Jonathan Blokker

CareerFoundry Case study

Chat-App

Overview:

Chat-app is a Native App for mobile devices allowing users to chat but also share images and locations. Users will be able to add their name in the main screen along with selecting a background color for their main chat screen.

Purpose and Context:

This app was built to learn and develop React Native. With the completed project, my hope is to display my experience in JavaScript.

Objective and Challenges:

My end goal is to have a complete app that I can add to my portfolio. The challenges faced in this project were learning the framework React Native from scratch and creating a fully functional app.

Tools used

- React Native
- Expo
- Google Firestore Database
- Gifted Chat

Creation

This app took me roughly two weeks to create and I worked on it three to four hours every day. I began the process by reviewing basic knowledge of React Native to have a base understanding of the framework that I would be dealing with. After reading, I began by going to my development environment and downloading the template that needed to be used for this app. I downloaded it in my terminal using *expo init hello-world*. While still in the terminal I downloaded Expo to work alongside my project. This was done with *npm install expo-cli --global.* I used the GiftedChat interface to help create this project. Using it allowed me to use chat bubbles like you see in WhatsApp or Facebook. Now that I had all my downloads completed, I created a component folder and two files within this folder for the chat-app. One being the start screen and the other being the chat screen and I added my code. After I had both pages built, I then created an account with Google Firebase and linked it to my code so that my app can store messages. I finished this app with adding the features of location, and image sharing to make it complete. During the development process I used Expo to work out bugs and follow my developmental progress.

Want to see the app? Follow these instructions

Download
the repository and open it in your development environment.

Add dependencies
in your environment by opening your terminal and typing
npm install

Install Expo
Install expo in your terminal
npm install expo-cli -g

Run expo
in your terminal
Expo start

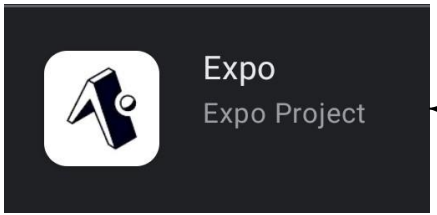
Scan the QR Code
After downloading the Expo app on your phone from the Google play store, scan the QR code on your computer.

You're done
The app will pull up on your phone for viewing

Hey, look here!
click this link to go to my repository. https://github.com/Jbblokker/Chat-app

How to install

1. First, you need to download the repository from Github. Now, open your downloaded file in your development environment. I used Visual Studio Code, but feel free to use what you're comfortable with.
2. Once the file is open, you will need to open your terminal and install all the dependencies of this app with *npm install*. When this is completely downloaded, you can continue with the next step.
3. You'll now want to download Expo, so you can view the chat-app on your mobile phone and your computer. You'll need to still be in your terminal and type *npm install expo-cli -g*. (We will go over installing it on your mobile device in just a minute.)
4. Once Expo is completely downloaded, you can start the application. You can do this in the terminal.
5. Type *expo start* and the Expo program will automatically begin and will pull up in your browser. Go ahead and go to this webpage. On this page you will see a QR-code on the bottom left hand side. You will scan this to deploy the Chat-app on your mobile device.
6. Before you can scan this QR-code, you need to download Expo on your mobile phone. You can find this in either the Apple store or in the Google Play Store. I have provided a photo of what it looks like in Google Play.
7. Once downloaded, open the app on your mobile device and you'll see at the top of the screen "Scan QR Code".
8. Scan the QR-code with your mobile device that is on your computer screen. Your screen will be redirected to the chat app.
9. There are no further steps but to enjoy the app and test the features!



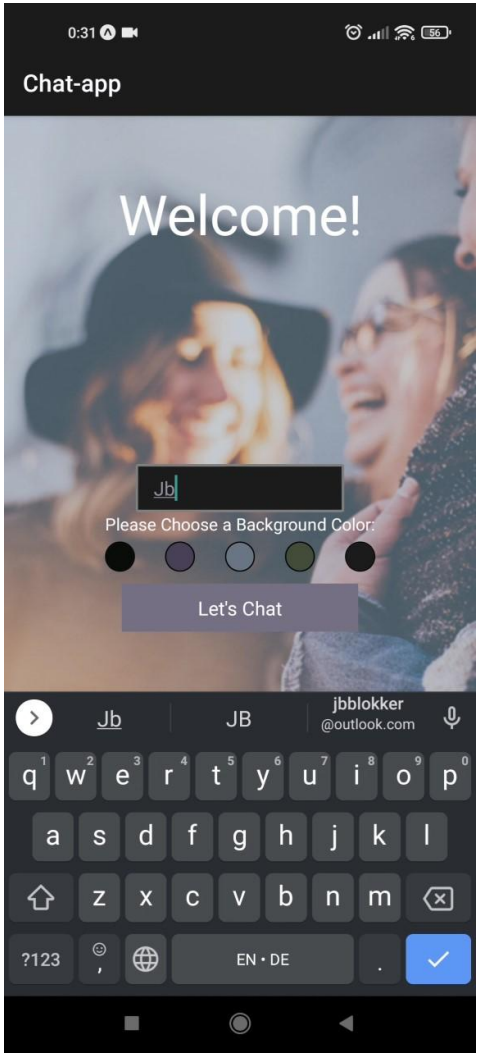
StyleSheets

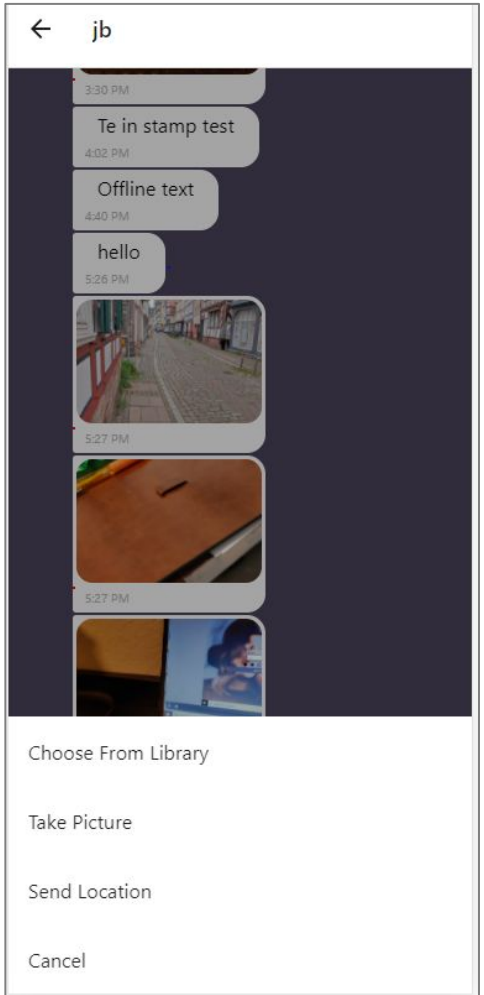
Unlike traditional websites, React Native uses no <CSS>. Though styling in React Native is similar, no separate CSS file is needed. You simply add it to the bottom of your page beneath the render. A key feature to pay attention to is that all components accept a prop named style. In addition, names are written in camel casing ‘exampleText ‘ instead of ‘example-text’.

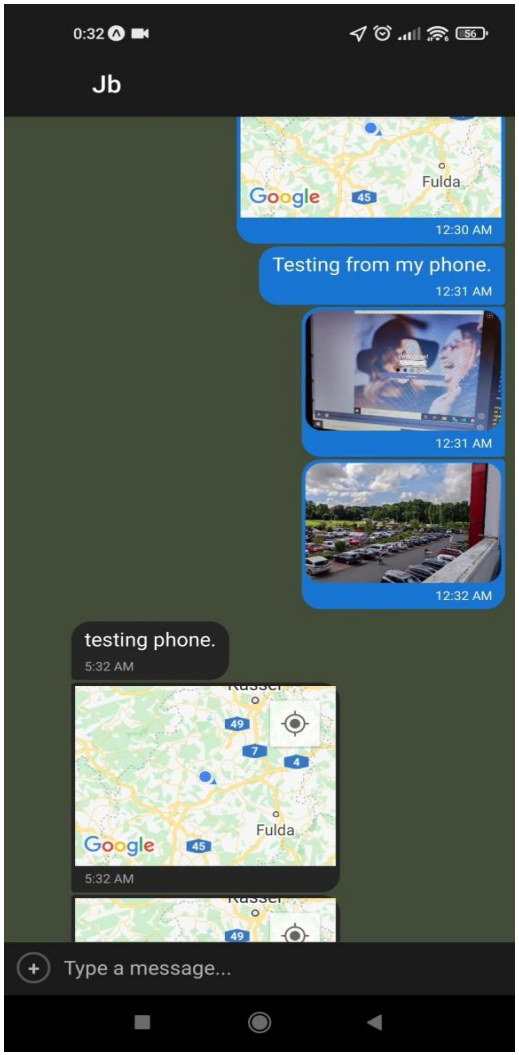
Though, at first, it seemed out of place to put all my styling beneath my render at the bottom of the page, the quick accessibility and continuation of camel casing made for an easy transition to adapting to this styling.

Want to learn more about stylesheets? Look here.
https://reactnative.dev/docs/style

Views of the App







Closing thoughts and challenges

The chat-app made with React Native was a good test of skill on being able to easily adapt to a new framework. I enjoyed learning the process of creating a mobile app and seeing the differences to regular web development.

One aspect I found challenging during this process of creating the chat-app was understanding the role of creating a fluid message system in componentDidMount. I initially struggled to clearly identify two users, which caused some bugs during development. Once I read more on the structure and looked at clear examples, I was able to clear up the issue.