React Native and a GoPro

First off, let’s get react Native installed:

Let's use the **react-native-cli** package to build a new project:

npx react-native@latest init imagekitReactNative

Navigate to the project directory:

cd imagekitReactNative/

Start the metro server.

npx react-native start

And now to run the app in the iOS simulator (you should have Xcode installed)

npm run ios

Or to run the app in the android simulator (you should have android studio installed)

npm run android

You should see the “**Welcome to React Native**” screen. This means the sample app has been set up correctly.

GoPro Video

My GoPro has a Wifi Spot where I can send commands over curl and I experimented with my GoPro to switch to video mode.

For example:

```
# SWITCH TO VIDEO MODE
```

```
curl "http://10.5.5.9/camera/CM?t=$GOPROPASS&p=%00"
# START/STOP RECORDING
curl "http://10.5.5.9/bacpac/SH?t=$GOPROPASS&p=%01"
# DELETE ALL FILES FROM CAMERA
curl "http://10.5.5.9/camera/DA?t=$GOPROPASS"
```

It has a file server (Cherokee) where you can see all cam files:

```
# See files on your browser
open "http://10.5.5.9:8080/gp/gpMediaList"
```

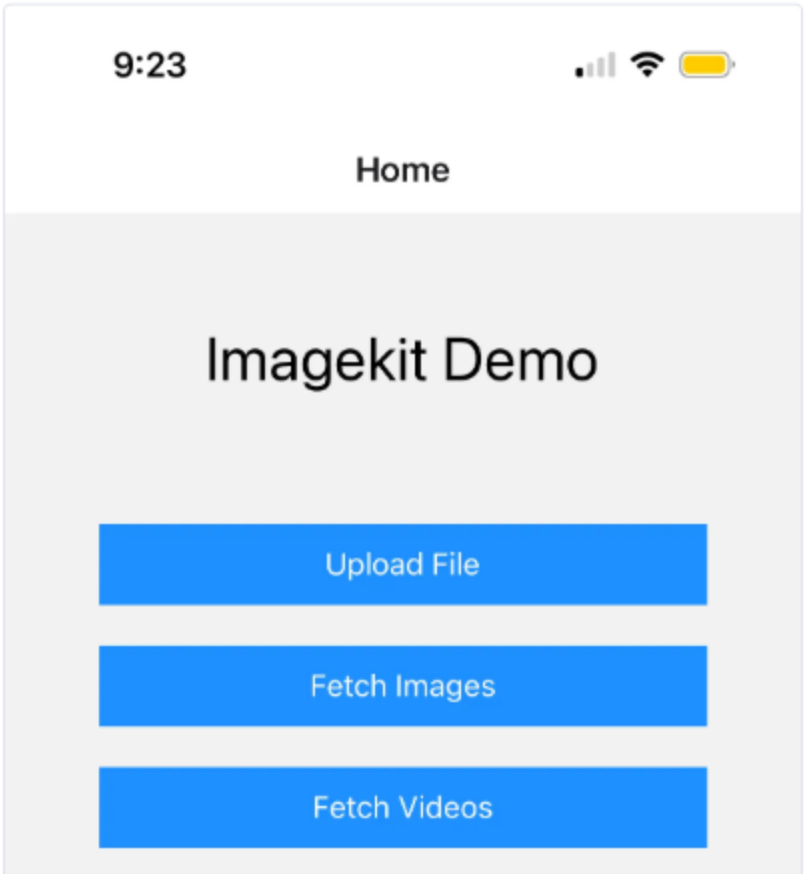
You can discover some neat tricks if you have VLC or ffmpeg on your mac:

```
# ENABLE STREAMING AND SEE LIVE VIDEO FROM VLC
curl -i "http://10.5.5.9/camera/PV?t=$GOPROPASS&p=%02"
vlc "http://10.5.5.9:8080/live/amba.m3u8"
# DOWNLOAD LAST 15 SECONDS OF SOME VIDEO ON YOUR GOPRO
ffmpeg -sseof -15 -i "$VIDEO_URL" -c copy output.mp4
```

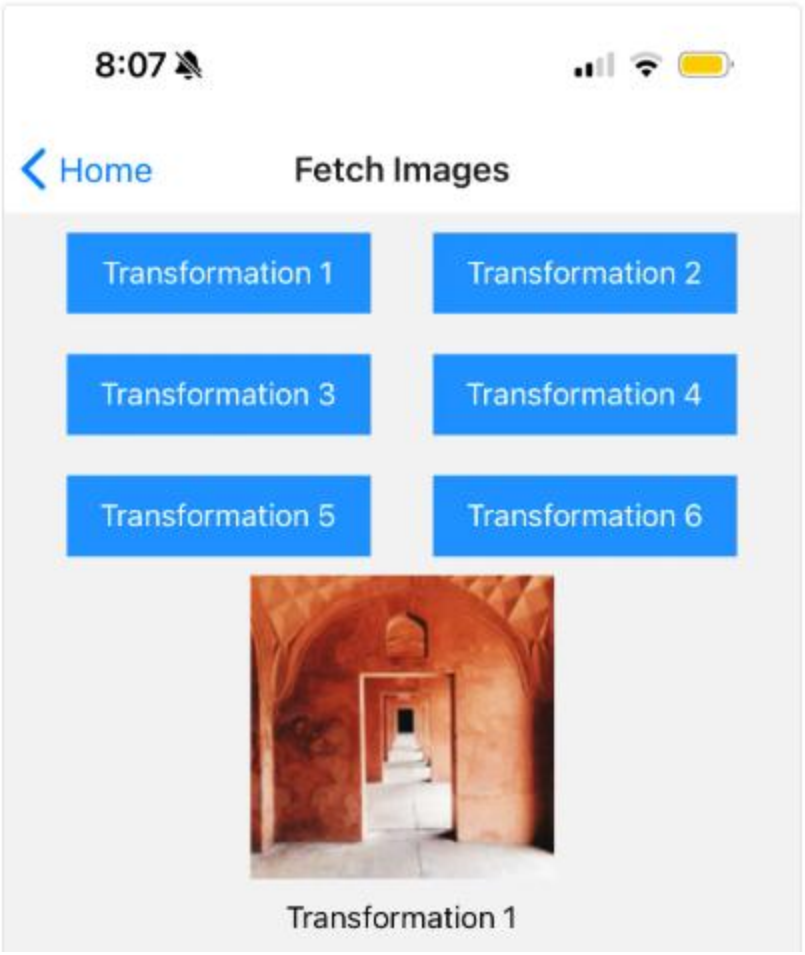
React Native is simple to use and I can use fetch API to execute most of the commands.

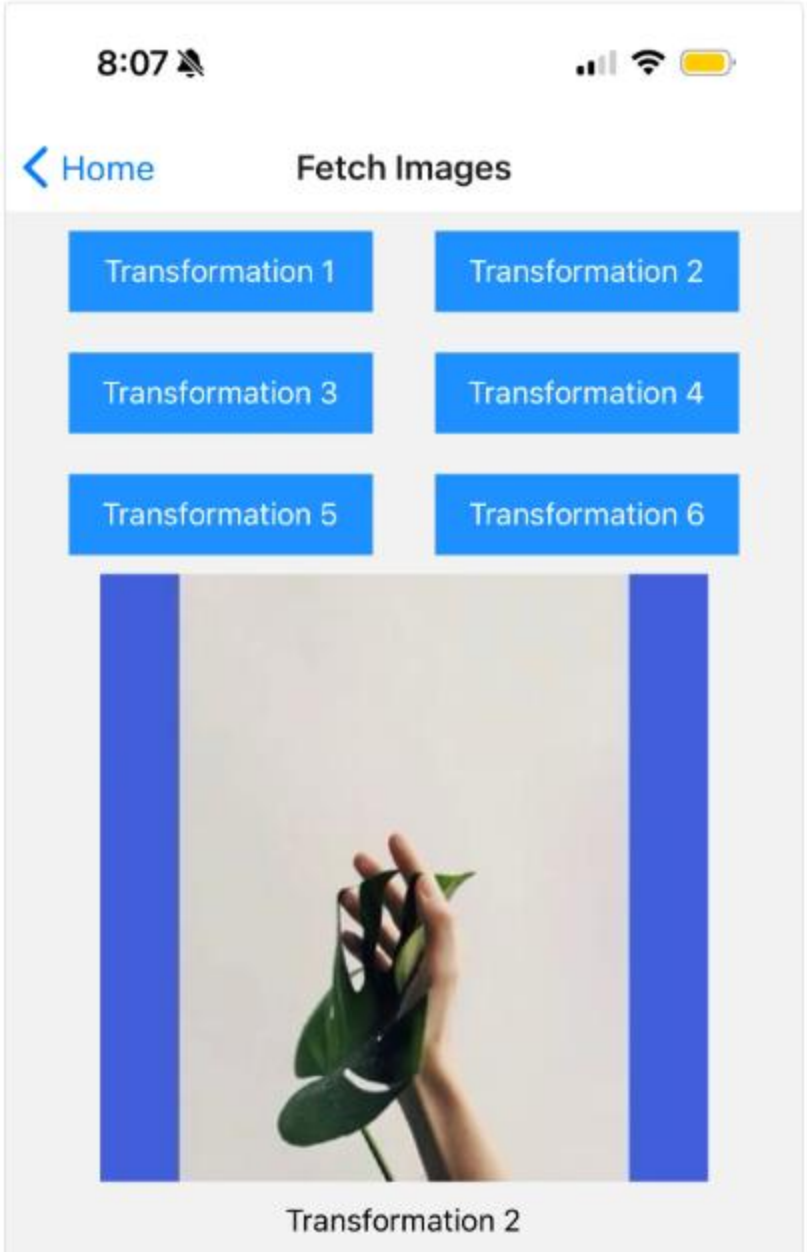
Examples of Reactive Native code

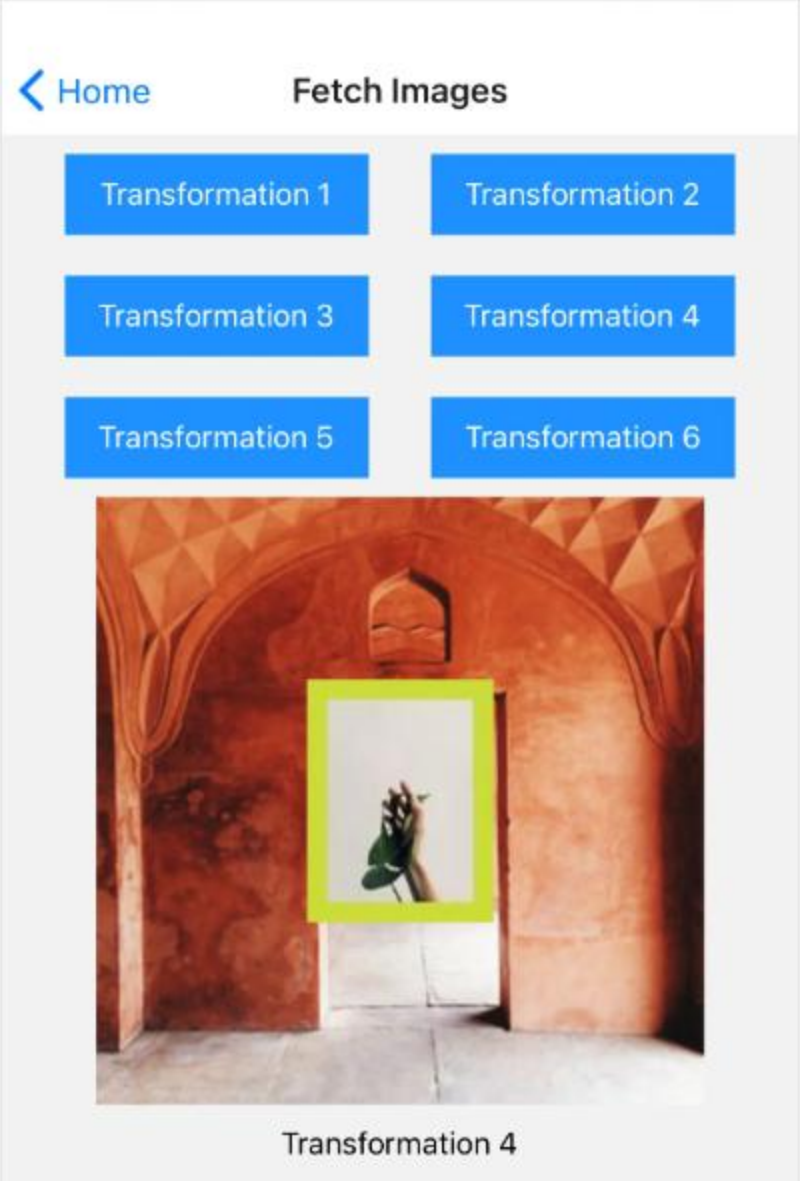
```jsx
import React from 'react';
import {View, Text} from 'react-native';
import Button from '../../components/Button/';
import getStyleSheet from './styles';

function Main({navigation}) {
  let styleSheet = getStyleSheet({});

  return (
    <>
      <View style={styleSheet.headContainer}>
        <Text style={styleSheet.text}>Imagekit Demo</Text>
      </View>
      <View style={styleSheet.btnContainer}>
        <View style={styleSheet.btnView}>
          <Button
            cssProps={styleSheet.buttonCssProps}
            onPress={() => navigation.navigate('Upload File')}>
            Upload File
          </Button>
        </View>
        <View style={styleSheet.btnView}>
          <Button
            cssProps={styleSheet.buttonCssProps}
            onPress={() => navigation.navigate('Fetch Images')}>
            Fetch Images
          </Button>
        </View>
        <View style={styleSheet.btnView}>
          <Button
            cssProps={styleSheet.buttonCssProps}
            onPress={() => navigation.navigate('Fetch Videos')}>
            Fetch Videos
          </Button>
        </View>
      </View>
    </>
  );
}

export default Main;
```


9:23
Home
Imagekit Demo
Upload File
Fetch Images
Fetch Videos

```
import React from 'react';
import {createStackNavigator} from '@react-navigation/stack';

import Main from './screens/Main';
import Fetch from './screens/Fetch';

const Stack = createStackNavigator();

function AppComponent() {
  return (
    <Stack.Navigator>
      <Stack.Screen name="Home" component={Main} />
      <Stack.Screen name="Fetch Images" component={Fetch} />
    </Stack.Navigator>
  );
}

export default AppComponent;
```


8:07
Home
Fetch Images
Transformation 1
Transformation 2
Transformation 3
Transformation 4
Transformation 5
Transformation 6
Transformation 1


8:07
Home
Fetch Images
Transformation 1
Transformation 2
Transformation 3
Transformation 4
Transformation 5
Transformation 6
Transformation 2



Experiment with Xcode and Cursor

I was not able to go far enough to complete the steps below, due to complexities with our business license and having Apple Business Identifiers during my initial setup, however, I did sign up for Apple Developer and I should have my DEV Identifier in a couple more days, to continue my project.

First create your project in Xcode, then open it in Cursor and index it. You can then edit your Swift code within Cursor,

using features like AI-powered assistance, quick documentation access, and smart code completion. Finally, apply your changes, refresh Xcode, and your modifications will be reflected in your Xcode project.

Here is an example of where I am at with Xcode: