Expo API
React Native UI Kits

CS571 – Mobile Application Development

Maharishi University of Management

Department of Computer Science

Associate Professor Asaad Saad

Maharishi International University - Fairfield, Iowa

What is Expo?

Expo is a framework/ platform for React applications providing a set of tools built around React Native and native platforms. It helps developers build, deploy, develop and quickly iterate on iOS, Android, and web apps from the same JavaScript/TypeScript codebase.

Expo is kind of an extension to ReactNative that offers a bunch of components and libraries already included to facilitate the development of mobile apps with React Native.

Advantages Of Using Expo

- Fast and simple project installation.
- Opens a convenient utility **Expo CLI** in a browser checking on the device it runs on, scan QR code, send the link via email to open the app in Expo Client.
- Develop from any OS, you just only need NodeJS and an internet connection.
- **Implicit Library linking** with react–native, linking is **not necessary** as everything is covered by Expo.

Advantages Of Using Expo

- **Expo Go client** is an app that is easily installed from the Google Play Store or Apple Store on your phone. It allows the opening of projects without build via XCode or Android Studio during the development process.
- Expo SDK offers the **collection of ready solutions**, such as working with the device accelerometer, camera, notifications, geolocation, etc.

Expo SDK

- Maps and Location
- Contacts
- Magnetometer
- Video, Audio and Fonts
- ImagePicker
- Camera

Refer to Expo SDK APIs for more details about installing and using each API: https://docs.expo.io/bare/unimodules-full-list/

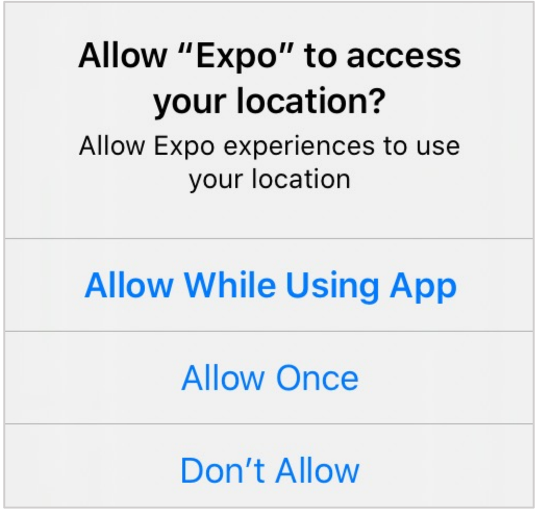
Other APIs from Expo SDK

- Push Notifications
- Calendar
- Gyroscope, Accelerometer, Pedometer
- Payments
- Facebook Login, Google Login
- Fingerprint Scanner
- SecureStore
- SMS

https://docs.expo.io/

App Permission

When it comes to adding functionality that can access potentially sensitive information on a user's device, such as their location, or possibly send them possibly unwanted push notifications, you will need to ask the user for their permission first.



Allow “Expo” to access your location?

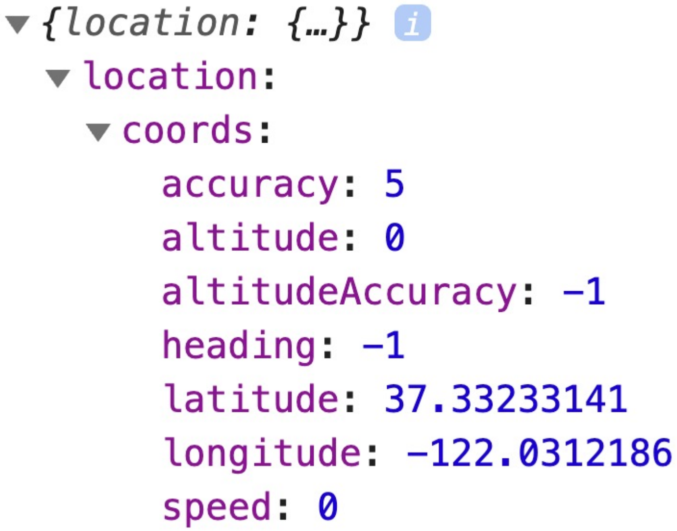
Allow Expo experiences to use your location

```
const { status } = await Location.requestPermissionsAsync();
const { status } = await Contacts.requestPermissionsAsync();
const { status } = await Camera.requestPermissionsAsync();
const { status } = await MediaLibrary.requestPermissionsAsync()
```

```
PermissionResponse {status, granted, canAskAgain}
status: 'granted', 'denied', 'undetermined'.
```

Location

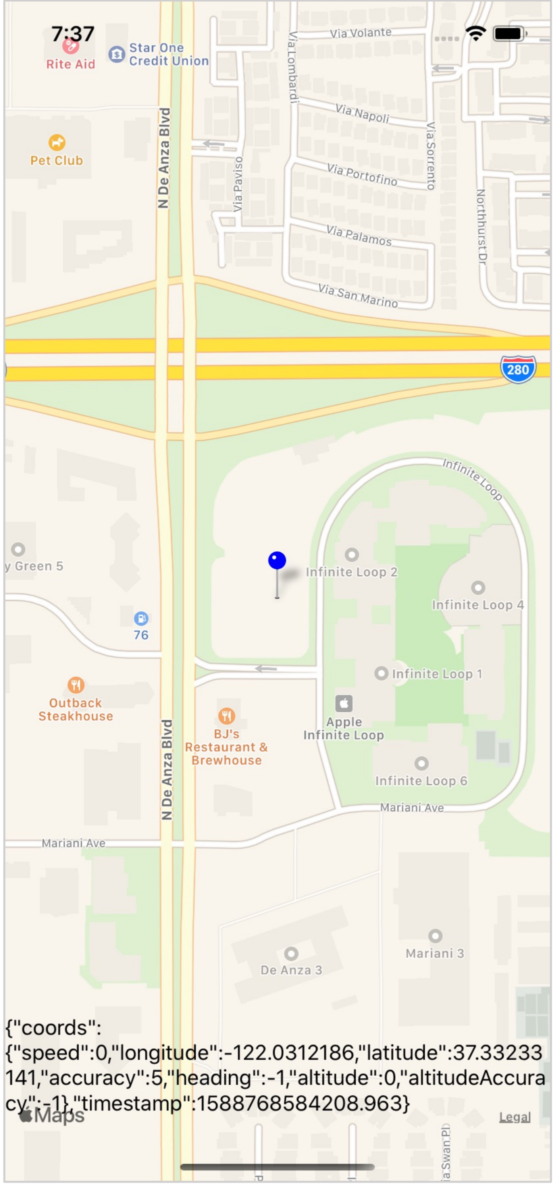
```
import * as Location from 'expo-location';

let location = await Location.getCurrentPositionAsync({});
const location = await Location.geocodeAsync('1000 N 4th Street, 52556');
const where = await Location.reverseGeocodeAsync(location.coords);
```



```
▼ {location: {…}} i
  ▼ location:
    ▼ coords:
        accuracy: 5
        altitude: 0
        altitudeAccuracy: -1
        heading: -1
        latitude: 37.33233141
        longitude: -122.0312186
        speed: 0
```

MapView and Marker

```
import MapView, { Marker } from 'react-native-maps';

<MapView
      initialRegion={{
        latitude: coords.latitude,
        longitude: coords.longtitude,
        latitudeDelta: 0.0922,
        longitudeDelta: 0.0421,
      }}
    >
      <Marker
        draggable
        coordinate={coords}
        title="MIU"
        description="Masters in Software Development"
        pinColor="blue"
      />
</MapView>
```



```
{"coords":
{"speed":0,"longitude":-122.0312186,"latitude":37.33233
141,"accuracy":5,"heading":-1,"altitude":0,"altitudeAccura
cy":-1},"timestamp":1588768584208.963}
```

MapView style

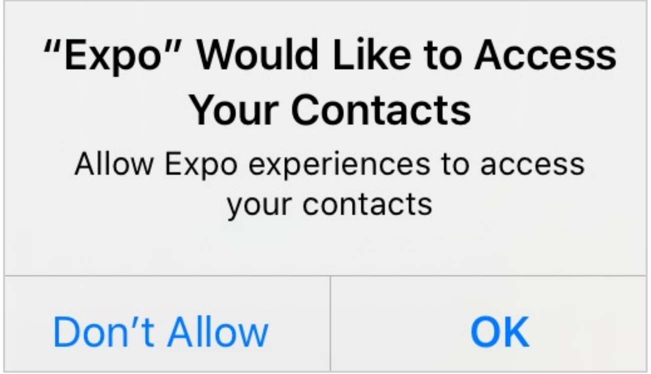
```
import { Dimensions, StyleSheet } from 'react-native';

StyleSheet.create({
  mapStyle: {
    width: Dimensions.get('window').width,
    height: Dimensions.get('window').height,
  },
});
```

Map Overlay

```
<>
   <MapView />
   <View style={{
        position: "absolute",
        bottom: 50,
        alignItems: 'center',
        justifyContent: 'center',
      }}>
        <Text>Overlay text</Text>
   </View>
</>
```

Contacts

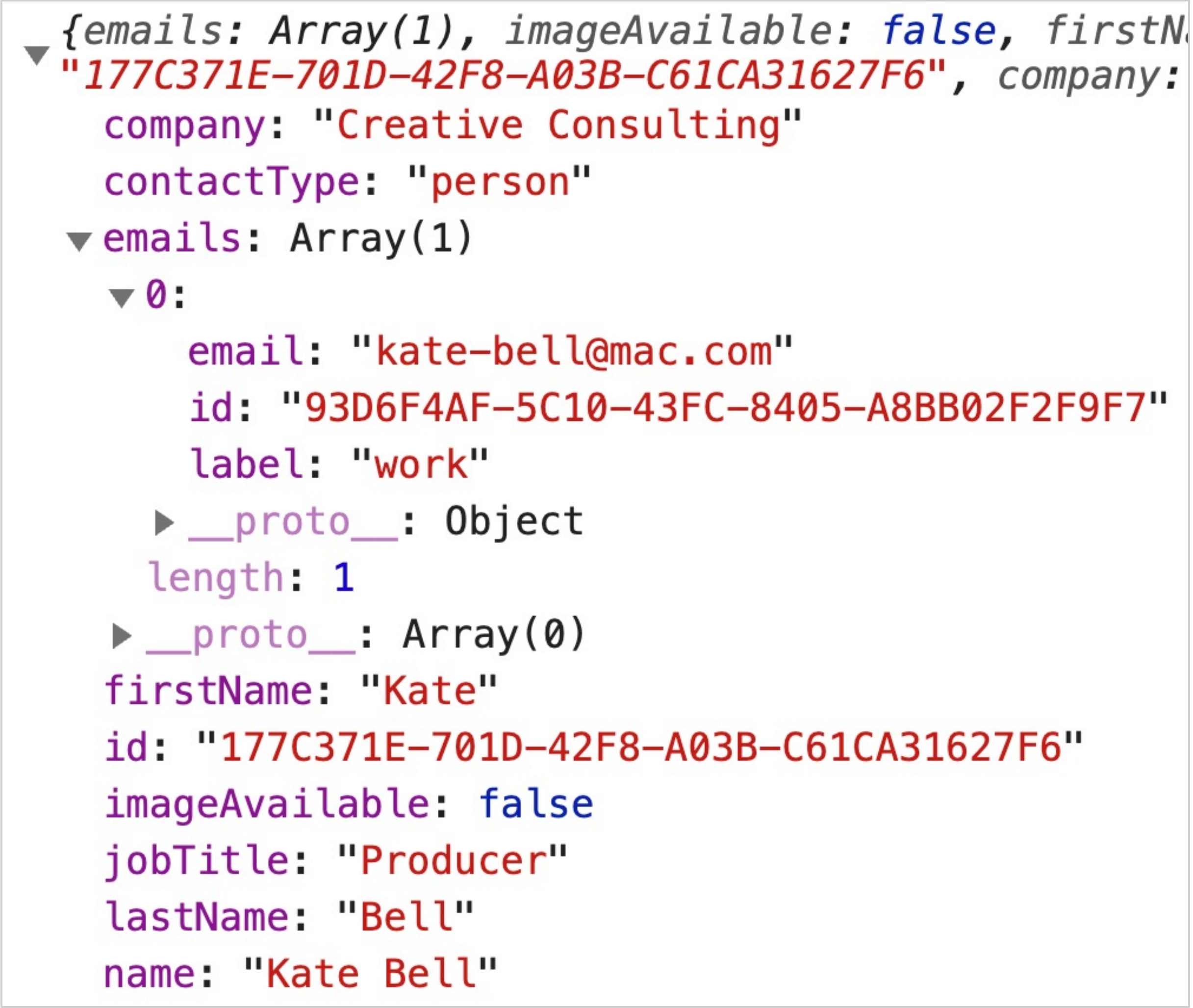
```
import * as Contacts from 'expo-contacts';

const {data} =
    await Contacts.getContactsAsync(
       {
          fields: [Contacts.Fields.Emails]
       });
```



“Expo” Would Like to Access Your Contacts

Allow Expo experiences to access your contacts

```
Contacts.requestPermissionsAsync();
```



```
▼ {emails: Array(1), imageAvailable: false, firstN
  "177C371E-701D-42F8-A03B-C61CA31627F6", company:
    company: "Creative Consulting"
    contactType: "person"
  ▼ emails: Array(1)
    ▼ 0:
        email: "kate-bell@mac.com"
        id: "93D6F4AF-5C10-43FC-8405-A8BB02F2F9F7"
        label: "work"
      ▶ __proto__: Object
      length: 1
    ▶ __proto__: Array(0)
    firstName: "Kate"
    id: "177C371E-701D-42F8-A03B-C61CA31627F6"
    imageAvailable: false
    jobTitle: "Producer"
    lastName: "Bell"
    name: "Kate Bell"
```

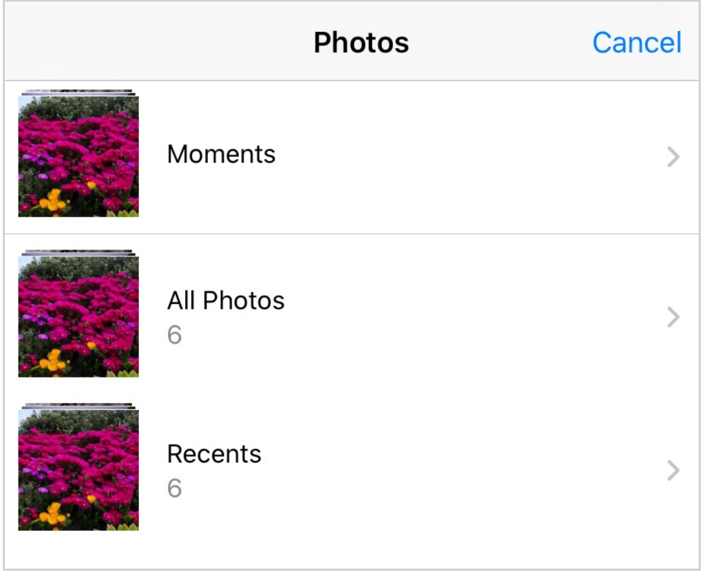
Magnetometer

```
import { Magnetometer } from 'expo-sensors';
export default class App extends React.Component {
  state = {
    MagnetometerData: { x:0, y:0, z:0 },
  };

  componentDidMount() {
    this._subscription = Magnetometer.addListener(result => {
      this.setState({ MagnetometerData: result });
    });
  }

  componentWillUnmount() {
    this._subscription && this._subscription.remove();
    this._subscription = null;
  }
}
```

ImagePicker

```
import * as ImagePicker from 'expo-image-picker';

state = { image: null };

<Button title="Pick an image from camera roll" onPress={handlePick} />
{state.image && <Image source={{ uri: state.image }} }

handlePick = async () => {
        let result = await ImagePicker.launchImageLibraryAsync({
           mediaTypes: ImagePicker.MediaTypeOptions.All,
           aspect: [4, 3],
           quality: 1,
        });
        if (!result.cancelled) this.setState({ image: result.uri });
};
```

ImagePicker



Camera

```
import { Camera } from 'expo-camera';
let camera = null;


<Camera
     style={{ flex: 1 }}
     type={Camera.Constants.Type.back}
     ratio="16:9"
     ref={ref => { camera = ref }}>

let photo = await camera.takePictureAsync();
```

photo is an object in memory, the object has reference to where the pic is temporary saved **photo.uri**

FileSystem

```
import * as FileSystem from 'expo-file-system';

let photo = await camera.takePictureAsync();

const timestamp = Date.now();
const to = `${FileSystem.documentDirectory}photos/Photo_${timestamp}.jpg`

FileSystem.copyAsync({
   from: photo.uri,
   to: to
})
```

MediaLibrary

It provides access to the user's media library, allowing to access the existing images and videos from the app, as well as save new ones.

```
import * as MediaLibrary from 'expo-media-library';

MediaLibrary.saveToLibraryAsync(photo.uri);
```

Reasons to Eject

- You want to add an NPM package to your project which requires a **react–native link**. If a third-party package states that you need to run the command react-native link as part of the setup process, then this package can't be used with expo.
- You are native oriented (Java/Kotlin/Swift/Objective C), and want to **integrate native module** into your React–native app but couldn't find a way to do it in the expo.
- You want to add a push notification library like **OneSignal** in your app and the library does not have expo integration.

Reasons Not to Eject

- Expo can **build binaries for you in the cloud**. If you eject, we can't automatically build for you any more.
- **Not comfortable writing native code**. Ejected apps will require you to manage Xcode and Android Studio projects.
- Painless React Native **upgrades** that come with Expo.
- You require **Expo's** push notification **services**. After ejecting, since Expo no longer manages your push credentials, you'll need to manage your own push notification pipeline.

npm run eject

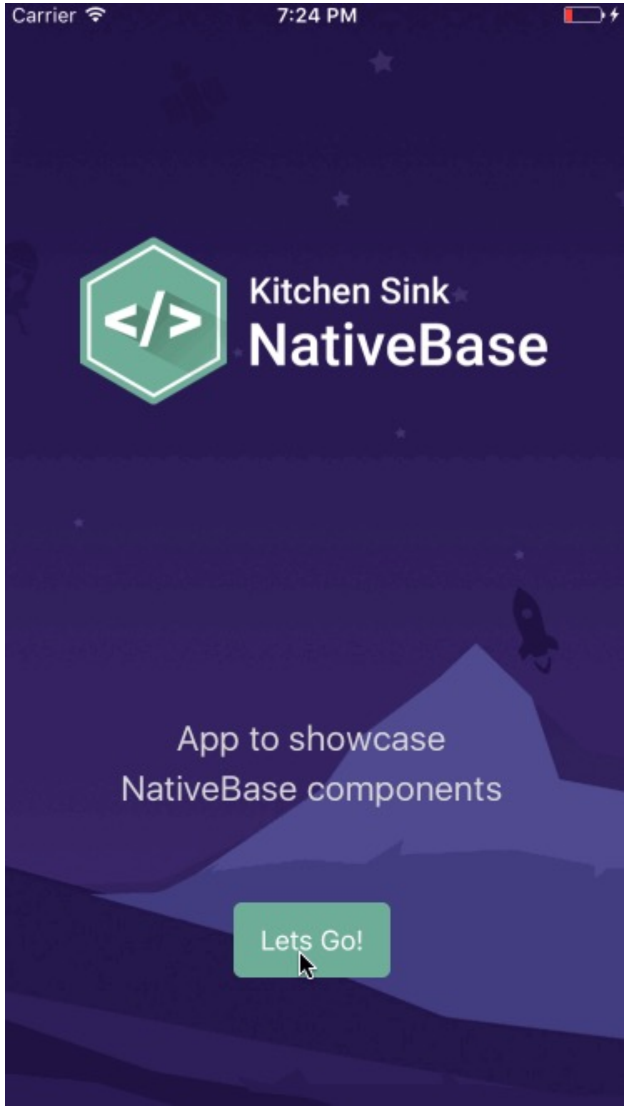
```
? How would you like to eject your app? (Use arrow keys)
❯ Bare: I'd like a bare React Native project.
  Cancel: I'll continue with my current project structure.
```

```
How would you like to eject from create-react-native-app?
   React Native: I’d like a regular React Native project.
What should your app appear as on a user’s home screen?
What should your Android Studio and Xcode projects be called?
```

Once you run the eject process, you're permanently ejected from Expo, you'll have native iOS (**`./ios`**) and Android (**`./android`**) projects and you will need to build on your computer.

React Native Elements

- Cross-Platform consistent design across android, iOS.
- Built completely in JavaScript. With Expo support.
- Community driven, 100% built by the community.



Expo Friendly

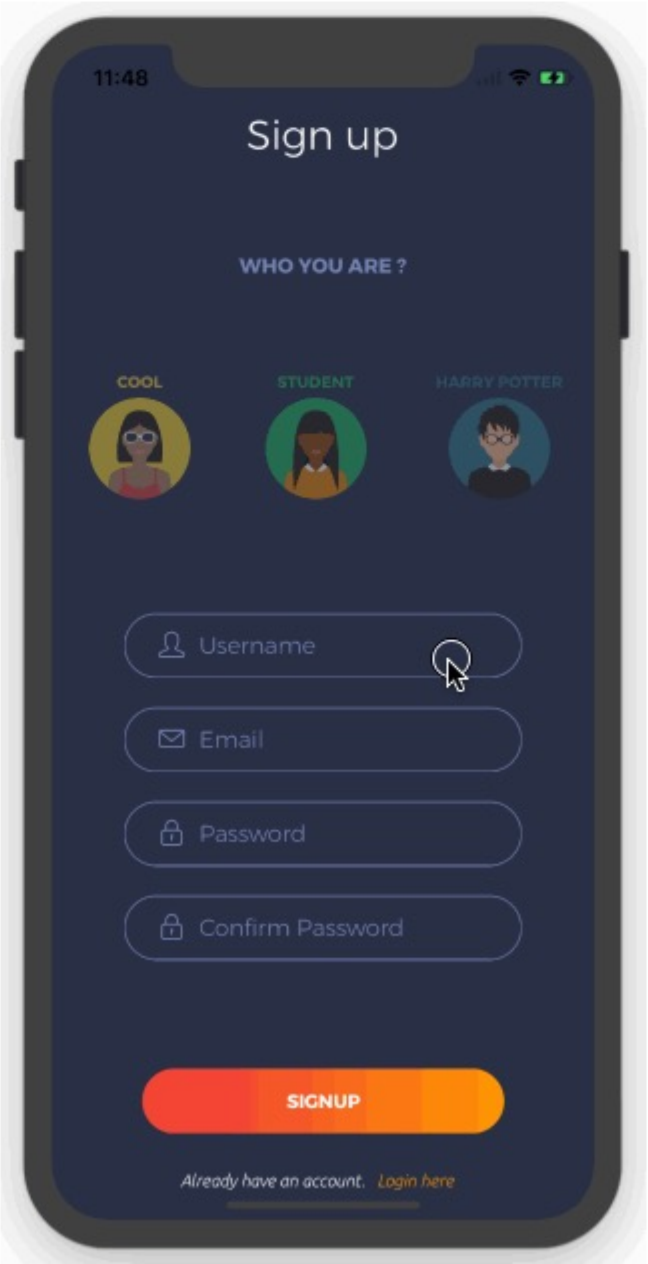
https://react-native-elements.github.io/react-native-elements/

NativeBase

NativeBase is a free and open source UI component library for React Native to build native mobile apps for iOS and Android platforms.

Can be integrated into Expo project.

Expo Friendly



https://docs.nativebase.io/

Nachos UI

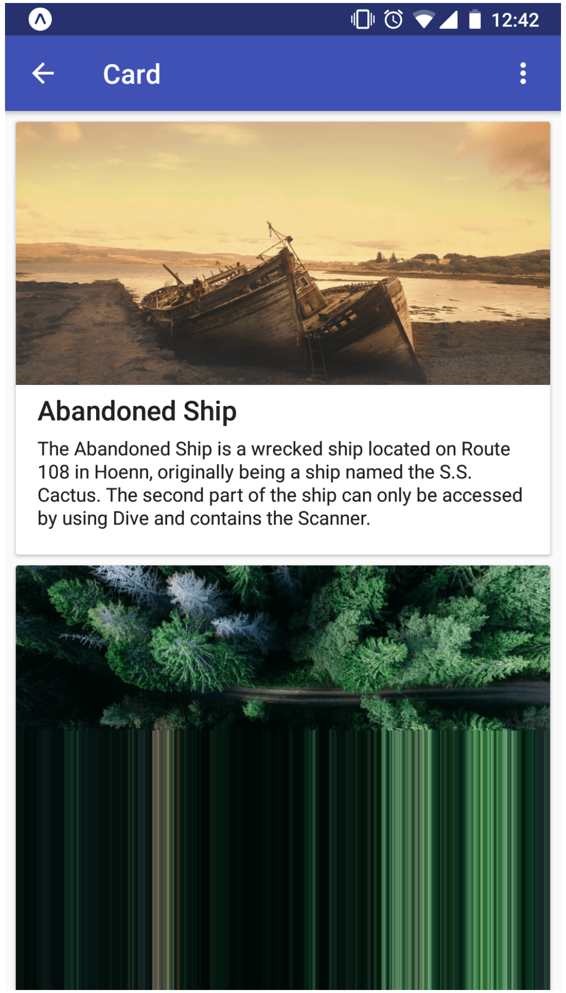
Nachos UI is a React Native component library with over 30 customizable components.

Expo Friendly

https://github.com/nachos-ui/nachos-ui

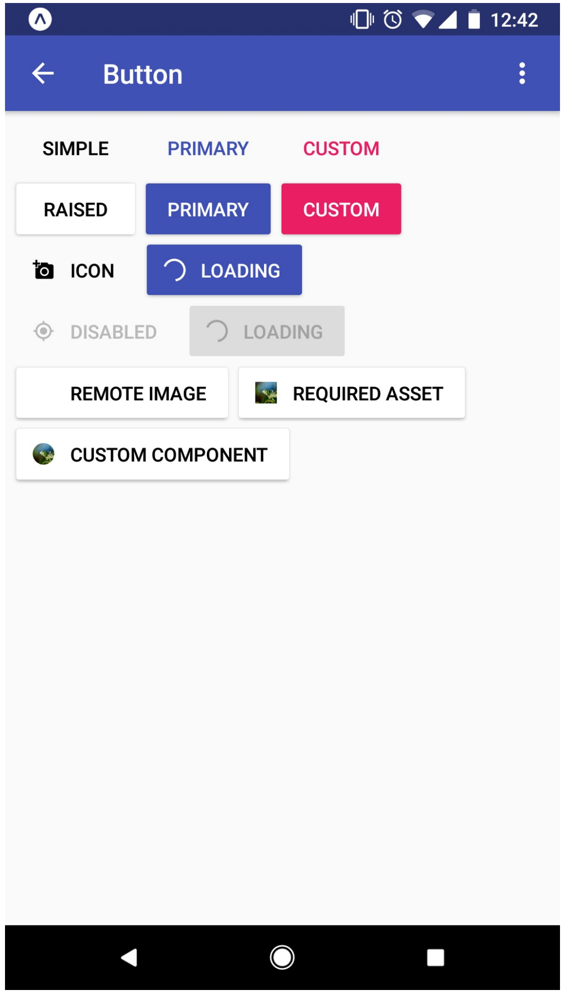
React Native Paper

React Native Paper is a cross-platform UI component library which follows the material design guidelines, with global theming support and an optional babel-plugin to reduce bundle-size.



Abandoned Ship

The Abandoned Ship is a wrecked ship located on Route 108 in Hoenn, originally being a ship named the S.S. Cactus. The second part of the ship can only be accessed by using Dive and contains the Scanner.



Expo Friendly

https://reactnativepaper.com/

Other UI Kits

- **React Native UI *Kitten*** *(https://akveo.github.io/react-native-ui-kitten/)*
- **React Native Material Kit** *(https://rnmk.xinthink.com/)*
- **React Native UI Library** *(https://wix.github.io/react-native-ui-lib/)*
- **Teaset** *(https://github.com/rilyu/teaset)*

Expo Friendly