

Chat mobile app

Development
process



*All usernames, passwords or other similar information shown in this presentation is fictive and for illustrative purpose only

TABLE OF CONTENTS

01

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

02

Developing the chat screen using Gifted Chat library

03

Connecting the app to a Cloud database

04

Implementing offline logic and client-side data storage

05

Implementing communication features and making the app accessible for screen readers

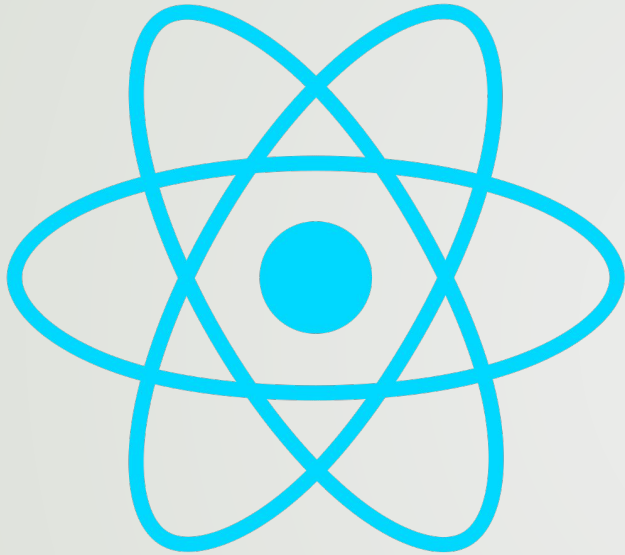
06

Finalizing code revision, refactoring and last testing

TABLE OF CONTENTS

07

Completing the README document



React Native

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 environment for efficient and well-oriented work from the beginning is not only necessary, but is also one of the keys to ensure smooth workflow and limit avoidable time loss due to inefficient project initialization.

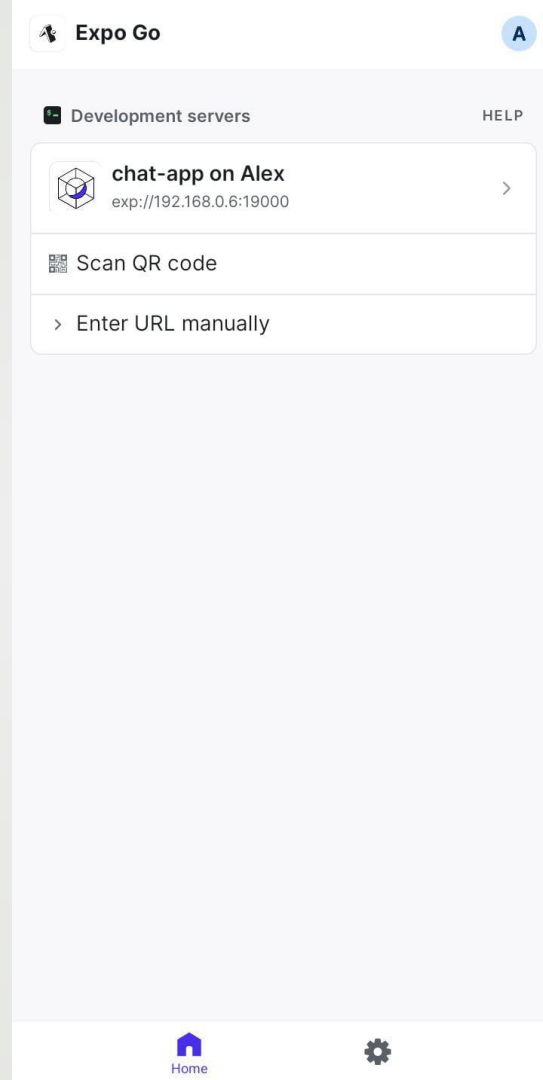
WHAT WAS THE GOAL

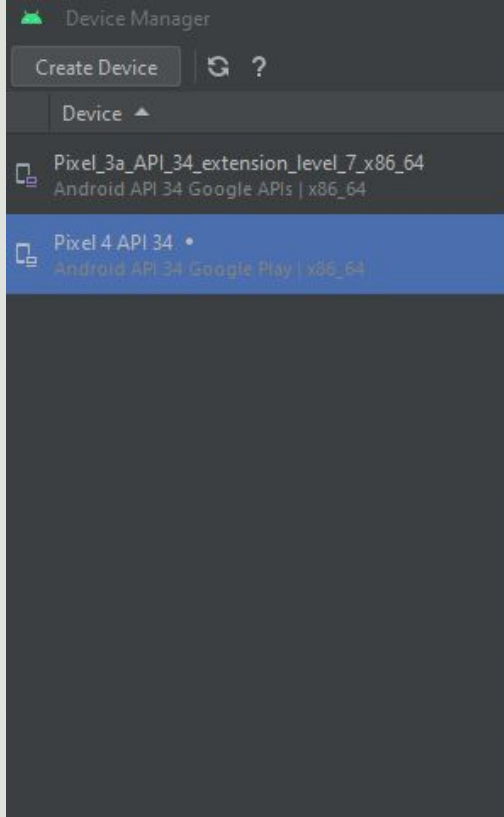
Setting up the development environment to ensure the project is properly initialized. More precisely, I:

- Updated 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 was 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

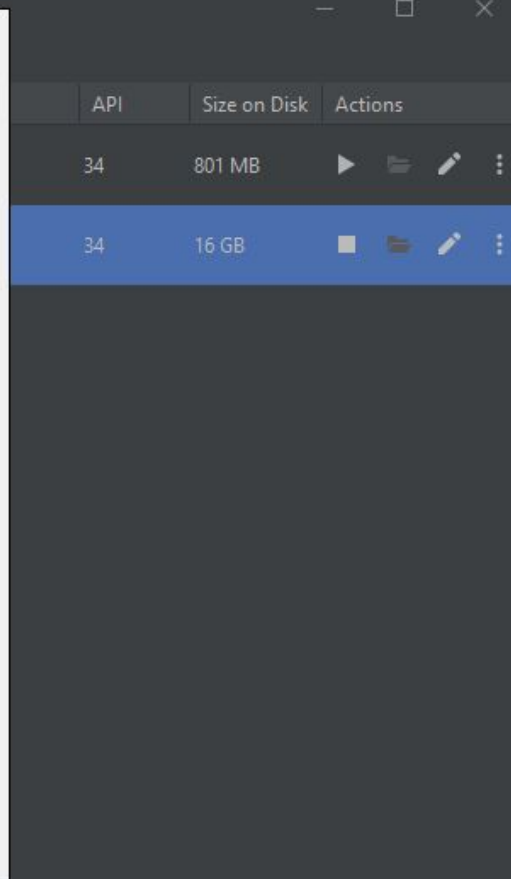
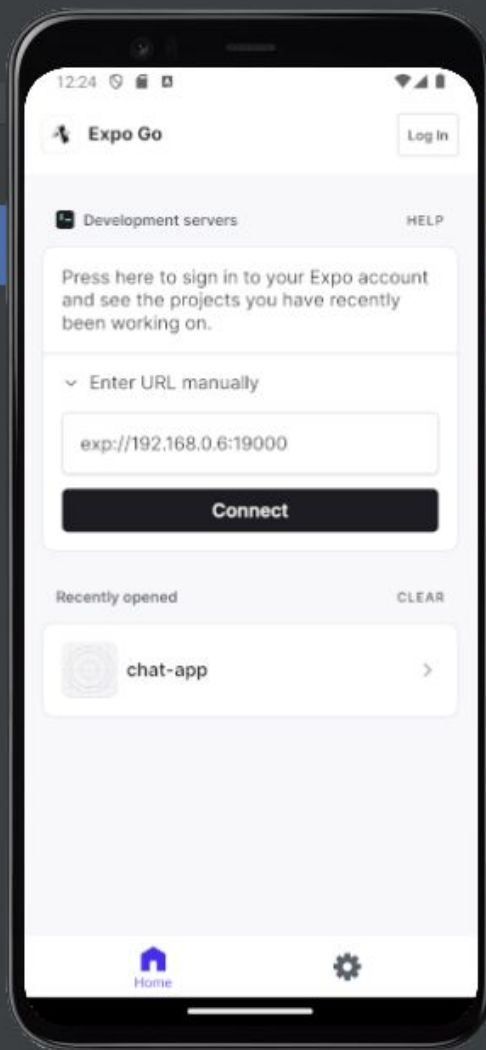


PROJECT RUNNING ON **EXPO GO** ON A PHYSICAL ANDROID DEVICE





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).

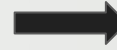
Creating the app layout / styling for the initial welcome screen and testing it on my own Android device as well as on Android Emulator (see next images).

CHAT APP INITIAL STARTING SCREEN DEVELOPMENT

Screen1

Hello Screen1!

GO TO SCREEN 2



Screen1

Hello Screen1!

Type your username here

GO TO SCREEN 2

CHAT APP INITIAL STARTING SCREEN DEVELOPMENT

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

Screen1

Hello Screen1!

TestUser

GO TO SCREEN 2



← TestUser

Hello Screen2!

CHAT APP INITIAL STARTING SCREEN DEVELOPMENT

*STYLING WELCOME SCREEN
ELEMENTS

StartScreen

Hello Screen1!

Your name

Start chatting



StartScreen

Hello Screen1!

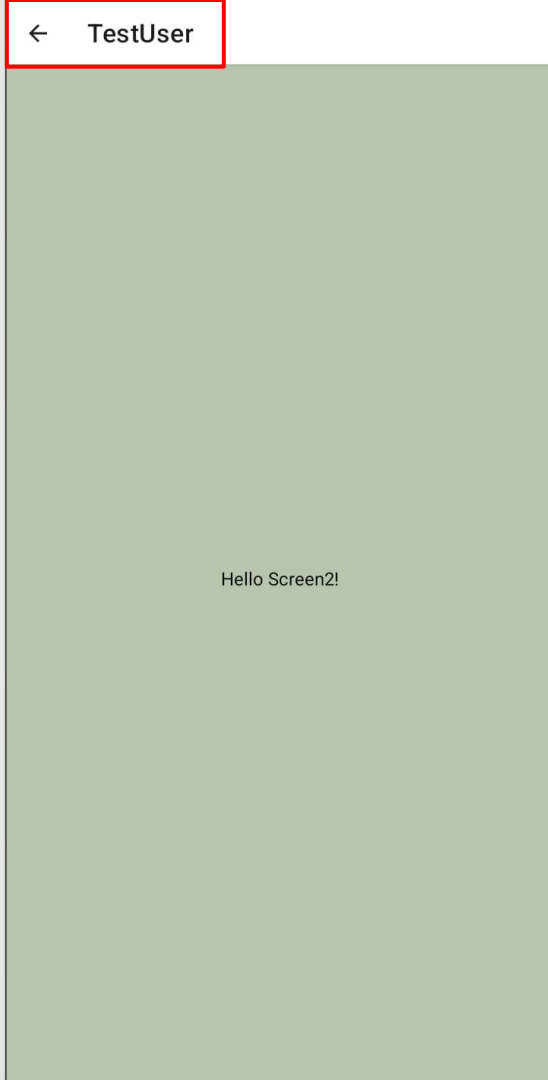
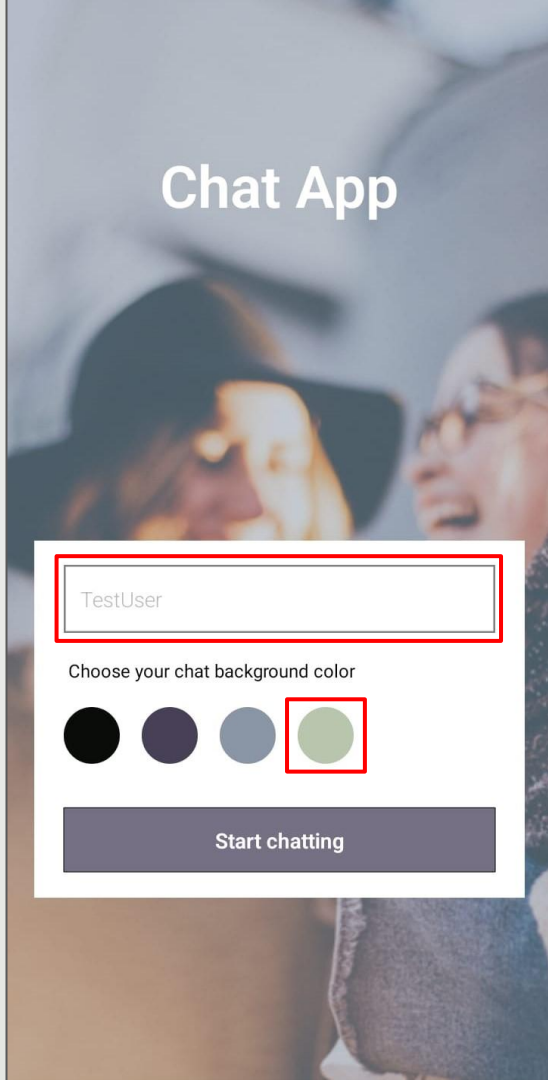
Your name

Start chatting

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 THE WELCOME STARTING
SCREEN



NAVIGATION TO CHAT SCREEN AFTER USERS PRESSED 'START CHATTING' ON THE WELCOME STARTING SCREEN

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

← TestUser

Hello Screen2!

← TestUser

Hello Screen2!



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.

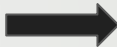
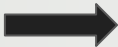
WHAT WAS THE GOAL

Developing the chat screen 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 (e.g.: message bubbles) to improve the visual.

Testing that everything is working as it should, using both my own Android device with Expo Go and Android Emulator.

CHAT SCREEN
DEVELOPMENT



Sep 6, 2023

This is a system message

Hello developer

9:10 PM

Type a message...

Sep 6, 2023

This is a system message

Hello developer

9:48 PM

Type a message...

Sep 6, 2023

This is a system message

Hello developer

9:48 PM

Hello

9:48 PM

Type a message...



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 necessary 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 on Google Cloud.

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

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.



Cloud Firestore

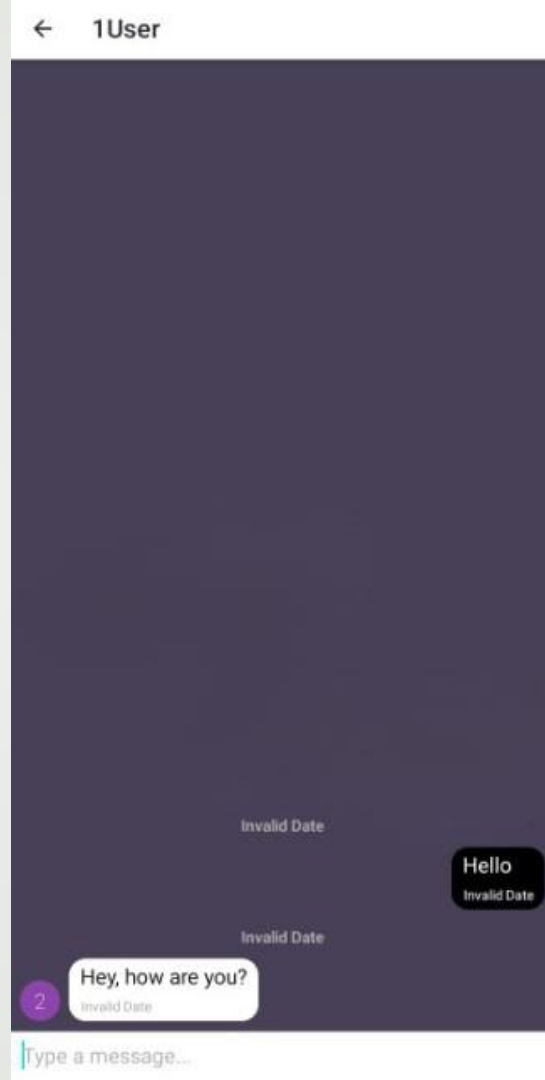
WHAT WAS THE GOAL (SUITE)

Implementing an anonymous authentication process for users using Firebase Authentication, 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 the Firestore database and locally on their device. Every time they go back on the app, this user ID is retrieved and users get connected with this

Testing the app to ensure everything is working as it should, using both Expo Go on my physical Android device and 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

The screenshot displays the Firebase console interface. On the left sidebar, the 'Firestore Database' and 'Authentication' options are highlighted with a red box. The main content area shows the 'chat-app-61899' project with the 'Cloud Firestore' database selected. A collection named 'messages' is visible, and a document with ID 'vcN7n3VKXrHX71zZNT0' is selected. The document details panel on the right shows the following fields:

- `_id`: "6ae961b0-05f2-4129-ad9c-ec346e8da78b"
- `createdAt`: 10 septembre 2023 à 14:28:41 UTC-6
- `text`: "Hello"
- `user`:
 - `_id`: "9ppWLhNiiLh5cVSlopixg9hEVEC2"
 - `name`: "1User"

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

The screenshot displays the Firebase console interface. On the left sidebar, the 'Firestore Database' and 'Authentication' options are highlighted with a red box. The main area shows the 'chat-app' project selected, with the 'messages' collection and a specific document highlighted. The document's content is as follows:

Field	Value
<code>_id</code>	<code>"7e88ab08-7bbc-4e74-ba1c-ad99532f815e"</code>
<code>createdAt</code>	<code>10 septembre 2023 à 14:28:42 UTC-6</code>
<code>text</code>	<code>"Hey, how are you?"</code>
<code>user</code>	<code>{_id: "XY5WKRiMONOTrWntZHDPEX9aCAq1", name: "2User"}</code>

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

At first, I tried to initialize the Firestore database in the code by importing and using the *getFirestore* function from `firebase / firestore` module. However, this caused incompatibility/network issues. After a few 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 found a solution and implement it in the code in replacement of the *getFirestore*, which resolves the issues.

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 app.



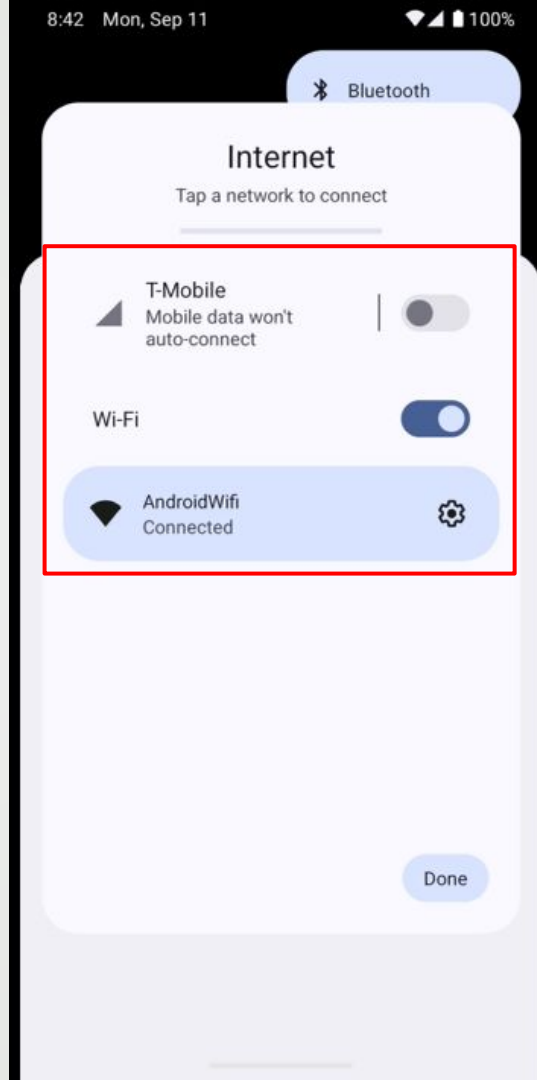
WHAT WAS THE GOAL

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.

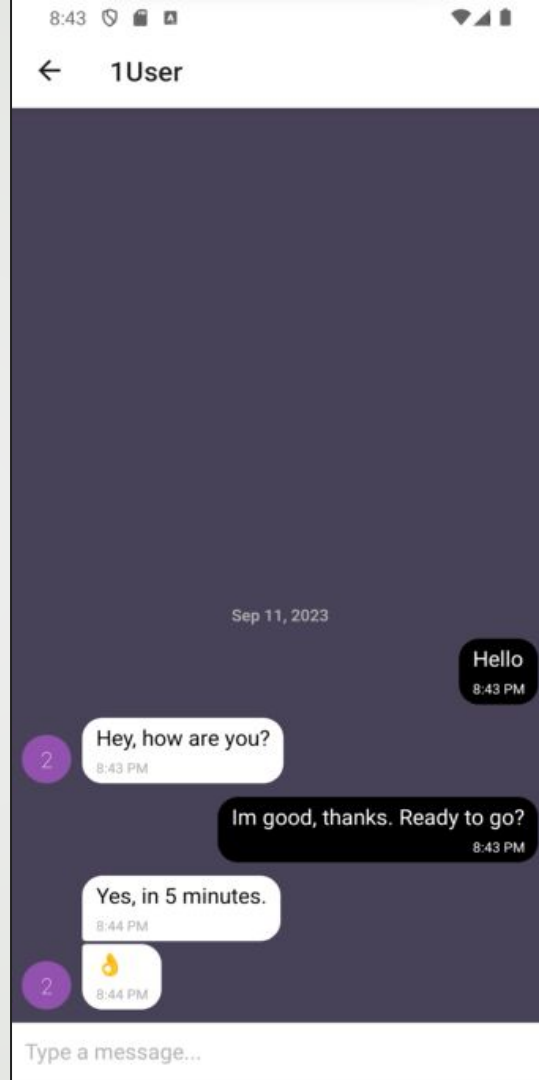
- To determine whether a user is online or not, `NetInfo` has been used (recommended package by the Expo team)
- If there's an internet connection, the data is fetched from the Firestore database, and if there's no internet connection, the data is fetched locally from `AsyncStorage`
- A code has also been written to ensure that the local storage cache is constantly kept up-to-date on new messages when there's an internet connection, so that the messages displayed from local storage when there's no network are also always up-to-date
- 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. When the internet connection is back, the text input field is shown again automatically, allowing users to write and send new messages

Testing the app to ensure everything is working as it should, using both Expo Go on my physical Android device and Android Emulator.

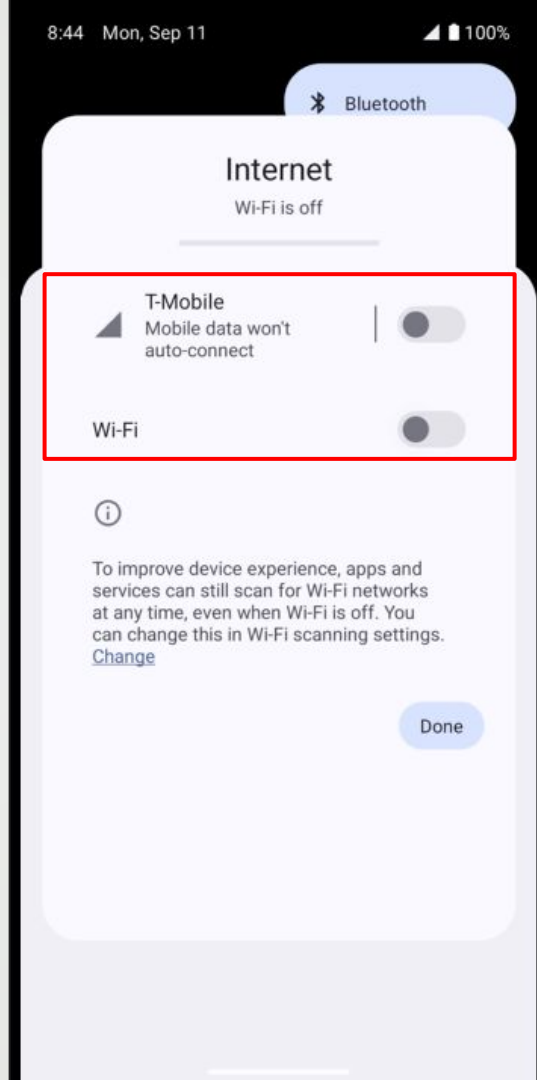
APP RUNNING WITH INTERNET



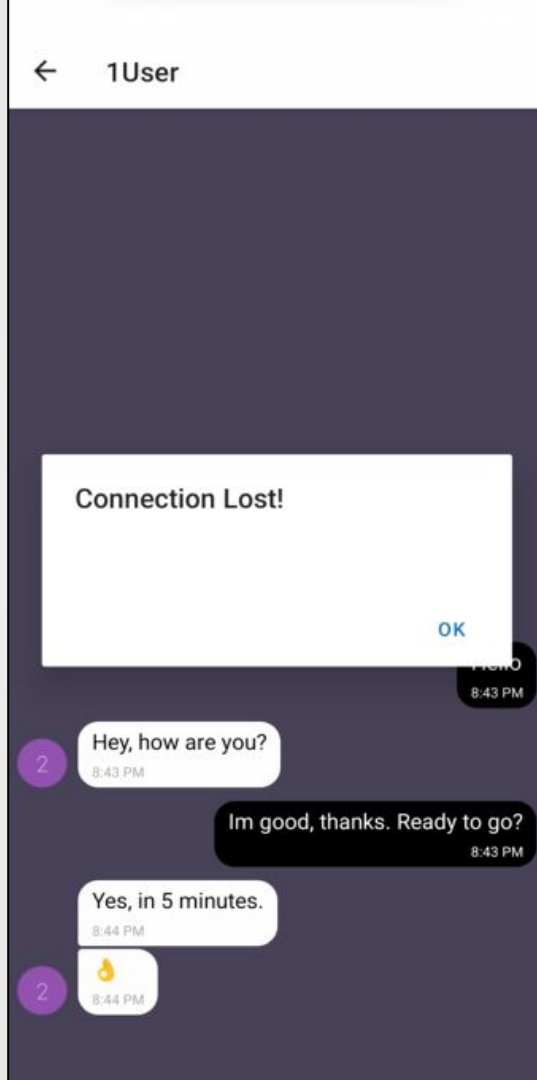
APP RUNNING WITH INTERNET - **USERS CAN SEND MESSAGES**



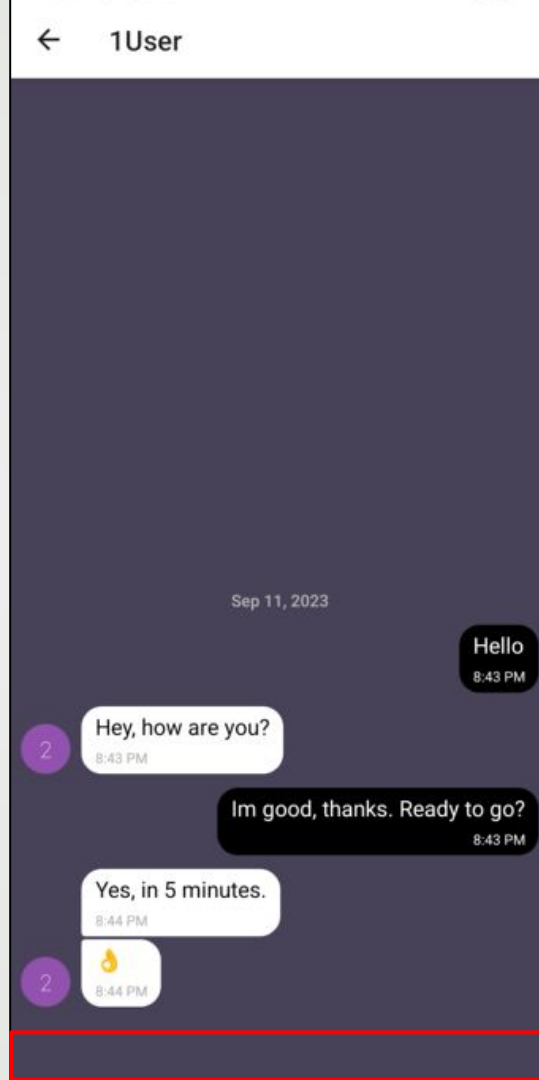
APP RUNNING WITHOUT INTERNET



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/RECEIVED BEFORE





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

Skills used

Research
Code writing
Debugging

WHY WAS THIS STEP IMPORTANT

Creating an app 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 product, it was important to add additional features, such as the possibility to send images, take photos, share location and send audios.

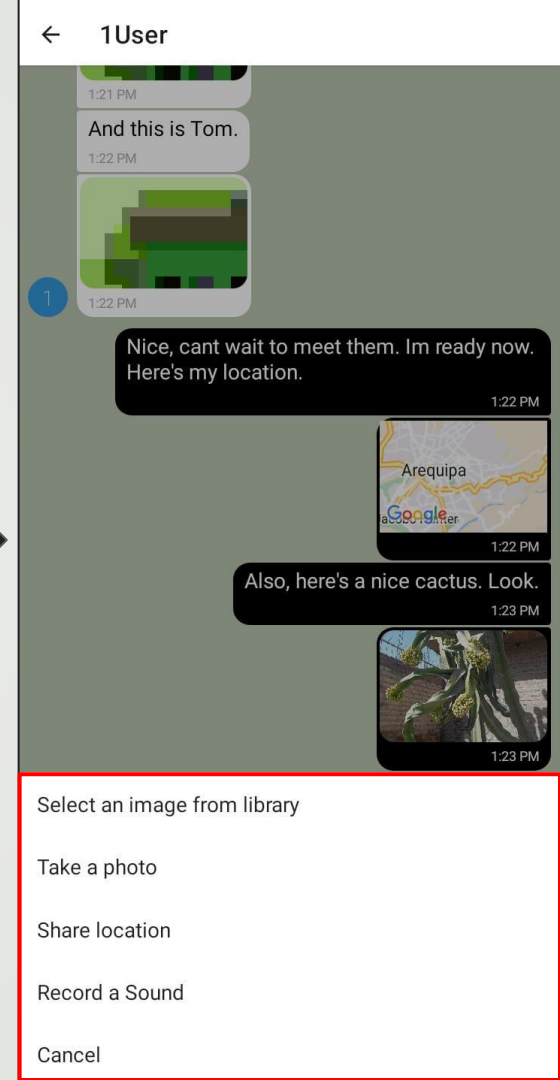
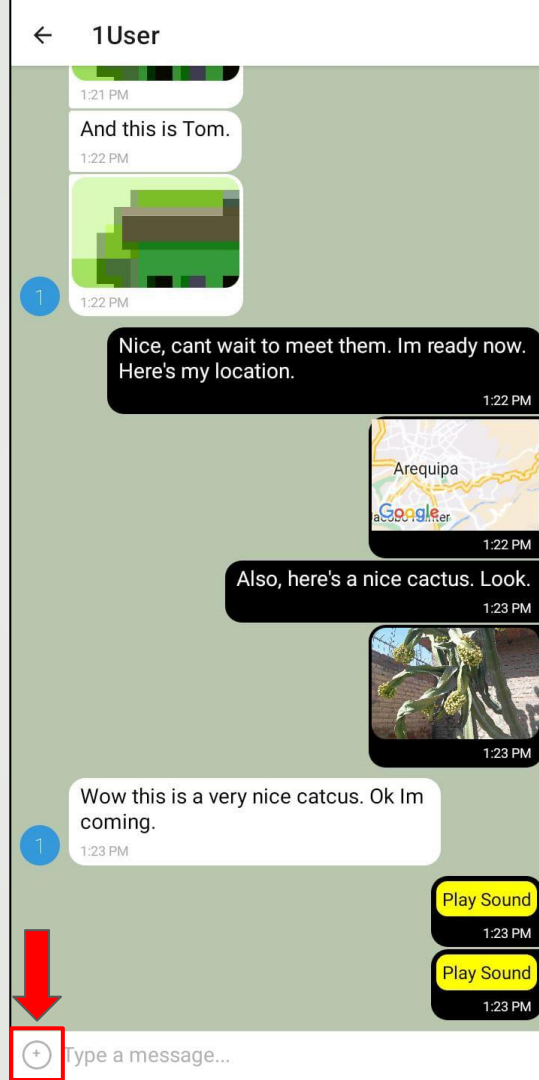
WHAT WAS THE GOAL

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 them, (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 / vocals, Expo's Audio API (*expo-av*) package has 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 FOR THE
FIRST TIME



Allow **Expo** to access
photos, media, and
files on your device?

DENY

ALLOW

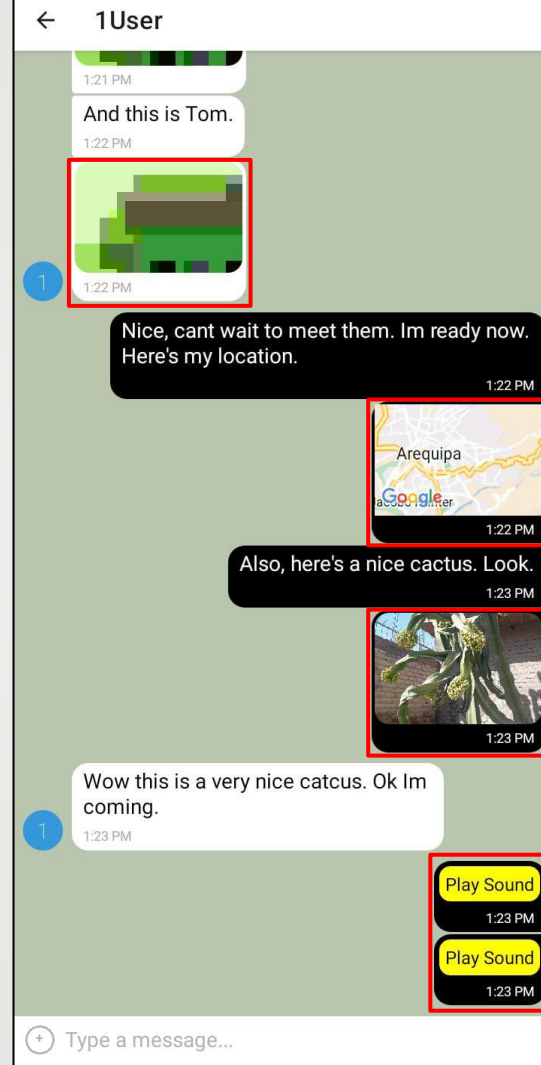
FROM TOP TO BOTTOM:

-IMAGE RECEIVED FROM
ANOTHER USER PHONE'S
LIBRARY

-SHARED LOCATION

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

-AUDIO VOCALS SENT AFTER
USING PHONE'S MICROPHONE





WHAT WAS THE GOAL (SUITE)


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

Adjusting the action (+) button (button users can click on to display more features) to make it more accessible, so that users with screen readers can understand its purpose (addition of an accessibility label, hint and role to the button code).

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

 **Firestore Database**

 **Authentication**

 **Storage**

Créer

Publier et surveiller

Analytics

Engager

Tous les produits

Spark
Sans frais 0 \$/mois


Mettre à niveau

chat-app ▾ Storage

[gs://chat-app-61899.appspot.com](#)

Importeur un fichier

undefined-1694648... ✕



Nom
undefined-1694648408441-bcf1e471-4dc...

Taille
21 625 octets

Type
image/jpeg

Date de création
13 sept. 2023, 17:40:10

Date de mise à jour
13 sept. 2023, 17:40:10

Emplacement du fichier

Autres métadonnées

	Nom	Taille	Type	Dernière modification
<input type="checkbox"/>	undefined-1694638865388-7623122a-e5cb-4976-8fd8-51ea5c6b1d7c.jpg	21.07 KB	image/jpeg	13 sept. 2023
<input type="checkbox"/>	undefined-1694648275824-763d4c07-86e7-4bf6-a30c-b1fc5b03c9fe.jpg	21.07 KB	image/jpeg	13 sept. 2023
<input type="checkbox"/>	undefined-1694648408441-bcf1e471-4dc1-4936-b996-e35139580863.jpg	21.12 KB	image/jpeg	13 sept. 2023
<input type="checkbox"/>	undefined-1694648456325-10c8f46a-eaef-4c18-8d54-bcf66d23cf6b.jpg	850.2 KB	image/jpeg	13 sept. 2023
<input type="checkbox"/>	undefined-1694655071722-recordin-g-9386b6c4-81fe-4612-ad7e-47ab5e4953ef.m4a	49.66 KB	audio/mpeg	13 sept. 2023
<input type="checkbox"/>	undefined-1694655080015-recordin-g-05944710-0564-42ee-8d65-69ef806bda84.m4a	49.96 KB	audio/mpeg	13 sept. 2023

DECISIONS MADE

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

I also added additional codes to the anonymous sign in method to ensure that each 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 later, the app re-uses this same user ID to log in users, thus allowing them to access their messages correctly between each session, and where they left it on their last connection



Finalizing code revision,
refactoring and last testing

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 modifications, additions or adjustments.

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

Carrying out several final tests on all the features to ensure everything work as it should.

```
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

CHALLENGES OR SPECIAL POINTS OF CONSIDERATION

I positioned myself from the point of view of future colleagues who could work on this project. How can I make this 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.



README .

md

Completing the README document

Skills used

Communication
Content writing

WHY WAS THIS STEP IMPORTANT

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

WHAT WAS THE GOAL

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

- Project description
- User interface
- User stories
- Technical aspects (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 at times. I also drew inspiration from other READMEs I've consulted for similar projects 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

