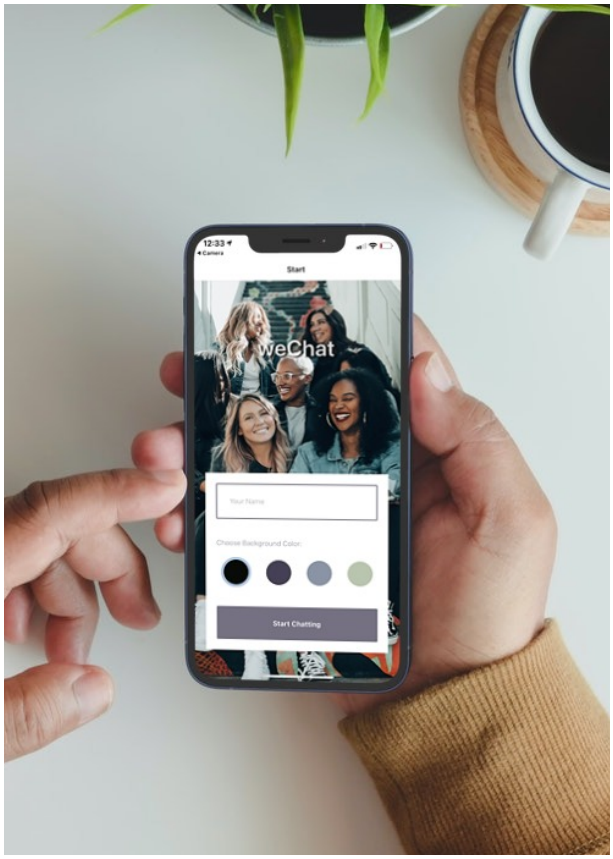
CASE STUDY: WECHAT



Overview

weChat is a messaging application for mobile devices built with Expo and React Native utilizing the popular Gifted Chat library. Messages are stored in Google Firebase and users will be able to look back through messages while offline.

Purpose & Context

weChat is a personal project I built as part of my web development course at CareerFoundry to demonstrate my mastery of React Native development skills. Mobile chat apps are among the most commonly downloaded and used apps in the world, so understanding how to build a chat app is an indispensable skill.

Objective

The aim of this project was to build a simple messaging application where users can send messages, pictures, and their location. Most importantly, this application can be used while offline when users want to look back through previously sent messages.

Role/Duration

My role was the Web Developer. This project took me about two weeks to finish and I was able to finish on time.

Technologies

- React Native
- Expo
- Google Firebase/Firestore Database
- Gifted Chat Library

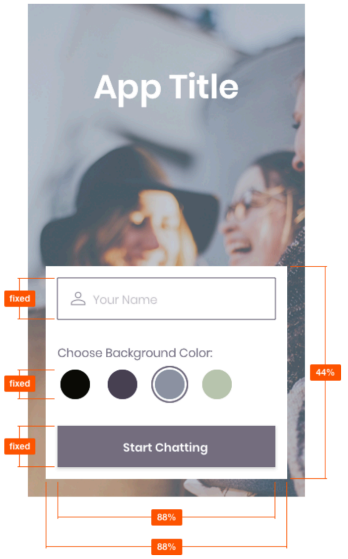
Approach

1. *Setting up Development Environment*

React Native was used to build a cross-platform native application (both iOS and Android) while being the most performant option as well. The main difference between React (a JavaScript library for creating user interfaces) and React Native (a JavaScript framework for

building native mobile apps) is that they target different platforms (browser vs mobile OS). This project also takes advantage of Expo which is an app that expands the developer environment by providing a means to view what you are building on your personal device and seeing the changes as you code.

Once I was able to set the environment up, the most time-consuming part of this project for me was creating the UI (shown to the right). While the brief provided an image for the UI, my job as a developer was to recreate it for my application. After completing the UI to the best of my ability, I made use of the Gifted Chat Library to provide me with the chat functionalities. This consisted of buttons, the chat-box, and the chat bubbles for my specific application.



Caption: Brief Example UI

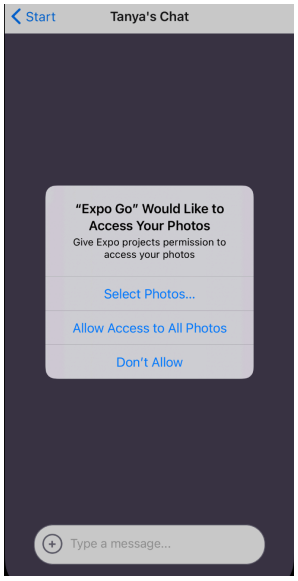
Caption: My finished product

2. Storing data on the Client-Side

After setting up the chat functionality using the Gifted Chat Library, Cloud Firestore (a tool that comes with Google Firebase) was used to handle the connection between clients and server, this also allows you to store data in its own database (a NoSQL database which means it stores data in documents).

I then set up anonymous authentications through Firebase, which means that each time you log in with your phone, it will identify you with a User ID based on your phone without the use of registering a profile. It will then read through the messages that you have stored in the database and load the messages onto the screen if there are any.

The application takes advantage of client-side storage or offline storage to save user-related data through a local storage system specifically for React Native - asyncStorage. Just as the name suggests, it provides a JavaScript API in which its methods are asynchronous or, in other words, do not block any other operations in my applications while waiting for a response.

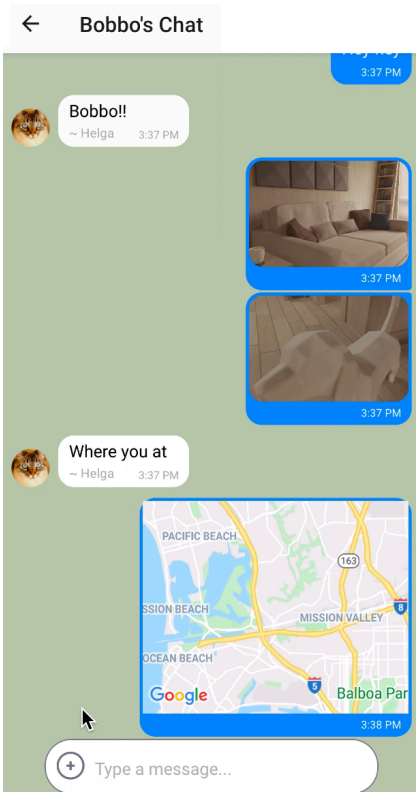


3. User Permissions

The final aspect of my project was to make sure that my application made use of user permissions when gaining access to the users' camera, photos, and location. An important part of developing applications is respecting their privacy, you must always think about how you're handling users' data. An important part of any messaging application is having access to a device's camera and the users' photos and location. So, I had to make sure that my application followed all ethical standards especially considering I want my users to know their privacy is of the utmost importance to me as a developer.

Final Reflections

I enjoyed this project because it was the first time I was able to directly code an application using a framework specifically for mobile devices. I ran into a lot of error messages, however, I found myself quickly able to find solutions as my process for handling errors within my code has been streamlined. Whereas in the past, I felt like they were terrifying roadblocks, now I understand it's a part of developing code and I look forward to finding a way to make it work.



Caption: My final chat interface.

I felt that creating the UI was a challenge for me because I wanted to make sure it had certain features for my users to experience. This led me to come up with some exciting new code that I will be sure to make use of in the future.

While working with local storage, there was one solution to a problem in which I had to make use of a less powerful alternative. I felt like I had to do this because the solutions I found were outside of my abilities. One such item was working with iOS cocoa pods when trying to solve the issue. Unfortunately, this was out of my realm of expertise and I had to fall back on an alternative solution.

Overall, I felt like I developed an efficient end product that met all of my goals for creating a messaging application with offline capabilities.