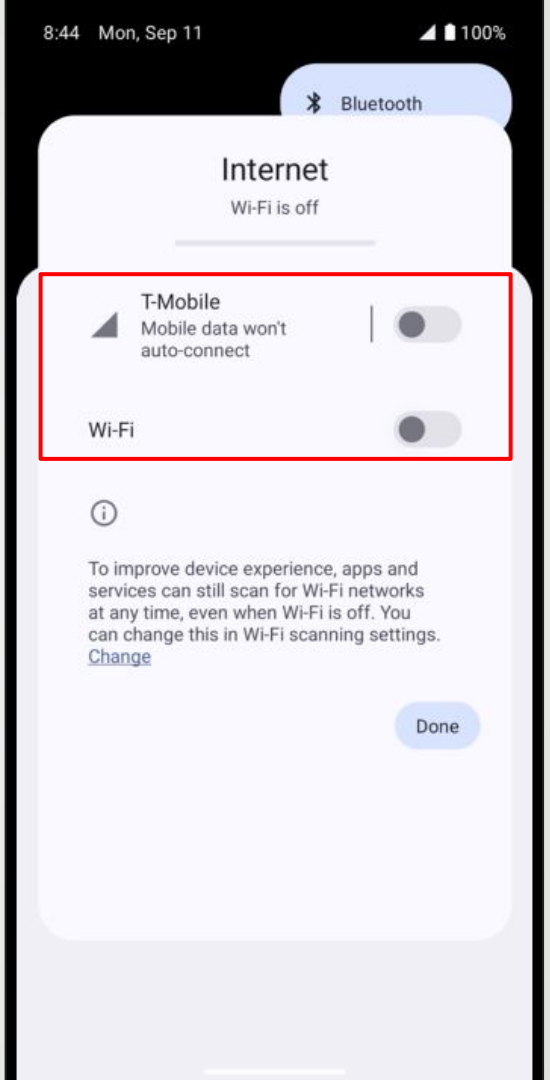
- To determine whether a user is online or not, NetInfo have been used (recommended package by the Expo team)
- If there's an internet connection, the data is fetched from the Firestore Database and if there's no internet connection, the data is fetched locally from AsyncStorage. A code has also been written to ensure that the local storage cache is always kept up to date on new messages when there's an internet connection.
- Also, since users can't send messages when they are offline and to make it clear, the text input field is hidden automatically when there's no internet (this also gives more space to display the messages). When the internet connection is back, the text input field is shown again automatically to users write and send new messages.

Testing the app to ensure everything works as expected, using both Expo Go on my physical Android device and an Android Emulator.

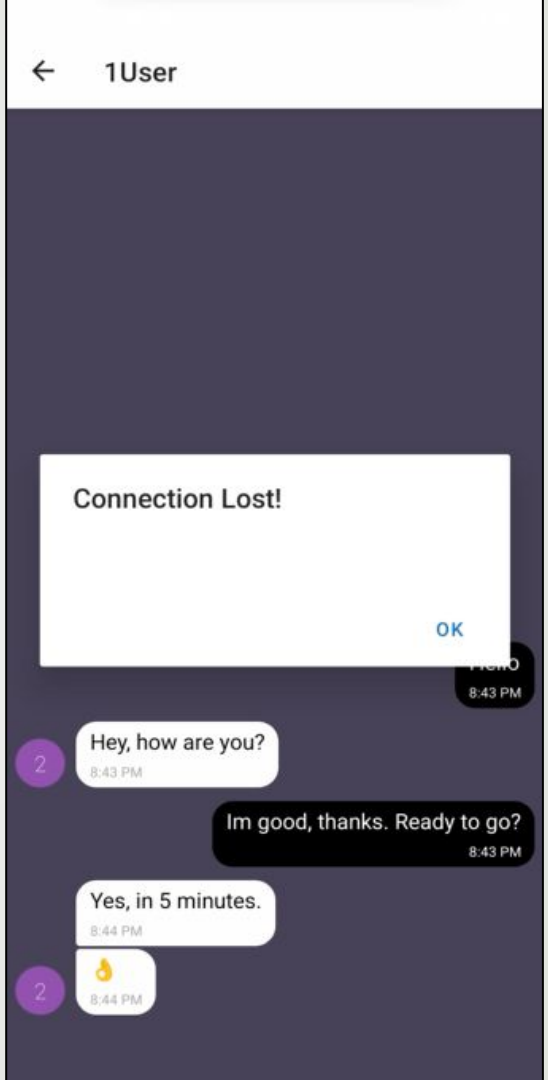
APP RUNNING **WITH INTERNET**

APP RUNNING WITH INTERNET - **USERS CAN SEND MESSAGES**

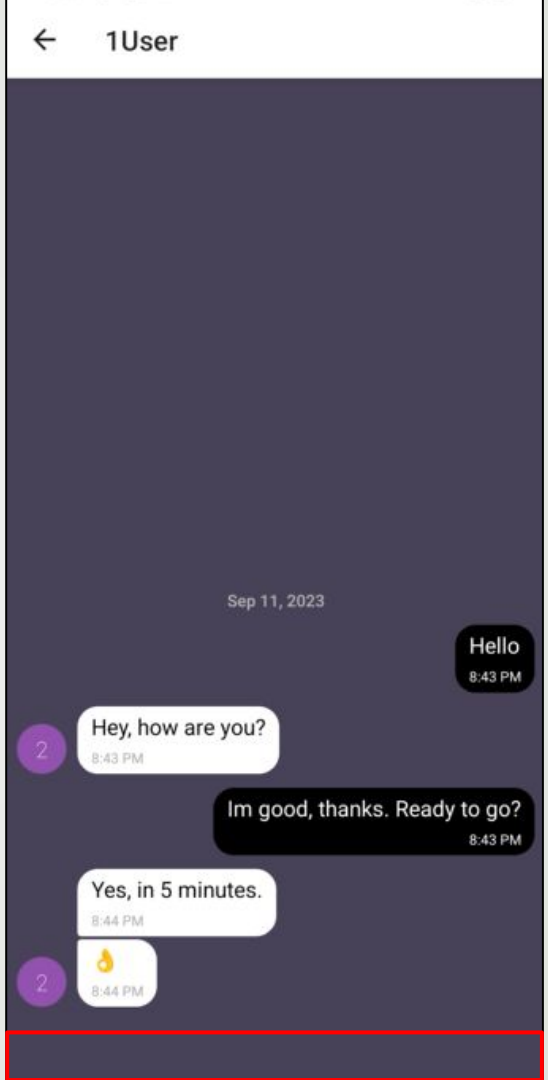
APP RUNNING **WITHOUT INTERNET**



ALERT LETTING USERS KNOW THEY ARE OFFLINE



APP RUNNING WITHOUT INTERNET - USERS CANNOT SEND MESSAGES (TEXT INPUT FIELD HIDDEN) BUT CAN STILL READ MESSAGES SENT BEFORE



05

Implementing communication features and **making the app accessible** for screen readers

Skills used

Research
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

Creating an application that allows users to send messages to each other is great, but for a more complete experience and to ensure to meet the expectations that users generally have for this type of application, it was important to add additional features, such as the ability to send images, take photos, share location and send audios.

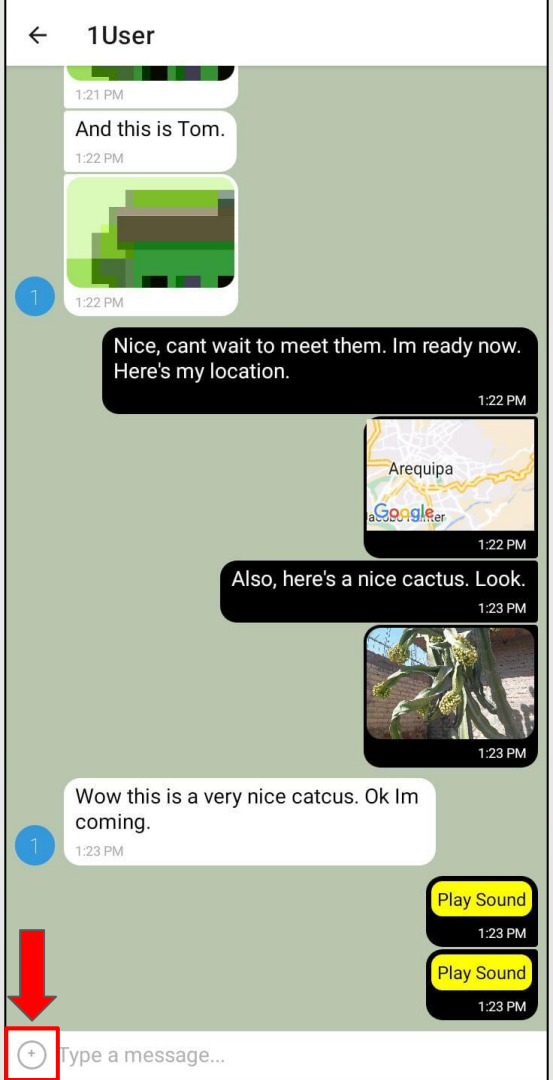
WHAT WAS THE GOAL

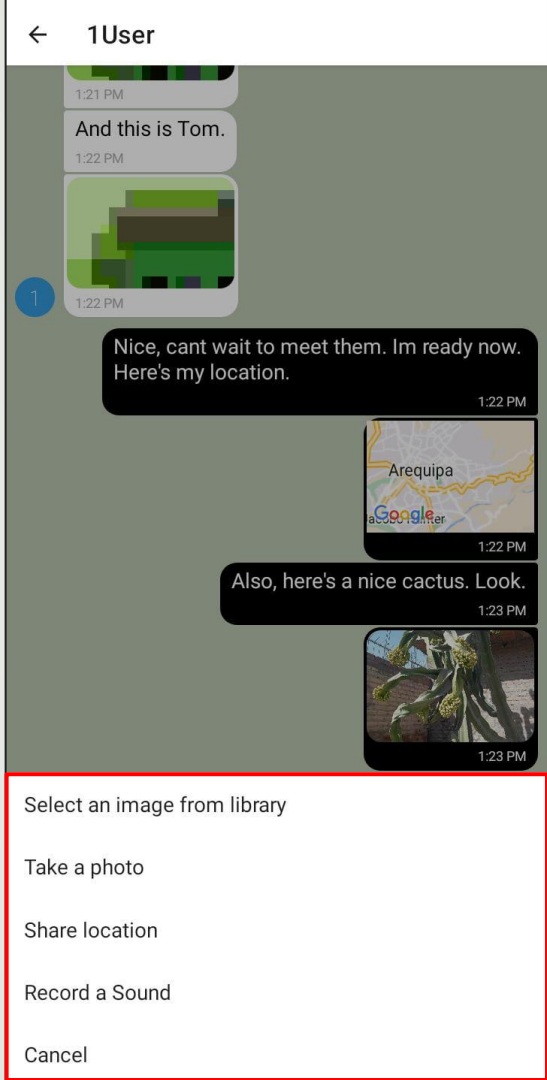
Adding additional features in the app to allow users to (1) send images from their phone image library, (2) take pictures with their phone camera and send it, (3) share their location and (4) send audio vocals.

- To allow users to send images from their library or take photos, Expo's *ImagePicker API* has been used.
- To allow users to send their location, the Expo packages *expo-location* and *react-native-maps* have been used.
- Finally, to allow users to send and receive audios, Expo's Audio API (expo-av) packages have been used.

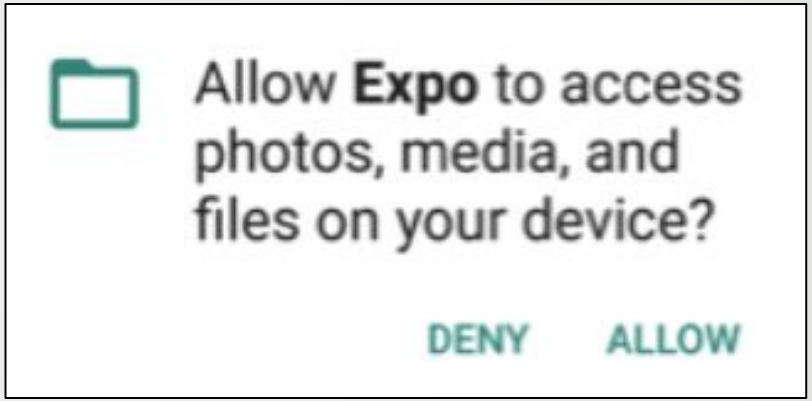
*Codes have also been added to ensure **users are asked for permission before accessing their library, camera, location and microphone**.

USERS CAN CLICK ON THE **(+)** **BUTTON** IN THE TEXT INPUT FIELD **TO DISPLAY MORE ACTIONS**





PERMISSION ASKED FIRST WHEN USERS CLICK ON ANY ACTION



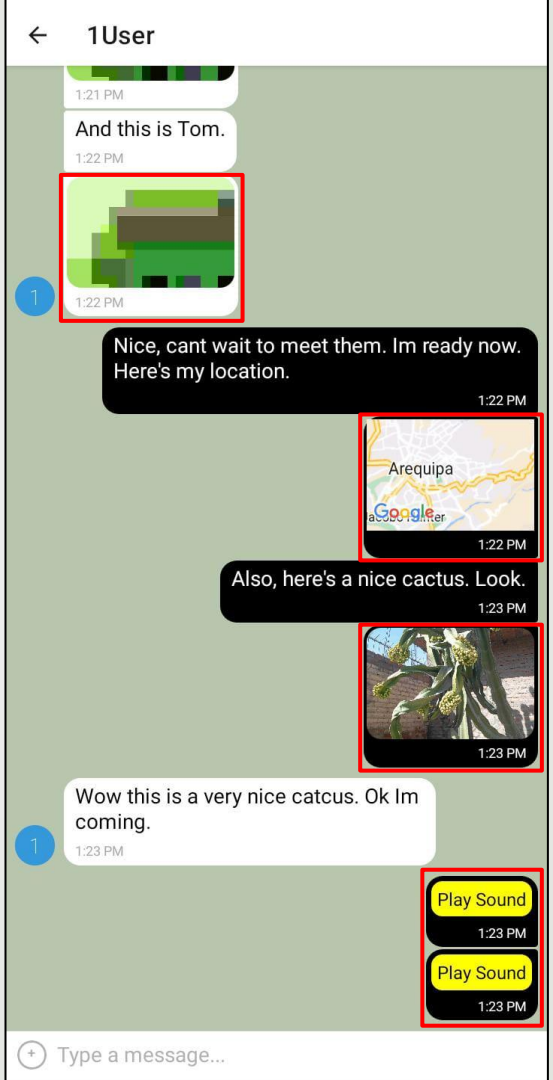
FROM TOP TO BOTTOM:

-IMAGE RECEIVED FROM ANOTHER USER PHONE'S LIBRARY

-SHARED LOCATION

-IMAGE SENT AFTER BEING TAKEN FROM USER PHONE'S CAMERA

-AUDIO VOCALS SENT

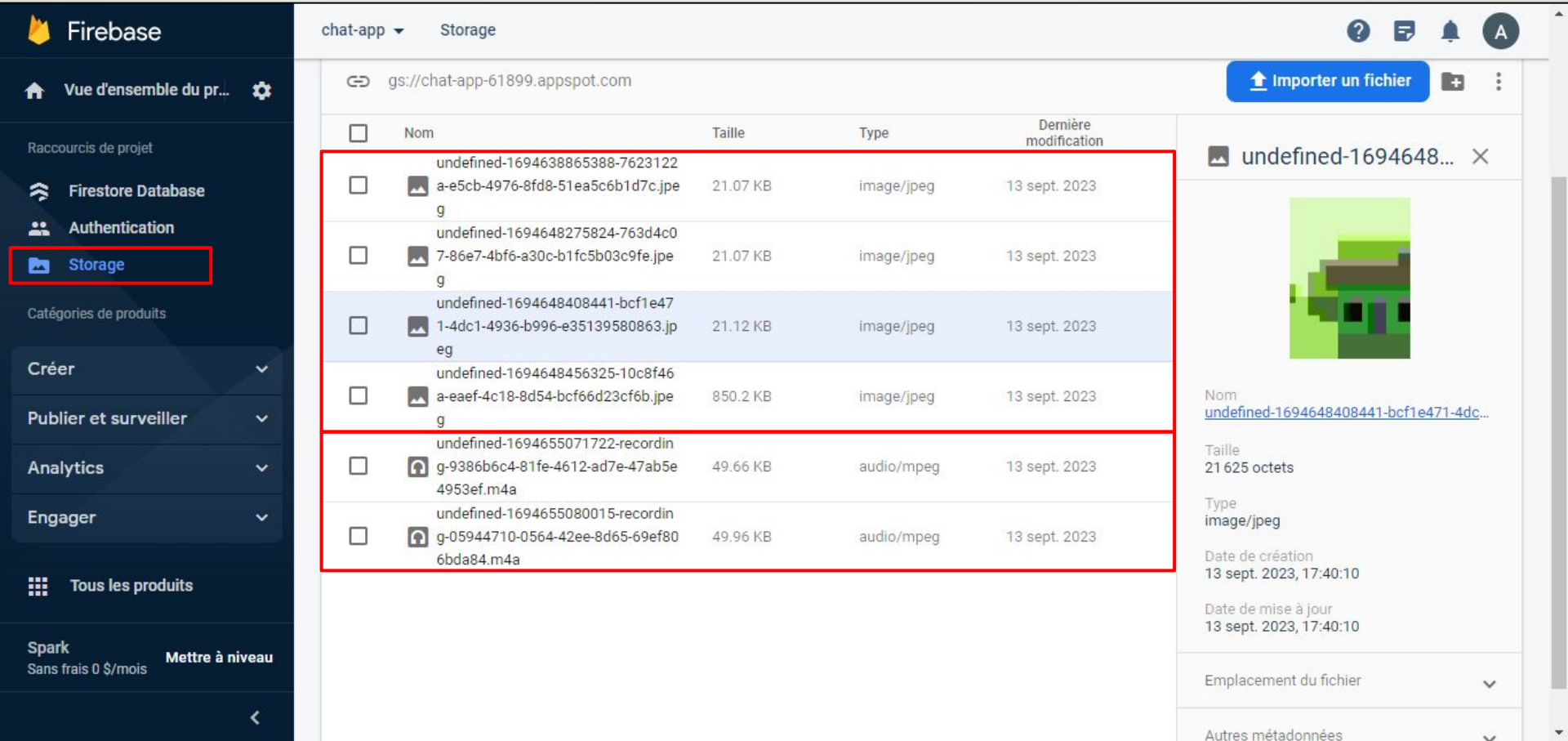


WHAT WAS THE GOAL (SUITE)

Setting up a Google Cloud Storage space on Firebase to store images and audios sent via chat app. Storing images and audios on a remote server allows all mobile devices to connect to the server to fetch the sent images and audios, and so access them.

Adjusting the action (+) button (the one user can click on to display more features such as sending pictures from phone's library or share location) to make it more accessible so that users with screen readers will understand its purpose (addition of an accessibility label, hint and role to the button).

IMAGES AND AUDIOS SENT ON CHAT APP BEING SAVED / STORED IN STORAGE SPACE ON FIREBASE



DECISIONS MADE

The audio vocal feature wasn’t part of the project requirement. However, I thought it was an addition that could bring great value to the user experience on chat app, so I proceed to implement it to enhance available features.

I also added additional code to the anonymous sign in method to ensure that the user ID persists between sessions.

- When users sign in in the app for the first time, a user ID is created, assigned to them and stored locally on their device.
- When users come back on the app, the app re-use this same user ID to sign in again users, thus allowing them to access their messages correctly between each session, and where they left it on their last connection.

11

Finalizing code revision, refactoring and last testings

Skills used

Critical thinking

WHY WAS THIS STEP IMPORTANT

Ensuring that the codes are optimized to facilitate possible appropriation by other developers in the future is useful and could possibly save time. It can also facilitate any future adjustments to the codes.

WHAT WAS THE GOAL

Reviewing each code to make sure everything was optimized as much as possible in order to facilitate future modification, addition or adjustment in the future.

Adding comments and clarifications points in the code where important for the benefit and better understanding of anyone else who may work on this project later.

```
//***Configuration for the Firebase taken from Firestore website to allow the whole app to connect
const firebaseConfig = {
  apiKey: [redacted]
  authDomain: [redacted]
  projectId: [redacted]
  storageBucket: [redacted]
  messagingSenderId: [redacted]
  appId: [redacted]
};


//***Initialize Firebase.
const app = initializeApp(firebaseConfig);


//***Initialize Cloud Firestore and get a reference to the service.
const db = initializeFirestore(app, {
  experimentalForceLongPolling: true
})


//***Used to initialize the storage handler (for storing photos
const storage = getStorage(app);
```

EXAMPLE OF CODE COMMENTS

TO FACILITATE UNDERSTANDING OF THE FILE AND POSSIBLE FUTURE UPDATES AND WORK

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

I positioned myself from the point of view of future colleagues who could work on my project. How can I make my project and my codes as clear as possible to promote its easy appropriation? I reviewed each file to bring improvements in certain places and add comments where I thought it could be useful.

07

README.md

Completing the README document for Chat app

Skills used

Communication
Content writing

WHY WAS THIS STEP IMPORTANT

Ensure Meet app is well documented and easily accessible by anyone interested.

WHAT WAS THE GOAL

Updating and completing the README file located in my Meet app Github repository. The goal was to ensure that all relevant information regarding Meet app is accessible under these five categories:

- Project description
- User interface
- User stories
- Technical aspect (overview)
- Technical aspects (development)
- App dependencies

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

Finding the right balance between giving the right level of information, while remaining as synthetic as possible. To help me, I made a first draft, which I then modified several times. I get inspired by other READMEs I've consulted and for which I found that the information presented was relevant.

DECISIONS MADE

I wrote the README documentation from A to Z, in terms of content, presentation and structure.

README SAMPLE - FULL VERSION ON GITHUB

README.md

Chat app documentation

Content

- Project description
- User interface
- User stories
- Technical aspects (overview)
- Technical aspects (development)
- App dependencies

Project description

Chat app was created to serve as a chatting app for mobile devices using React Native. Users can go on Chat app and exchange with other users using the same app. Chat app includes features and UIs to allow users to send messages, images, their location and more.

Chat app can be broken down in the five following points:

- **Who** — For any users who want to use a chat mobile app.
- **What** — A native chat app built with React Native.
- **When** — Whenever users want to communicate with each other.