Installation Instructions

Bus App. (Android only)

This install is for Windows to Android. For macOS, visit https://reactnative.dev/docs/environment-setup and follow the instructions for React Native CLI Quickstart download for macOS to Android.

To install the Bus App, download the files off the website.

(skip if you already have Android Studio downloaded)

Install Android Studio if you do not already have it. Make sure to check these boxes:

- Android SDK
- Android SDK Platform
- Performance (Intel ® HAXM)
- Android Virtual Device

When the installation has finished and you are looking at the Welcome screen, install the Android SDK. React Native apps require Android 9 (Pie) SDK. You can do this from the Welcome screen by clicking 'Configure' and then selecting "SDK Manager".

Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the Android 9 (Pie) entry, then make sure the following are checked:

- Android SDK Platform 28
- Intel x86 Atom_64 System Image or Google APIs Intel x86 Atom System Image

After this, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand "Android SDK Build-Tools" entry, then make sure that 28.0.3 is selected.

Lastly, click "Apply" to download and install the Android SDK and related build tools.

(pick up here if you already had Android Studio installed)

Create a new application by opening the command line, moving into your desired directory, and typing:

```
npx react-native init MyApp
```

If you are using a physical device, you can simply plug your device into your computer using a USB cable. (Make sure you have debugging mode enabled. Instructions for that are found here.)

Congrats! You have created your React Native app. Now, it's time to upload the files.

Open App.js in your text editor of choice (I used Visual Studio Code). Go to my website, download and open App.txt under the Bus App files section and copy and paste the code from that file into App.js in your text editor.

Now that you have the code, let's run the app!

Go back to the command line and type:

```
cd MyApp
```

```
npx react-native start
```

This will start the Metro Bundler. A window will pop up; leave the window open! Learn more about it here.

Next, to start the application run, type:

```
npx react-native run-android
```

Congrats! Your app should be running. If you are having problems, you can visit this link.

If you make changes, you can open the Metro Bundler window and simply type 'r' and this will reload your app.

Parent App. (Android and iPhone compatible)

Open the command line and type the following:

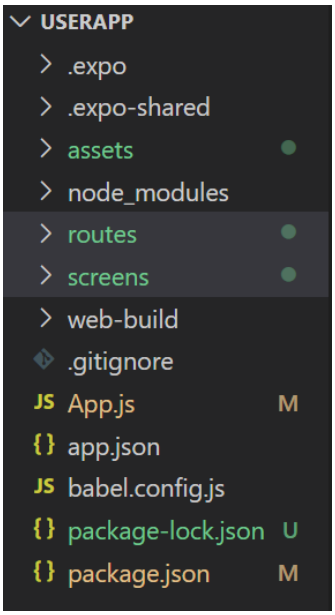
```
npm install -g expo-cli
```

Then, create a new project by running the following commands:

```
expo init MyApp
```

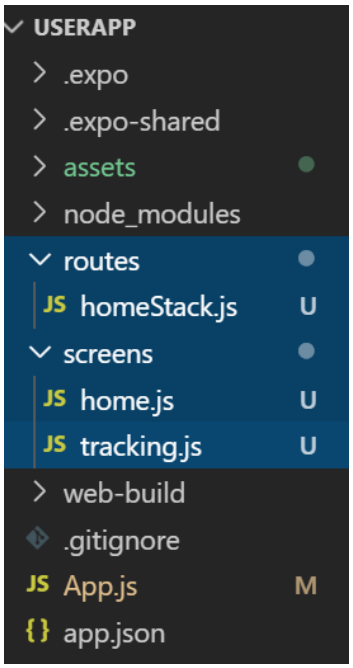
```
cd MyApp
```

Open the folder in your text editor (I recommend Visual Studio Code). Add two files: one called 'screens' and one called 'routes'.



Next, in the routes folder, create a file called homeStack.js. Open the homeStack.txt file from my website and copy and paste the code from the text file into homeStack.js.

Next, in the screens folder, create two files: one called 'home.js' and the other called 'tracking.js'. Open the home.txt and tracking.txt files from my website and copy and paste the code from those text files into their respective .js files.



Now, install the Expo client app on your mobile device with the following logo:

Run the app by running one of the following in the command line:

```
npm start
```

or

```
expo start
```

A window will pop up in your browser: scan the QR code from the browser on your mobile device. The app should start running!

If you make changes to any of the files, they should reload automatically once you save your changes.

That's it! If you have any questions or there is anything that wasn't clear, you can visit this link to help: https://reactnative.dev/docs/environment-setup

If you have a server and you want to run this with your own server, you can also upload the sendData.php and the corresponding .json files. In each app, you will have to change the server link that a fetch request is sent to. In the Parent App, these are in the getRouteData and getCurrentData functions. In the Bus App, this is in the sendDataToServer function.

The routeData.json and dataTest.json are two files that can be used to test! The routeData.json file contains a made-up route with some stops in it. The dataTest.json file contains some made-up bus location data and stops at each stop in the routeData.json file.