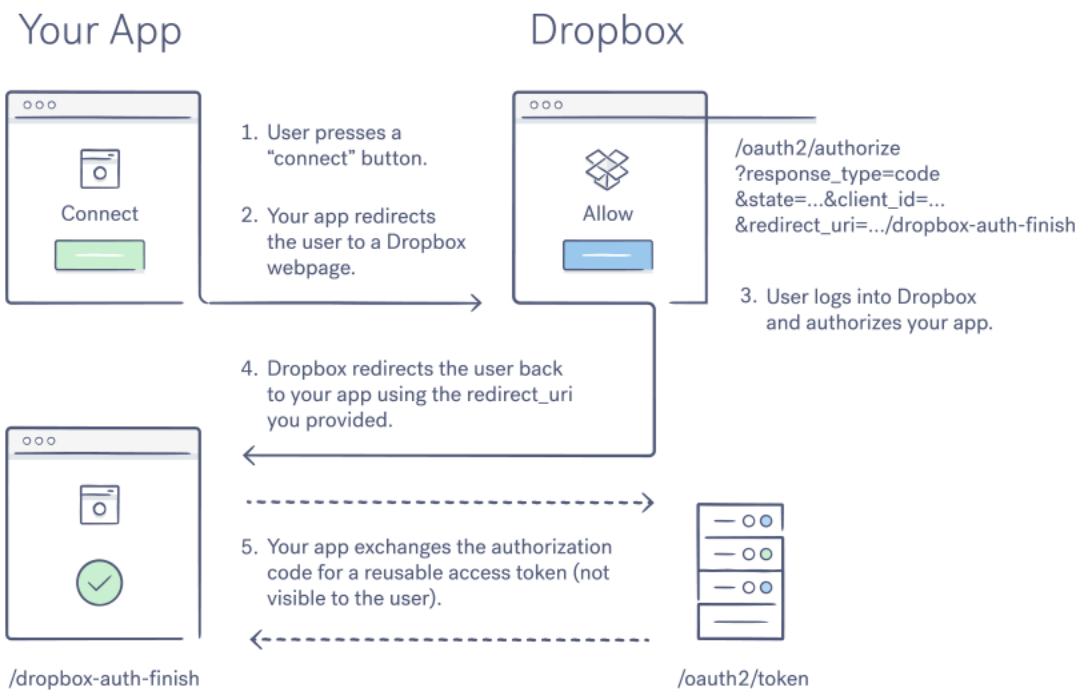
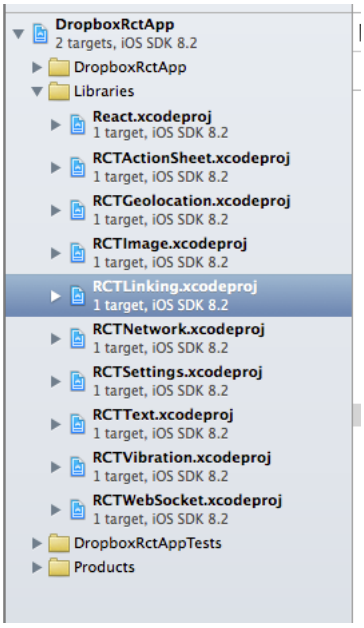
Dropbox POC (React Native + OAuth2 + Dropbox)

1. Dropbox has a good explanation about the authentication using OAuth2 (https://www.dropbox.com/developers/reference/oauthguide).

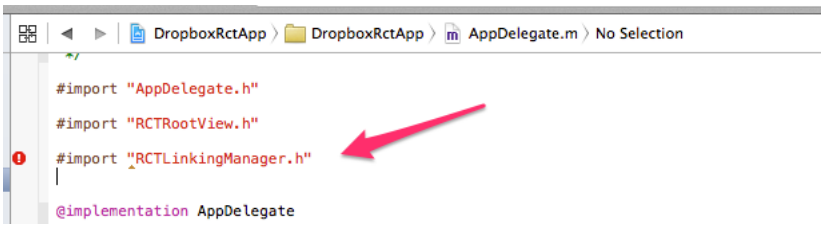


2. command "react-native init DropboxRctApp" to create a react native app
3. open DropboxRctApp.xcodeproj in Xcode. We will use React's LinkingIOS library to got to the OAuth2 authorization page and handle the redirect.

- Link the LinkingIOS library to our app

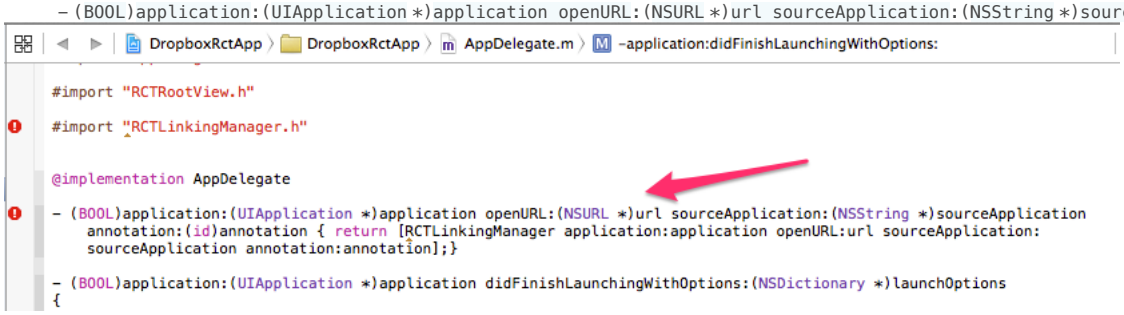


- Add #import "RCTLinkingManager.h" to the top of AppDelegate.m file



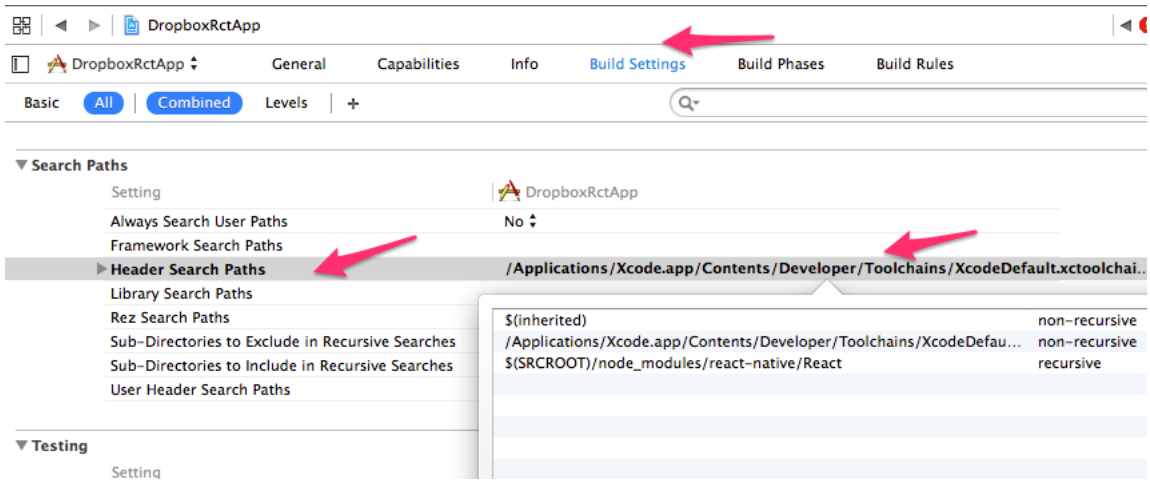
- Since we want to listen to incoming app links during our app's execution, we need to add following code to our AppDelegate.m file, right under

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceAppli
```

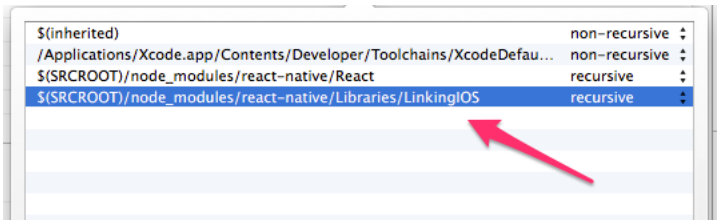


4. Notice there are two errors in the AppDelegate.m code. We need to change build setting to eliminate these errors.

- Go to the Build Setting and find out the Search Header and open the value



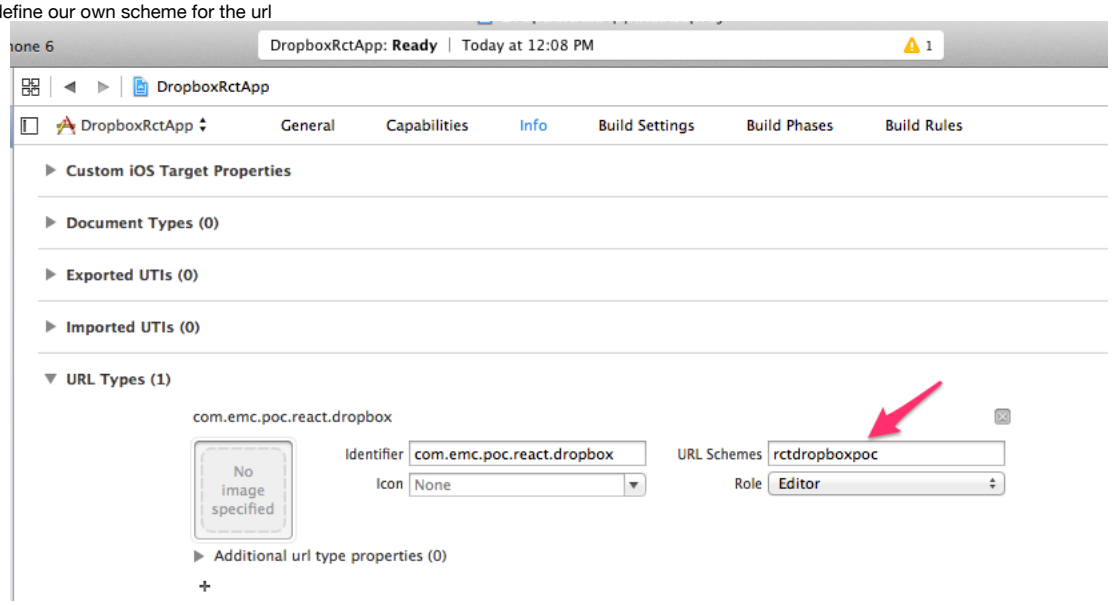
- Add new entry for the LinkingIOS and set it to be recursive



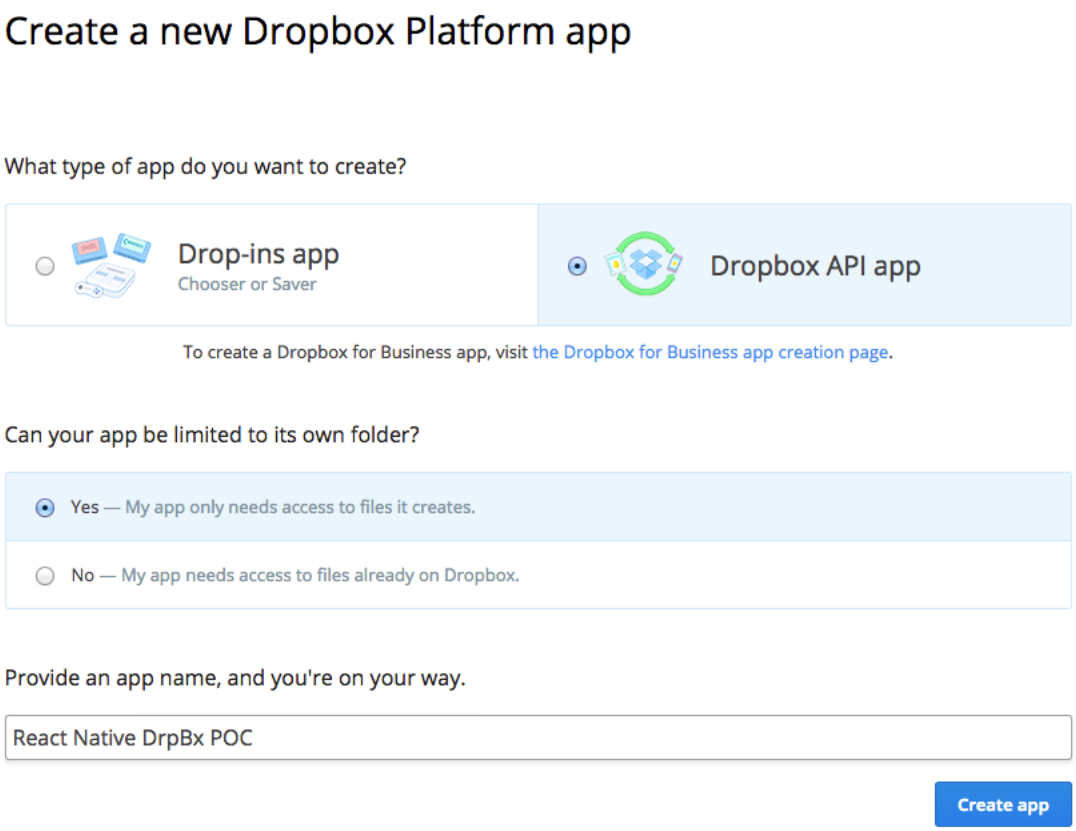
- The errors have gone.

```
DropboxRctApp 〉 DropboxRctApp 〉 AppDelegate.m 〉 -application:didFinishLaunchingWithOptions:

#import "AppDelegate.h"

#import "RCTRootView.h"

#import "RCTLinkingManager.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication
    annotation:(id)annotation { return [RCTLinkingManager application:application openURL:url sourceApplication:
    sourceApplication annotation:annotation];}

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
```

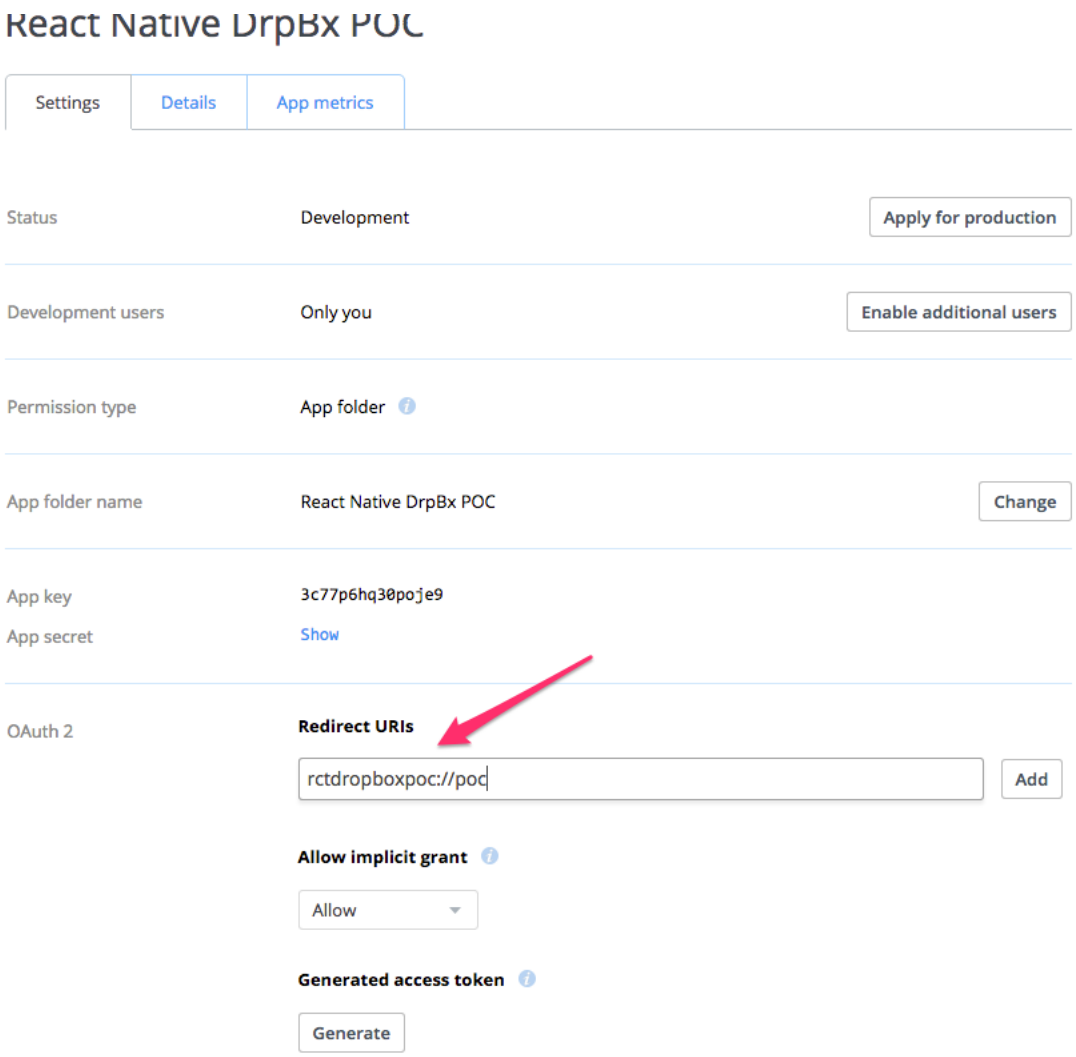
5. We can define our own scheme for the url



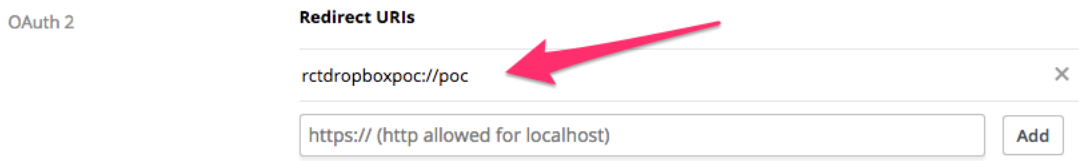
6. We need to create a new Dropbox platform app. Go to https://www.dropbox.com/developers/apps and click the “Create App” button. Create it with



7. After the app being created, setup the redirect URL wight the scheme you defined in your app.



- Don't forget to click "Add" button after giving redirect URL



8. Also, make a file config.js in the root of your React Native project called config.js. Add the app key from the screen above.

```
module.exports = {
  app_key: '3c77p6hq30poje9'
}
```

9. coding in index.ios.js

- add config and shittyQs (install it by run "nom install shitty-qs" in the root directory of our app. Also declare the LinkingIOS and TouchableHighli

```
'use strict';

var React = require('react-native');
var config = require('./config.js');
var shittyQs = require('shitty-qs');

var {
  AppRegistry,
  StyleSheet,
  Text,
  View,
  TouchableHighlight,
  LinkingIOS
} = React;
```

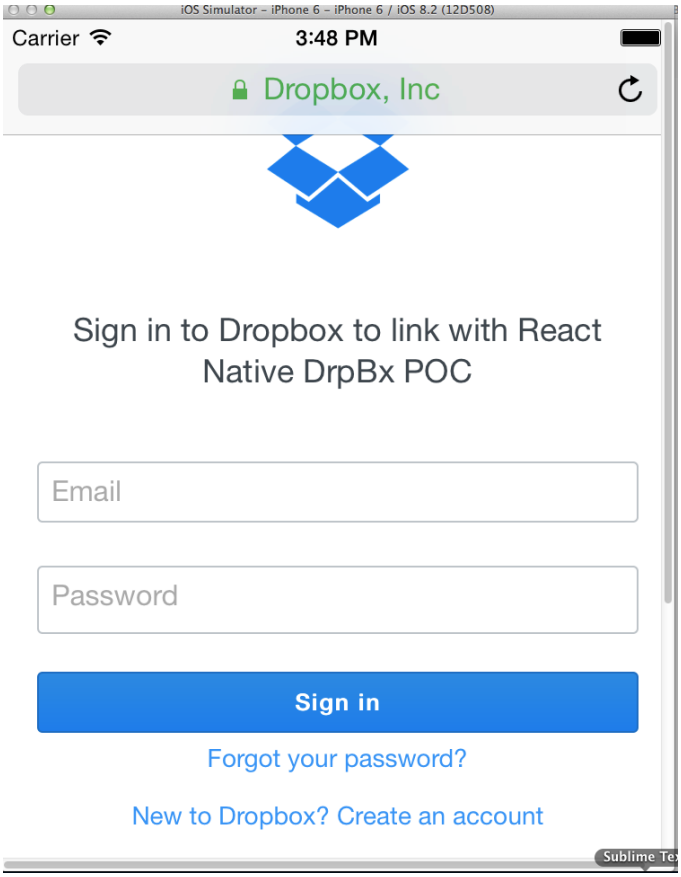
- present the Dropbox OAuth2 approval page in our app by creating a function dropboxOAuth2 and call it in ComponentWillMount()

```
function dropboxOAuth2 (appKey, callback) {

  LinkingIOS.openURL([
    'https://www.dropbox.com/1/oauth2/authorize',
    '?response_type=token',
    '&client_id=',
    appKey,
    '&redirect_uri=oauth2poc://zpoc'
  ].join(''));
}
```

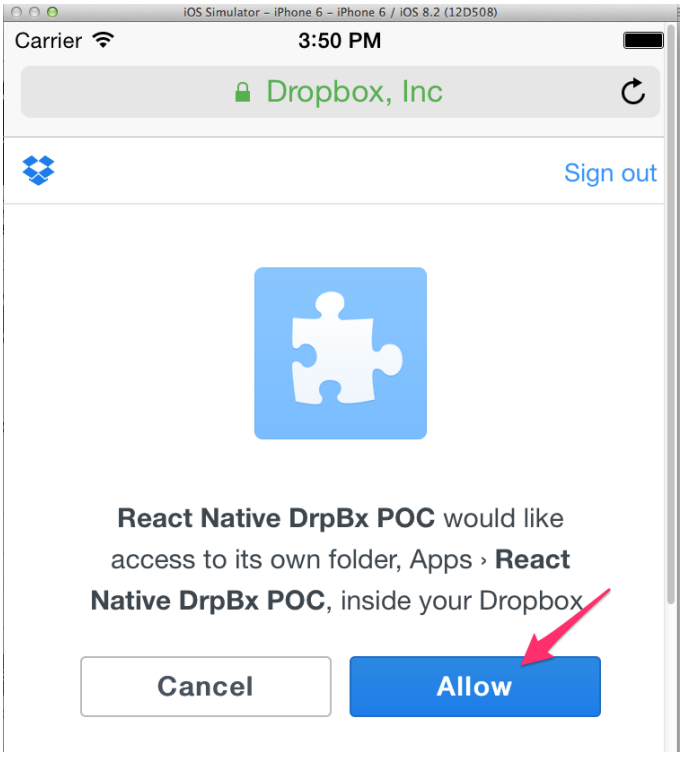
- At this point when the app starts up, it should immediately take you to Dropbox's Oauth page, which will then redirect you back to the app.
- Now we need to set up a listener using LinkingIOS so that we can get the access token that we need to make API calls.
  - add following code into our dropboxOAuth2 method. Be aware that the listener is removed after the url being handled.

```
function dropboxOAuth2 (appKey) {

  LinkingIOS.addEventListener('url', handleUrl);

  function handleUrl (event) {
    console.log(event.url);
    LinkingIOS.removeEventListener('url', handleUrl);
  }

  LinkingIOS.openURL([
    'https://www.dropbox.com/1/oauth2/authorize',
    '?response_type=token',
    '&client_id=',
    appKey,
    '&redirect_uri=rctdropboxpoc://poc'
  ].join(''));
}
```

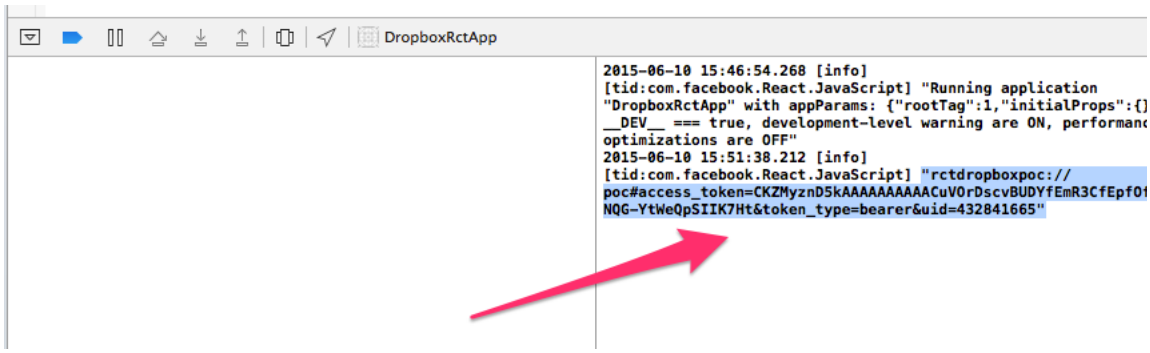
- build and run the app from Xcode, in the simulator you will see the Dropbox login page



- Click “Allow”, you will be redirected to you app home page



- From Xcode console, you can find the url like "`rctdropboxpoc://poc#access_token=CKZMyznD5kAAAAAAAAAACuVOrDscvBUDYfEmR3CfEpfOf3JNQG-YtWeQpS`



- We need to extract access_token value from the query string. Here we leverage the shitty-qs. We add a callback function and have it saved the

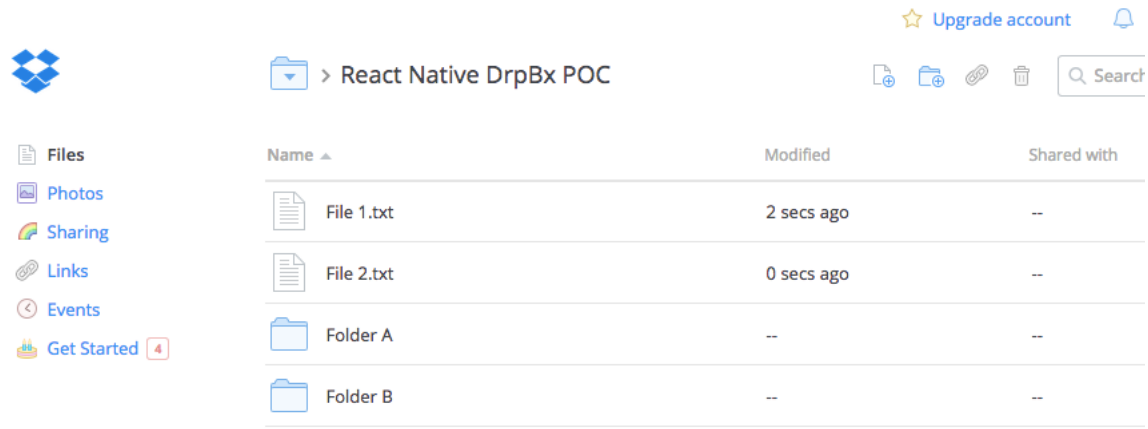
```
function dropboxOAuth2 (appKey, callback) {

  LinkingIOS.addEventListener('url', handleUrl);

  function handleUrl (event) {
    console.log(event.url);
    var [, query_string] = event.url.match(/\#(.*)/);
    var query = shittyQs(query_string);
    callback(null, query.access_token, query.uid);
    LinkingIOS.removeEventListener('url', handleUrl);
  }

  LinkingIOS.openURL([
    'https://www.dropbox.com/1/oauth2/authorize',
    '?response_type=token',
    '&client_id=',
    appKey,
    '&redirect_uri=rctdropboxpoc://poc'
  ].join(''));
}
```

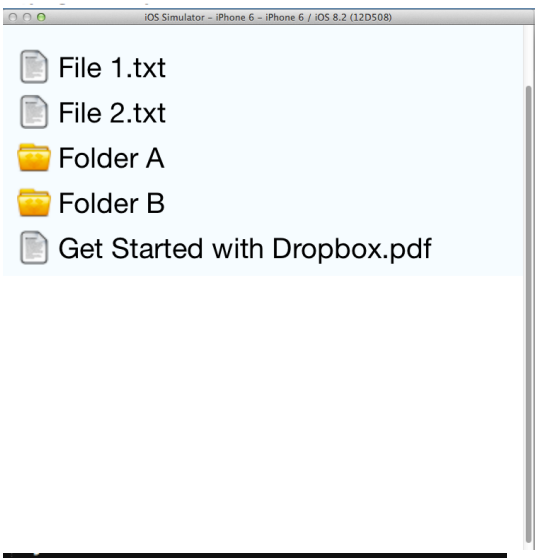
```
var DropboxRctApp = React.createClass({
  componentDidMount: function () {
    dropboxOAuth2(config.app_key, (err, accessToken) => {
      if (err) {
        console.log(err);
      }
      this.setState({ access_token: accessToken });
    });
  },
```

- We can show the access token in the app.

```
  render: function() {
    return (
      <View style={styles.container}>
        <Text style={styles.instructions}>
          Access Token: {this.state && this.state.access_token}
        </Text>
      </View>
    );
  }
```

- Let's add a ListView to list all the contents under your Dropbox app






```
return (
  <View tyle={styles.topContainer}>
    <View style={styles.outerContainer}>
      <ListView
        dataSource={this.state.dataSource}
        renderRow={this.renderContent}
        style={styles.listView}
      />
  </View>
);
```

The ListView has a dataSource which points to the state.dataSource. As long as this value changed, the ListView will render itself with ne

```
fetchContents: function(accessToken) {
  var authorizationString = 'Bearer ' + accessToken;
  fetch(
    'https://api.dropbox.com/1/metadata/auto/', {
      method: 'GET',
      headers: {
        'Authorization': authorizationString
      }
    })
    .then((response) => response.json())
    .then((responseData) => {
      this.setState({
        dataSource: this.state.dataSource.cloneWithRows(responseData.contents),
        access_token: accessToken
      });
    })
    .done();
},
```

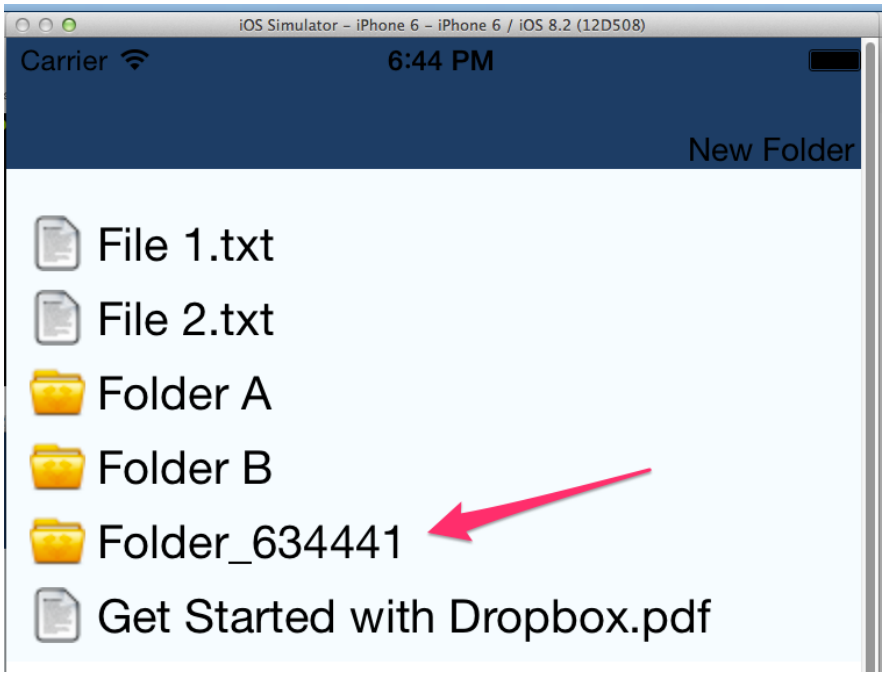
Be aware that the Bearer token is being set into the header (which is one of OAuth2 authorization mechanism). Upon the request success

- Add a create new folder action to illustrate the POST request operation. A TouchableHeighlight component is being added on top of the list view

```
render: function() {
  if (!this.state.access_token) {
    return this.renderLoadingView();
  }
  return (
    <View tyle={styles.topContainer}>
      <TouchableHighlight
        onPress={this.onNewFolderPressed}
        style={styles.touchableContainer}>
        <Text>New Folder</Text>
      </TouchableHighlight>
      <View style={styles.outerContainer}>
        <ListView
          dataSource={this.state.dataSource}
          renderRow={this.renderContent}
          style={styles.listView}
        />
      </View>
    </View>
  );
}
```

The onPress handler is being defined as below. Notice that in the callback function, we call fetchContents() explicitly since the response data only con
automatically. We're adding the access token in the header and posting to the create_folder endpoint. The option root determines where the folder will

```
var DropboxRctApp = React.createClass({
  onNewFolderPressed: function() {
    var authorizeString = 'Bearer ' + (this.state && this.state.access_token);
    var path = 'Folder_' + Math.random().toString().substring(12);
    fetch(
      'https://api.dropbox.com/1/fileops/create_folder', {
        method: 'POST',
        headers: {
          'Authorization': authorizeString
        },
        body: 'root=auto&path=' + path
      }
    ).then(function() {
      this.fetchContents(this.state.access_token);
    }.bind(this));
  },
```



- There's actually a big security issue with this implementation. It might be possible for an attacker to send a URL to our app, containing their acc

```
function dropboxOAuth2 (appKey, callback) {
  var state = Math.random() + '';
  LinkingIOS.addEventListener('url', handleUrl);

  function handleUrl (event) {
    console.log(event.url);
    var [, query_string] = event.url.match(/\#(.*)/);
    var query = shittyQs(query_string);
    if (state === query.state) {
      callback(null, query.access_token, query.uid);
    }
    else {
      callback(new Error('Oauth2 security error'));
    }
    LinkingIOS.removeEventListener('url', handleUrl);
  }

  LinkingIOS.openURL([
    'https://www.dropbox.com/1/oauth2/authorize',
    '?response_type=token',
    '&client_id=',
    appKey,
    '&redirect_uri=rctdropboxpoc://poc',
    '&state=',
    state
  ].join(''));
}
```

- This is the file of full index.ios.js Generic File

10. build and run in the device

- Follows the instruction of React Native, make device and develop machine in the same network also change the localhost to real ip, the app stil

```
 * `inet` value under `en0:`) and make sure your computer and iOS device are
 * on the same Wi-Fi network.
 */

jsCodeLocation = [NSURL URLWithString:@"http://192.168.1.19:8081/index.ios.bundle"];

/**
 * OPTION 2
 * Load from pre-bundled file on disk. To re-generate the static bundle
 * from the root of your project directory, run
```

- Using Option 2, uncomment the following line, the build from command line react-native bundle —minify. In the Xcode, archive it and deploy it t

```
/**
 * OPTION 2
 * Load from pre-bundled file on disk. To re-generate the static bundle
 * from the root of your project directory, run
 *
 * $ react-native bundle --minify
 *
 * see http://facebook.github.io/react-native/docs/runningondevice.html
 */

jsCodeLocation = [[NSBundle mainBundle] URLForResource:@"main" withExtension:@"jsbundle"];
```