NEXT
CHAT

CASE STUDY

NextChat

a React Native App

Developer: Ian Baumeister

Repository: https://github.com/ibxibx/nextchat

Setup Instructions: https://github.com/ibxibx/nextchat?tab=readme-ov-file#-setup-instructions

During my Career Foundry Full Stack Web Development course, apart from the largest and most complex project which was the MyMDB app, I particularly enjoyed working on my NextChat - React Native mobile app for chatting.

The objective was to build a chat app for mobile devices using React Native. The app will provide users with a chat interface and options to share images and their location.

It's quite a feature-rich app that I built and tested with my tutor and mentor. The basic code snippets and parts were provided along the course set of lessons that encompassed the development of this app to start with and slowly enrich the app step by step with all the following features:

Tech Stack and Specs

- React Native
- Expo
- Google Firestore Database
- Google Firebase Authentication
- Gifted Chat library
- JavaScript

Supported Platforms

- iOS
- Android

Data Storage

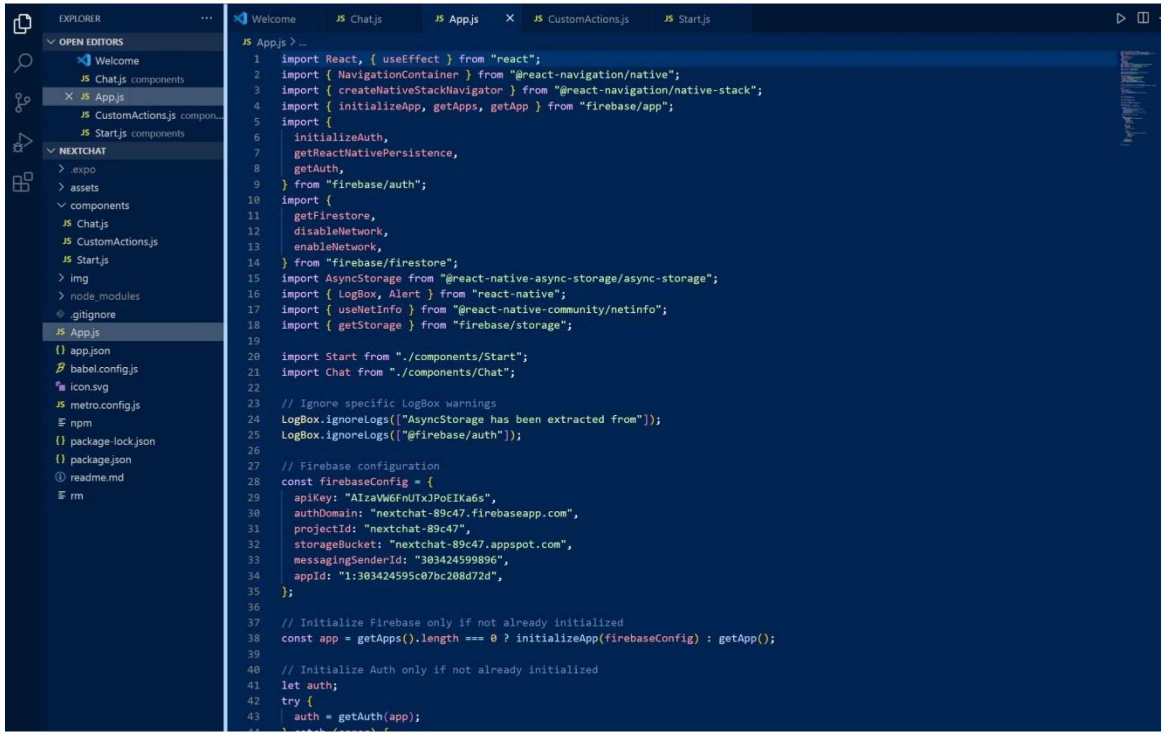
- Online storage: Google Firestore Database
- Offline storage: Local device storage

Authentication

- Anonymous authentication via Google Firebase

Image Handling

- Pick images from phone's library



```javascript
import React, { useEffect } from "react";
import { NavigationContainer } from "@react-navigation/native";
import { createNativeStackNavigator } from "@react-navigation/native-stack";
import { initializeApp, getApps, getApp } from "firebase/app";
import {
  initializeAuth,
  getReactNativePersistence,
  getAuth,
} from "firebase/auth";
import {
  getFirestore,
  disableNetwork,
  enableNetwork,
} from "firebase/firestore";
import AsyncStorage from "@react-native-async-storage/async-storage";
import { LogBox, Alert } from "react-native";
import { useNetInfo } from "@react-native-community/netinfo";
import { getStorage } from "firebase/storage";

import Start from "./components/Start";
import Chat from "./components/Chat";

// Ignore specific LogBox warnings
LogBox.ignoreLogs(["AsyncStorage has been extracted from"]);
LogBox.ignoreLogs(["@firebase/auth"]);

// Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaVW6FnUTxJPoEIKa6s",
  authDomain: "nextchat-89c47.firebaseapp.com",
  projectId: "nextchat-89c47",
  storageBucket: "nextchat-89c47.appspot.com",
  messagingSenderId: "303424599896",
  appId: "1:303424595c07bc208d72d",
};

// Initialize Firebase only if not already initialized
const app = getApps().length === 0 ? initializeApp(firebaseConfig) : getApp();

// Initialize Auth only if not already initialized
let auth;
try {
  auth = getAuth(app);
```

App.js Code Screenshot in Visual Studio Code

- Take photos using device's camera
- Store images in Firebase Cloud Storage

Location Sharing

- Read user's location data
- Send location data in map view via chat

Accessibility

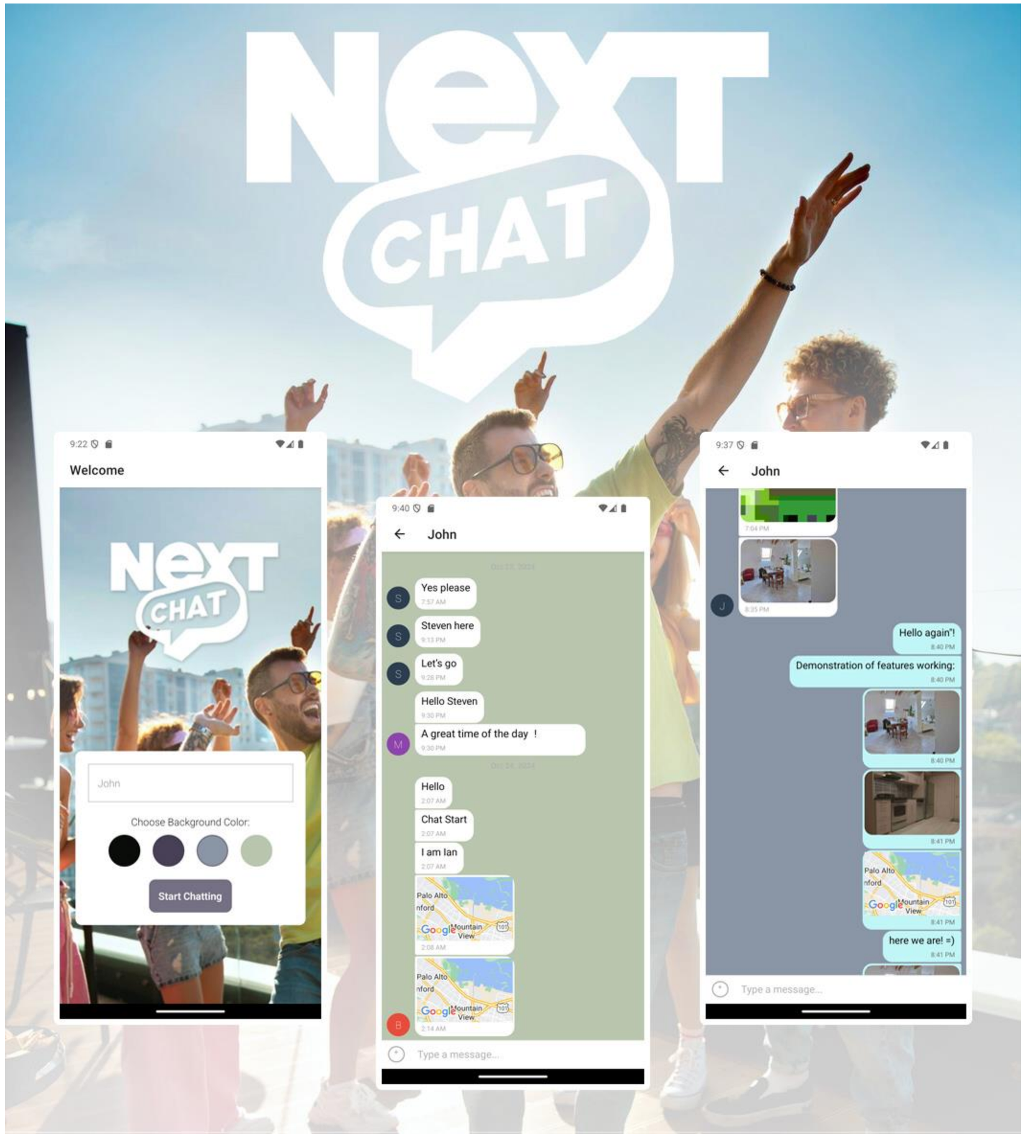
- Compatible with screen readers for users with visual impairments

App Features

- User-friendly chat interface
- Image sharing capabilities
- Real-time location sharing
- Offline message reading
- Customizable chat room backgrounds
- Screen reader compatibility for accessibility

User Needs

- Easy chat room entry for quick conversations
- Sending text messages to friends and family
- Sharing images of current activities
- Sharing location data



Yes please
7:57 AM
Steven here
9:13 PM
Let's go
9:28 PM
Hello Steven
9:30 PM
A great time of the day !
9:30 PM
Hello
2:07 AM
Chat Start
2:07 AM
I am Ian
2:07 AM
2:08 AM
2:14 AM

7:04 PM
8:35 PM
Hello again"!
8:40 PM
Demonstration of features working:
8:40 PM
8:40 PM
8:41 PM
8:41 PM
here we are! =)
8:41 PM

NextChat – App: Poster with the Start Screen and Chat Screens

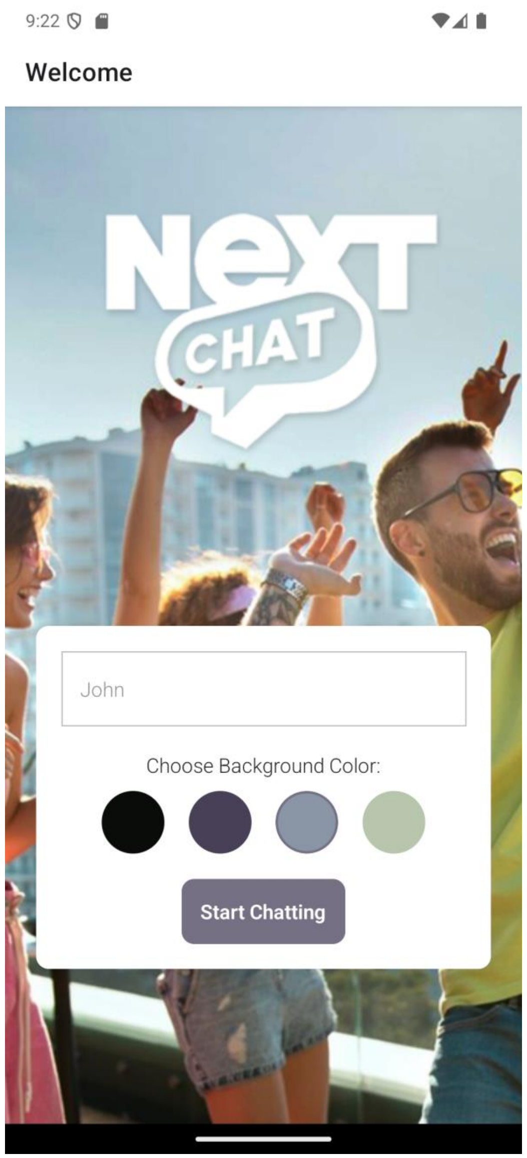
- Offline message access
- Accessibility for users with visual impairments

User Stories

- As a new user, I want to effortlessly join a chat room so I can quickly start conversations with my friends and family.
- As a user, I want the ability to send messages to my friends and family to stay updated and exchange news.
- As a user, I want to send images to my friends to share moments and show what I'm currently doing.
- As a user, I want to share my location with my friends to let them know where I am.
- As a user, I want to access my messages offline so I can revisit conversations anytime.
- As a visually impaired user, I want the chat app to be fully compatible with a screen reader so I can engage seamlessly with the chat interface.

Technical Requirements

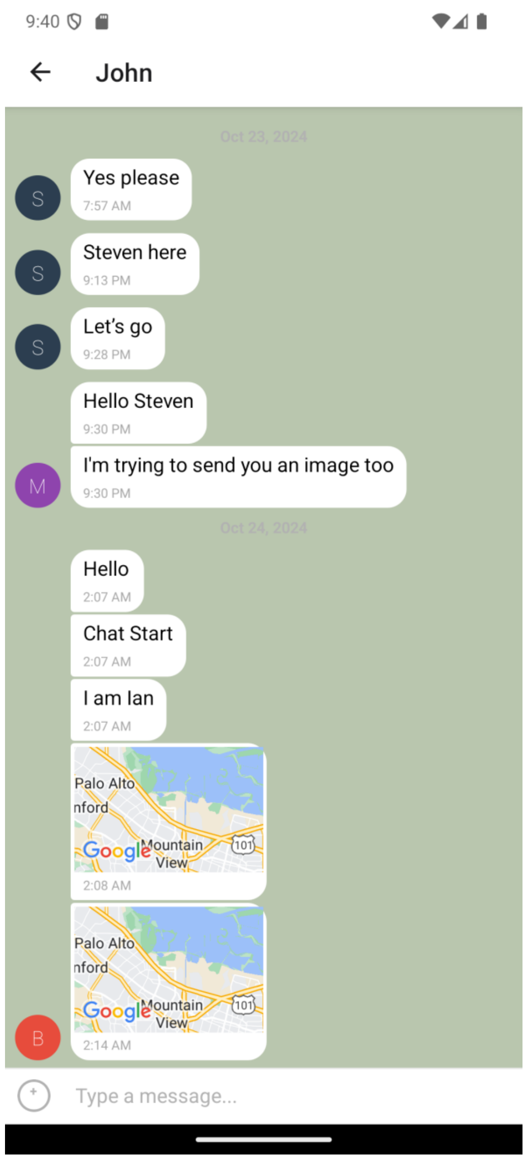
- Developed using React Native and Expo
- Styled according to the provided start screen design layout
- Chat conversations stored in Google Firestore Database
- Anonymous user authentication via Google Firebase
- Image picking from phone's library and camera integration
- Firebase Cloud Storage for image storage, chat conversations
- Device location data reading and sharing
- Gifted Chat library for chat interface and functionality
- Well-commented codebase


9:22

Welcome

John

Choose Background Color:


9:40

John

Oct 23, 2024

S Yes please 7:57 AM

S Steven here 9:13 PM

S Let's go 9:28 PM

Hello Steven 9:30 PM

M I'm trying to send you an image too 9:30 PM

Oct 24, 2024

Hello 2:07 AM

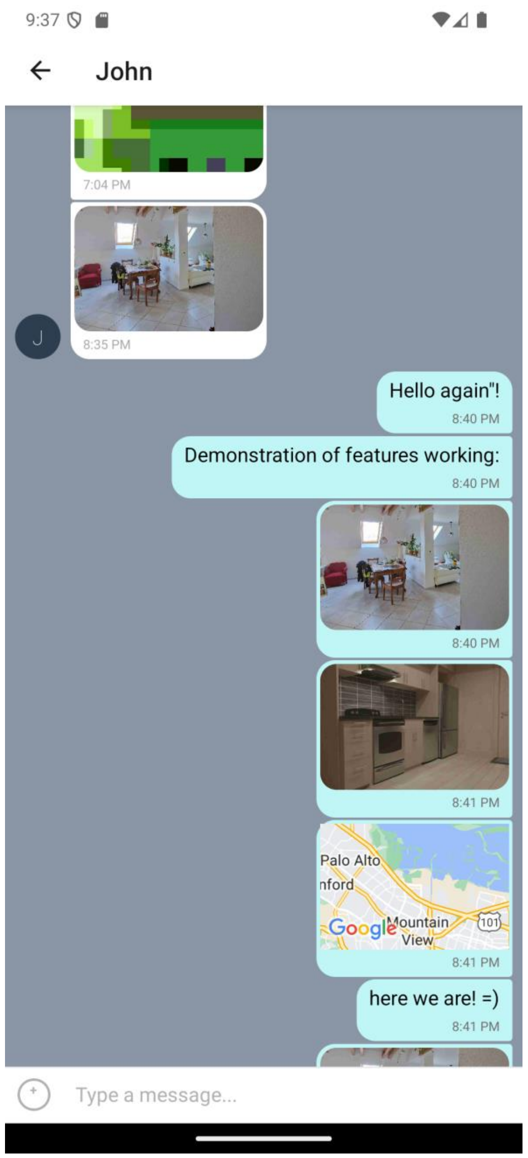
Chat Start 2:07 AM

I am Ian 2:07 AM

2:08 AM

B 2:14 AM

Type a message...


9:37

John

7:04 PM

J 8:35 PM

Hello again"! 8:40 PM

Demonstration of features working: 8:40 PM

8:40 PM

8:41 PM

8:41 PM

here we are! =) 8:41 PM

Type a message...

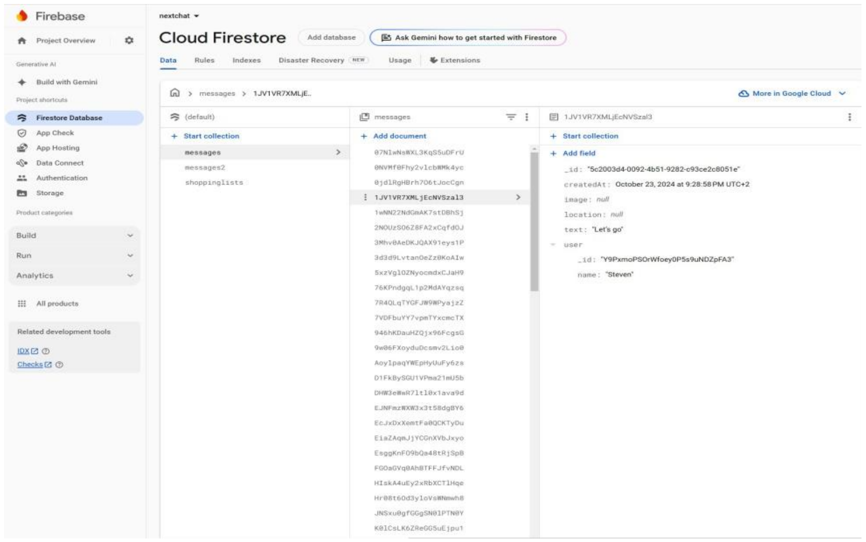
Screen Captions: Start Screen, Chat Screen, Chat Screen

The Development Steps, Challenges & Solutions

1. **Expo App Setup and testing on the Smartphone**
   First, I've set up the Expo project with the boilerplate for the chat app.

   As I was using Expo Software to create the boilerplate for the app and then use it for further testing and development, for some reason the Expo app wouldn't render the app on my phone to test and see the app code at work. The recommended older version of Node 16.19.0 did not work with Expo on my android phone.

   Using Node 16.19.0, I tried many older versions of the Expo app, to get a working combo, installing all these versions (up to 3 years old) one after the other but all these efforts did not render the app on Expo.



Cloud Firestore

messages > 1JV1VR7XMLjE..

```
(default)
  messages
  messages2
  shoppinglists

messages
  07NlwNsWXL3KqS5uDFrU
  0NVMf0Fhy2vlcbWMk4yc
  0jdlRgHBrh7O6tJocCgn
  1JV1VR7XMLjEcNVSzal3
  1wNN22NdGmAK7stDBhSj
  2NOUzSO6Z8FA2xCqfdOJ
  3Mhv0AeDKJQAX91eys1P
  3d3d9LvtanOeZz0KoAIw
  5xzVglOZNyocmdxCJaH9
  76KPndgqL1p2MdAYqzsq
  7R4QLqTYGFJW9WPyajzZ
  7VDFbuYY7vpmTYxcmcTX
  946hKDauHZQjx96FcgsG
  9w06FXoyduDcsmv2Lio0
  AoylpaqYWEpHyUuFy6zs
  D1FkBySGU1VPma21mU5b
  DHW3eWwR71tl0x1ava9d
  EJNFmzWXW3x3t58dgBY6
  EcJxDxXemtFa0QCKTyDu
  EiaZAqmJjYCGnXVbJxyo
  EsggKnFO9bQa48tRjSpB
  FGOaGVq0AhBTFFJfvNDL
  HIskA4uEy2xRbXCTlHqe
  Hr08t6Od3yloVsWNmwh8
  JNSxu0gfGGgSN01PTN0Y
  K01CsLK6ZReGG5uEjpu1

1JV1VR7XMLjEcNVSzal3
  _id: "5c2003d4-0092-4b51-9282-c93ce2c8051e"
  createdAt: October 23, 2024 at 9:28:58 PM UTC+2
  image: null
  location: null
  text: "Let's go"
  user
    _id: "Y9PxmoPSOrWfoey0P5s9uNDZpFA3"
    name: "Steven"
```

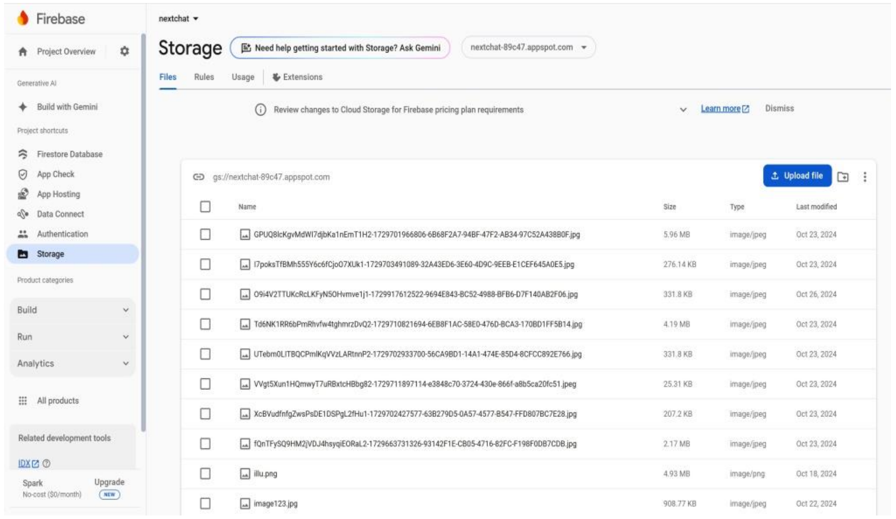
Server Side: Google Firestore

2. **Building the Interface, App, Start and Chat Screens.**
   Using Expo Snack and Android Studio

   Then I've built the general welcome screen with its layout Start.js and added the custom colour function (for the Chat Screen background) along with the user id name authentication.

   After using the IPv4 address as the IP address to render the app on my mobile - which didn't work either - I found that the Expo Snack Box online had the right settings and environment to test the app. I used the sandbox/snack until the integration with Google Firebase came into play! The Expo Snack was a short time solution – it did not work at all with Google Firebase with all the further tweaking - it didn't throw an error, but it also didn't render the app in the preview panel - meaning Expo Snack and Google Firebase have a has a compatibility issue.

   Android Studio played a huge role here as an emulator, where I was able to render the app and test it on Expo's local hosting, over and over



Storage

Review changes to Cloud Storage for Firebase pricing plan requirements

gs://nextchat-89c47.appspot.com

Name	Size	Type	Last modified
GPUQ8lcKgvMdWI7djbKa1nEmT1H2-1729701966806-6B68F2A7-94BF-47F2-AB34-97C52A438B0F.jpg	5.96 MB	image/jpeg	Oct 23, 2024
I7poksTfBMh555Y6c6fCjo07XUk1-1729703491089-32A43ED6-3E60-4D9C-9EEB-E1CEF645A0E5.jpg	276.14 KB	image/jpeg	Oct 23, 2024
O9i4V2TTUKcRcLKFyN5OHvmve1j1-1729917612522-9694E843-BC52-4988-BFB6-D7F140AB2F06.jpg	331.8 KB	image/jpeg	Oct 26, 2024
Td6NK1RR6bPmRhvfw4tghmrzDvQ2-1729710821694-6EB8F1AC-58E0-476D-BCA3-170BD1FF5B14.jpg	4.19 MB	image/jpeg	Oct 23, 2024
UTebm0LITBQCPmlKqVVzLARtnnP2-1729702933700-56CA9BD1-14A1-474E-85D4-8CFCC892E766.jpg	331.8 KB	image/jpeg	Oct 23, 2024
VVgt5Xun1HQmwyT7uRBxtcHBbg82-1729711897114-e3848c70-3724-430e-866f-a8b5ca20fc51.jpeg	25.31 KB	image/jpeg	Oct 23, 2024
XcBVudfnfgZwsPsDE1DSPgL2fHu1-1729702427577-63B279D5-0A57-4577-B547-FFD807BC7E28.jpg	207.2 KB	image/jpeg	Oct 23, 2024
fQnTFySQ9HM2jVDJ4hsyqiEORaL2-1729663731326-93142F1E-CB05-4716-82FC-F198F0DB7CDB.jpg	2.17 MB	image/jpeg	Oct 23, 2024
illu.png	4.93 MB	image/png	Oct 18, 2024
image123.jpg	908.77 KB	image/jpeg	Oct 22, 2024

Google Firebase – Photo / Media Storage

again until everything worked and ran smoothly.

3. **Google Firestore Configuration and Integration**
After that I've set up the Google Firestore with its Storage and tested them by creating a sample database with a sample collection of data and integrated it into the app. I configured and integrated Google Firebase as the storage provider for all messages and media files. After adding the media to virtual memory in Android Studio, I saw the code in action that handles the transfer and storage of the media to the Firebase drive. It was great to see it working seamlessly.

4. **Chat Screen and its Functions**
The Chat component features a familiar and widely used mobile chat layout, which is recommended for successful app adoption. I added the second screen, Chat.js, which logs the username and displays the background colour chosen on the intro screen. Later, I've added the offline mode handling functionality and the display of the messages without the messaging bar when the internet connection is interrupted or lost.

Media Storage, Selection and Sending Problem. I was not able to figure out how the local storage on the virtual mobile phone in the Expo test environment stores the media files: photos and videos, which I couldn't access (even though the correct code was implemented).
In the component code, in the 'CustomActions' component:

```
// Handle image upload process and send message with image URL
 const uploadAndSendImage = async (imageURI) => {
 try {
 const uniqueRefString = generateReference(imageURI);
 const newUploadRef = ref(storage, uniqueRefString);
 const response = await fetch(imageURI);
 const blob = await response.blob();
 const snapshot = await uploadBytes(newUploadRef, blob);
 const imageURL = await getDownloadURL(snapshot.ref);

onSend({
_id: uniqueRefString,
createdAt: new Date(),
user: {
_id: userID,
 name: "User",
 },
 image: imageURL,
 });
  } catch (error) {
 console.error("Error uploading image", error);
 Alert.alert("Error uploading image", error.message);
 }
};
```

This function would not work, nor would the ‘ImagePicker’ from the gallery. It took me several steps to understand why and to analyse the error logs in Expo. We went through the code with my mentor, breaking it down into pieces and trying to test each part separately. Then we realised that I was not using the correct code for the correct User Id function and its use in Chat and ‘CustomActions’ (a misspelling and incorrect implementation that was preventing it from working), which was preventing the media handling components from performing their function.

5. **Custom Actions – the NextChat coolest Features**
   I added all the features in CustomActions.js and tested them one by one, the camera with front and rear view ‘ImagePicker’ - sending these images, the current location sending, I've done the design customisation of the chat bubbles with time stamps, text colour etc. using the Gifted Chat tool.

   One of the tasks was to simulate a live chat between two users at the same time and record a screen video - which wouldn't work as Expo wouldn't work on the mobile - so I tried to solve this problem using ‘Blue Stacks X’ as an Android emulator, downloaded and installed the Expo app on it, tried to run it on my screen next to Android Studio - but the same story - as with the mobile - didn't work, but I tried every possible way known to me at the time to create and showcase a live chat. Only Android Studio rendered my Expo app correctly and fully.

6. **The Finale, App Testing, My Surprise Moment**
   I did all the testing and debugging until the app was bug free and functional and updated the GitHub repository.

   We’ve managed to test the app from 2 devices at the same time - as my mentor Stephen was running my app code locally on his PC in Android Studio during our Skype call - and I was running the same code on my PC, we even laughed - he thought he was testing the app alone on the call and asked - what are these messages appearing - I replied - no way - are we chatting live now!? My app (with the entire tech in it) facilitated a live chat between 2 continents - my chin dropped…to my surprise the app felt ‘alive’! But then I realised that if the Google Firebase stores and displays the messages from all the active users - it should display all messages in real time as well, even though the app was running on 2 local servers on 2 different continents... the chat still worked for us. It was fun and a truly memorable moment. The power of technology never ceases to surprise, inspire and move me!

Branding

I did a quick search online and used an Claude.ai tool to come up with over a hundred cool possible brand names (2-word combinations) that would capture the essence of the product's functionality, I chose NextChat for maximum clarity, and designed the logo by overlaying the word "Next" with a chat bubble containing the word "Chat" forming a bold and stamp-like assertive logotype.

Final thoughts and reflections

As I've started to develop my first native mobile app, I've noticed the simplicity, versatility, precision and practicality of the React Native JavaScript-based mobile app framework.

While building the structure of a chat-type app with its components and dependencies, middleware and features, it created a deeper understanding and appreciation of the framework and its benefits.

My role in this project was to create a feature-rich basic native chat app and learn from the testing practices using Expo App and Android Studio emulators.

The unique challenges of building, integrating and testing features such as user-friendly chat layout, photo sharing, location sharing, offline message reading, customisable chat room backgrounds, screen reader compatibility for accessibility was my first deep dive into mastering React Native.

The technical stack also includes Expo for building and deploying the app, the Google Firestore database and Google Firebase for storing and serving custom chat data, and the Gifted Chat library for the chat tools.

I've followed the project deliverables with a focus on fast and clean implementation, without distracting myself with cool and fancy extensions or features. Adhering to 'less is more' and practicality for maximum efficiency and 'lean development' were my key principles. Avoiding overly complicated component structures, early testing on real devices, logic and function based code structure were my takeaways from this experience. Looking back at the deficiencies of Expo and the time-consuming testing on Android Studio I'd prefer a smarter developer tool (which I'll find soon) that makes testing and error logs easy to work upon and faster to debug, plus the feature to test the apps on several parallel emulators on same screen.

Through the NextChat project, I've gained vital knowledge, a good feel for the current tech stack combination, and the ability to assess the strengths and opportunities of building a React Native project.

The further improvements and features I see for NextChat:

- offering a password secured log in authentication
- audio call and video call functionality
- offering video or limited time video-clip taking, saving and sending
- audio recording, saving and sending
- browsing of video files in the gallery and sending them
- file or document browsing in the device memory and sending
- real location detection and live location sending, customization of chat background with images.