ChatApp: A Mobile Development Case Study.

PURPOSE

This project was created as part of my web development course at Career Foundry to demonstrate mastery of full stack JavaScript development.

OBJECTIVE

The aim of the project is 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.

C:\Users\nagen\OneDrive\Desktop\Chat-App\components\Chat.js

```javascript
import { useState, useEffect } from "react";
import { StyleSheet, View, Platform, KeyboardAvoidingView } from "react-native";
import { Bubble, GiftedChat, InputToolbar } from "react-native-gifted-chat";
import { collection, query, orderBy, onSnapshot, addDoc, serverTimestamp } from "firebase/firestore
import AsyncStorage from "@react-native-async-storage/async-storage";
import CustomActions from "./CustomActions";
import MapView from "react-native-maps";

const Chat = ({ route, navigation, db, isConnected, storage }) => {
  const [messages, setMessages] = useState([]);
  const { userId, name, backgroundColor } = route.params;

  useEffect(() => {
    navigation.setOptions({ title: name });

    let unsubscribe;

    // Load cached messages from AsyncStorage for offline support
    const loadCachedMessages = async () => {
      try {
        const cachedMessages = await AsyncStorage.getItem("messages");
        if (cachedMessages) {
          setMessages(JSON.parse(cachedMessages));
        }
      } catch (error) {
        console.error("Failed to load messages from cache:", error);
      }
    };

    if (isConnected) {
      // Create query to get messages in descending order
      const messagesQuery = query(collection(db, "messages"), orderBy("createdAt", "desc"));

      // Subscribe to real-time updates from Firestore
      unsubscribe = onSnapshot(messagesQuery, async (snapshot) => {
        const messagesList = snapshot.docs.map(doc => {
          const data = doc.data();
          return {
            _id: doc.id,
            ...data,
            createdAt: data.createdAt ? data.createdAt.toDate() : new Date()
```

Duration

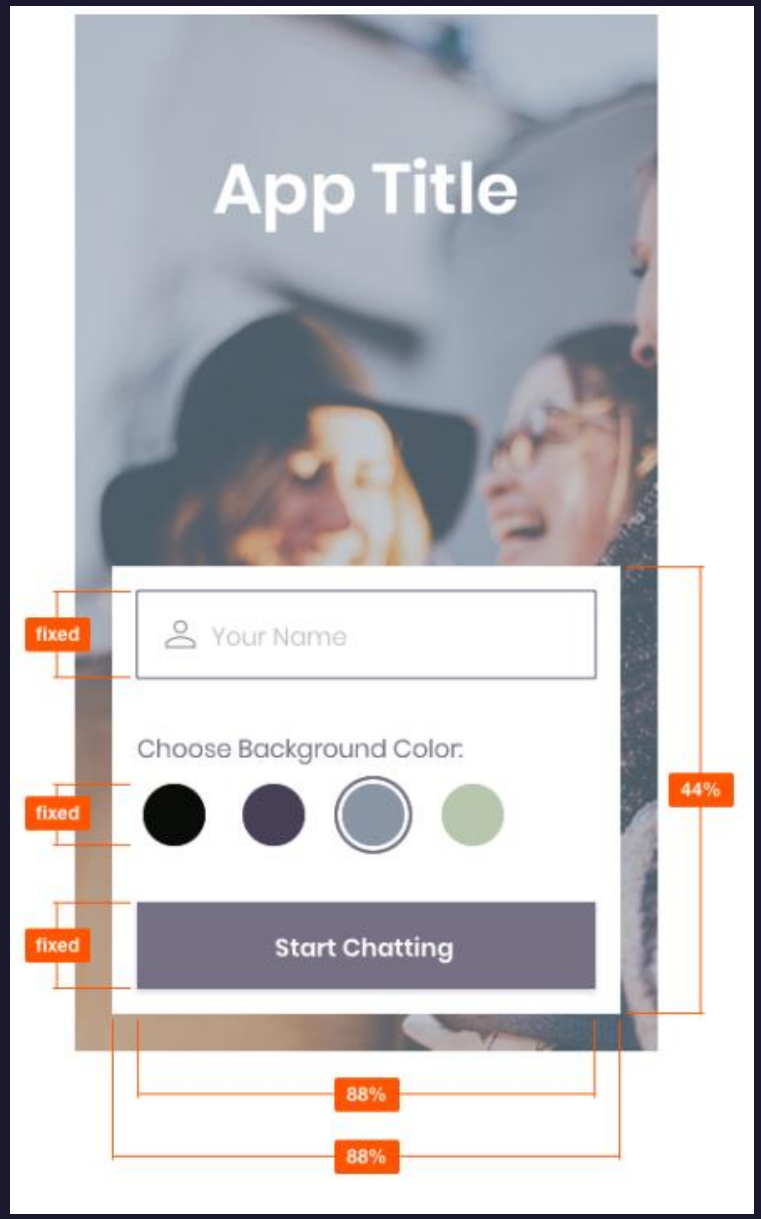
Developing the client-side of the chat app took more time than setting up the backend. I focused on understanding how React Native components, navigation, and asynchronous data handling work together to create a smooth user experience. Special attention was given to integrating Firebase services like Firestore, Authentication, and Storage, while also ensuring offline functionality and accessibility.

Credits

- Project Developer: Padmaja Pinnika
- Tutor: Ezequiel De Simone
- Mentor: Neal Peters

Methodologies & Tools

- **React Native** – Core framework for building cross-platform mobile UI
- **Expo** – Streamlined development and deployment environment
- **Firebase** – Backend services (Firestore, Storage, Authentication)
- **Gifted Chat** – Prebuilt chat UI components
- **Google Maps API** – Location sharing integration
- **AsyncStorage** – Offline message caching
- **React Navigation** – For managing screen transitions
- **Accessibility Testing** – Screen reader compatibility and inclusive design
- **Git & GitHub** – Version control and source code hosting



Design & Planning

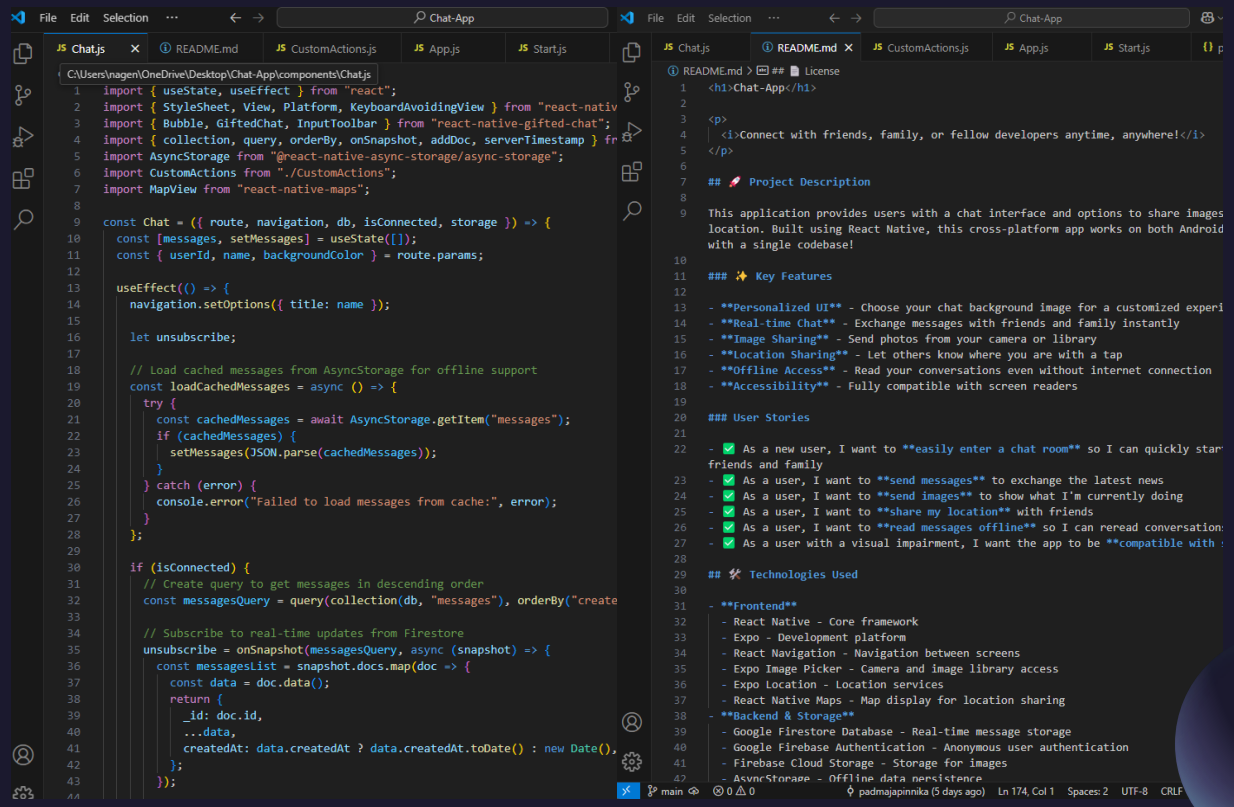
The project began with wireframing the user flow, from a welcome screen to a fully functional chat interface. Key considerations included:

- User authentication

- Storage for messages and media

- Offline functionality

Development Process

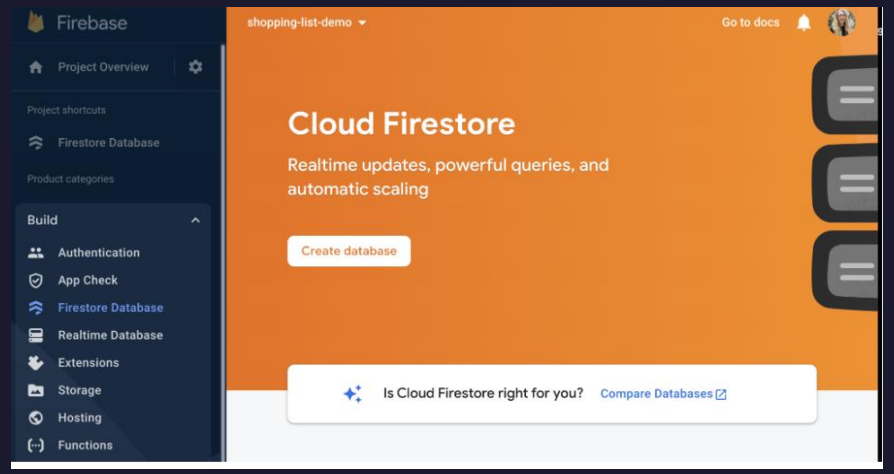


The app was developed using React Native for the frontend and Firebase for real-time database and storage. Key steps included:

- Setting up user authentication with **Firebase Auth**
- Implementing real-time messaging with **Cloud Firestore**
- Integrating **Firebase Storage** for image and media uploads
- Adding **offline support** using AsyncStorage

SERVER-SIDE

- **Implemented Firebase BaaS** to manage authentication, data storage, and media handling using built-in SDKs and APIs
- **Used Firestore (NoSQL)** for real-time message syncing and Firebase Storage for secure image uploads
- **Ensured secure communication** via HTTPS, storing and retrieving data in JSON format

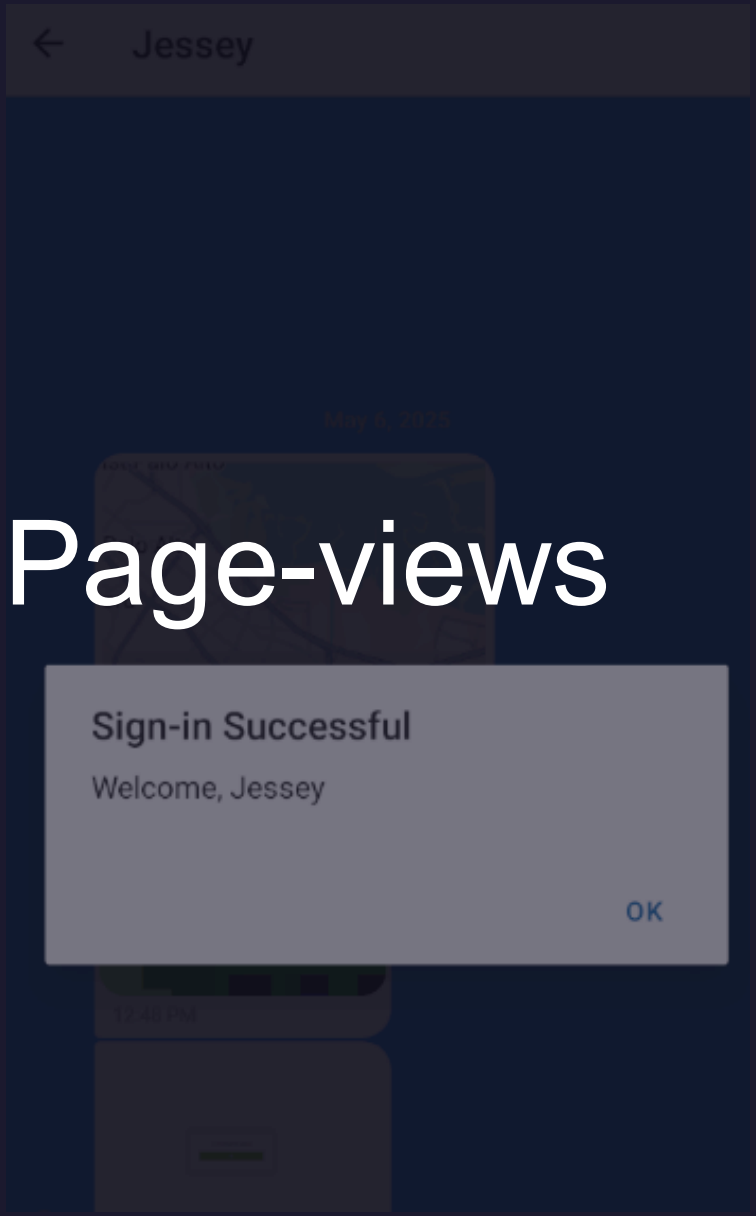


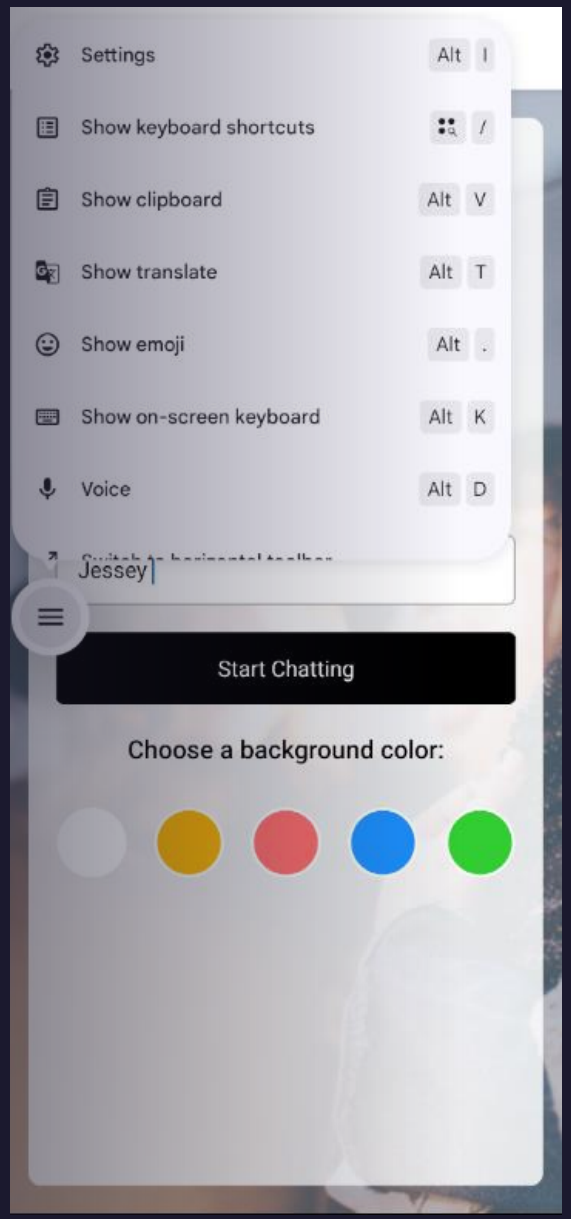
```
{
  "name": "chat-app",
  "version": "1.0.0",
  "main": "index.js",
  ▷Debug
  "scripts": {
    "start": "expo start",
    "android": "expo run:android",
    "ios": "expo run:ios",
    "web": "expo start --web"
  },
  "dependencies": {
    "@react-native-async-storage/async-storage": "^1.24.0",
    "@react-native-community/netinfo": "^11.4.1",
    "@react-navigation/native": "^7.1.6",
    "@react-navigation/native-stack": "^7.3.10",
    "expo": "^52.0.46",
    "expo-image-picker": "~16.0.6",
    "expo-location": "~18.0.10",
    "expo-status-bar": "~2.0.1",
    "firebase": "^10.3.1",
    "react": "18.3.1",
    "react-native": "0.76.9",
    "react-native-elements": "^3.4.3",
    "react-native-gifted-chat": "2.6.3",
    "react-native-keyboard-aware-scroll-view": "^0.9.5",
    "react-native-maps": "1.18.0",
    "react-native-reanimated": "~3.16.1",
    "react-native-safe-area-context": "4.12.0",
    "react-native-screens": "~4.4.0"
  },
  "devDependencies": {
    "@babel/core": "^7.20.0"
  },
  "private": true
}
```

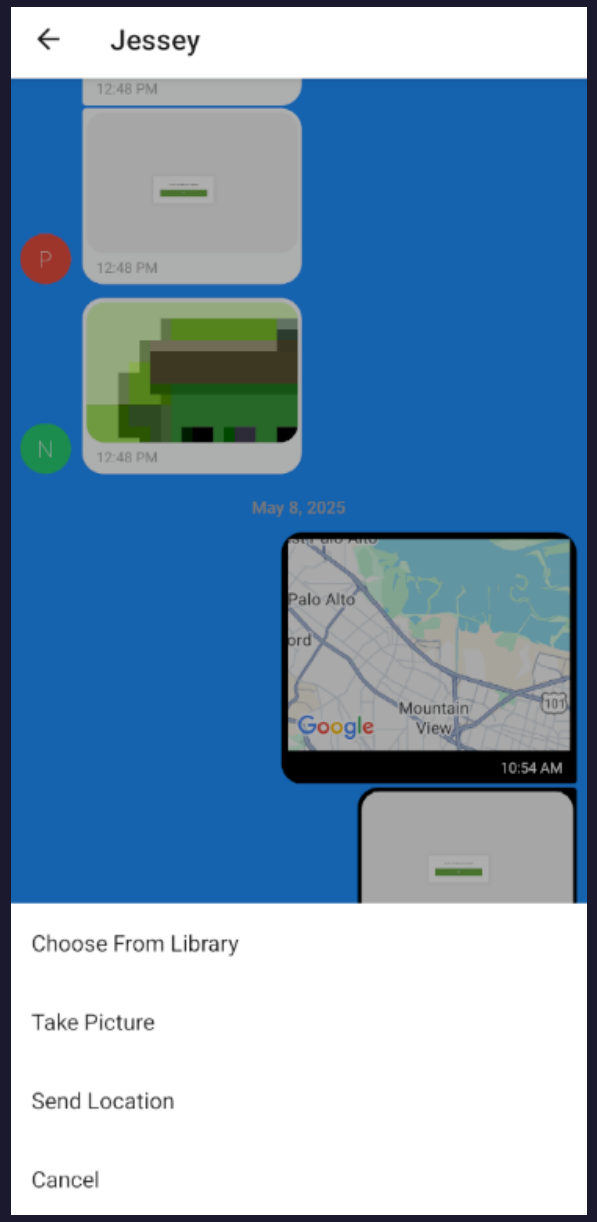
CLIENT-SIDE

- **Built with React Native and Expo** for cross-platform performance and smooth user experience
- **Integrated Gifted Chat, React Navigation, and AsyncStorage** to support messaging UI, screen transitions, and offline caching
- **Connected to Firebase services** for real-time messaging, image uploads, and location sharing via Google Maps API using JSON data format

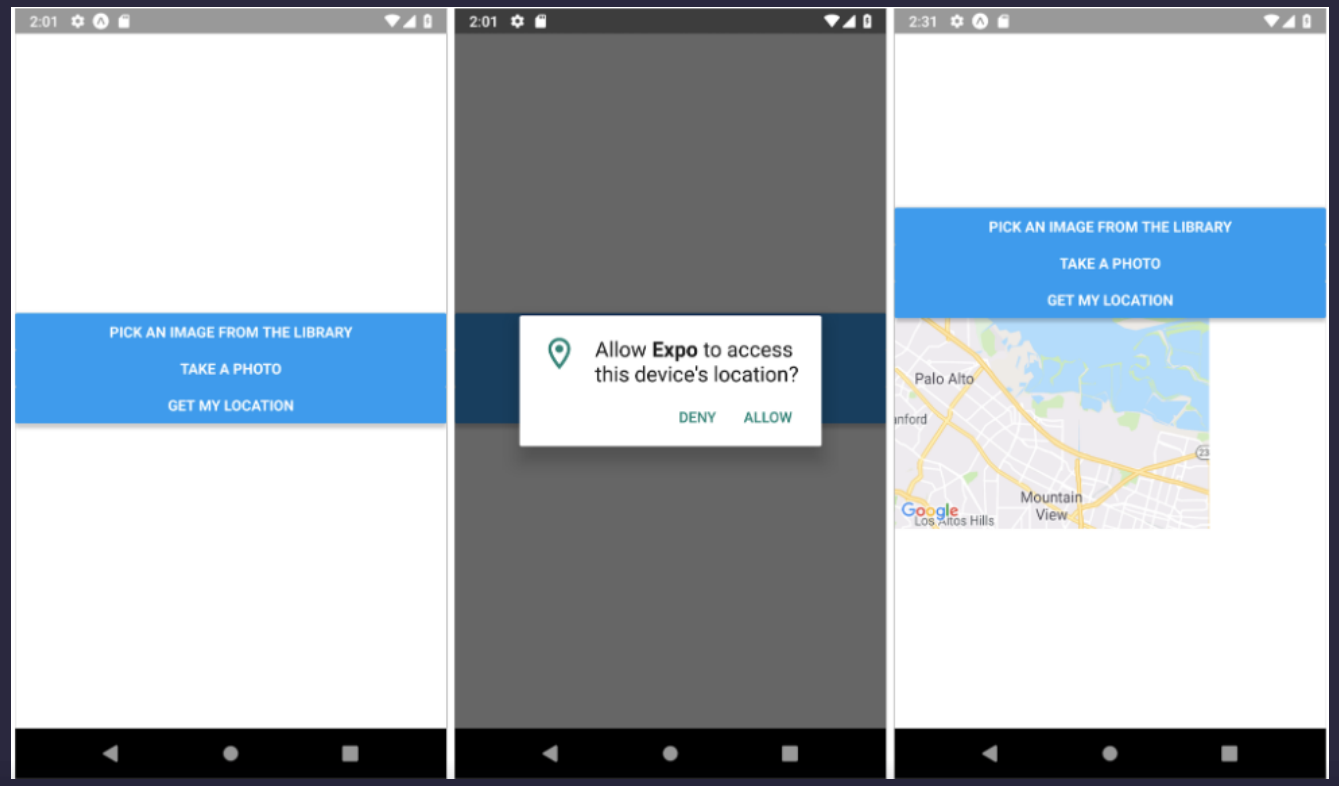
Page-views

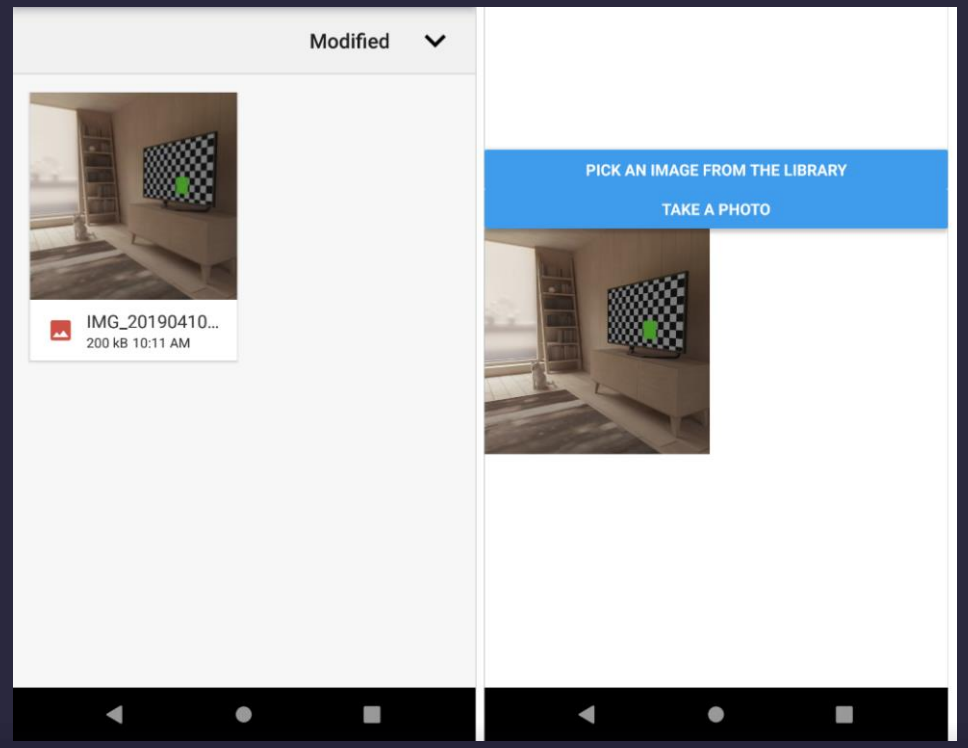

Jessey
Sign-in Successful
Welcome, Jessey
OK


Settings
Alt I
Show keyboard shortcuts
/
Show clipboard
Alt V
Show translate
Alt T
Show emoji
Alt .
Show on-screen keyboard
Alt K
Voice
Alt D
Jessey
Start Chatting
Choose a background color:


Jessey
12:48 PM
P
12:48 PM
N
12:48 PM
May 8, 2025
Palo Alto
Google
Mountain View
10:54 AM
Choose From Library
Take Picture
Send Location
Cancel

Page view-2


2:01
PICK AN IMAGE FROM THE LIBRARY
TAKE A PHOTO
GET MY LOCATION
2:01
Allow **Expo** to access this device's location?
DENY
ALLOW
2:31
PICK AN IMAGE FROM THE LIBRARY
TAKE A PHOTO
GET MY LOCATION
Palo Alto
Mountain View
Google


Modified
IMG_20190410...
200 kB 10:11 AM
PICK AN IMAGE FROM THE LIBRARY
TAKE A PHOTO

What Went Well

- Successfully set up the development environment using **React Native and Expo**, which made building and testing the app efficient.
- Designed a clean and functional **Start screen** using native components, following the provided UI guidelines.
- Integrated the **Gifted Chat** library to build a responsive and intuitive chat interface.
- Enabled **anonymous user authentication** with **Firebase**, making it easy for users to access the app without sign-up friction.
- Connected the app to **Cloud Firestore** to store and retrieve messages in real-time.
- Implemented **offline support** using AsyncStorage, allowing users to access previous messages without an internet connection.
- Allowed users to **send images** from their library or camera, and **share their current location** in a map view through the chat.

Key Learnings & Future Improvements

- I faced some challenges with **permissions handling** and syncing messages between local and cloud storage, but these helped me deepen my understanding of mobile app data flow.
- In the future, I plan to **optimize image uploads**, **enhance accessibility**, and **test across more devices**. I also want to implement full **user authentication** and deploy the app to **app stores** for real-world feedback.

Final Outcome

- ChatApp delivers a robust chat experience with real-time messaging, media sharing, offline support, and accessibility features. Its simple, intuitive design ensures ease of use, backed by Firebase's reliable infrastructure.

To view the code for this project, [click here]