

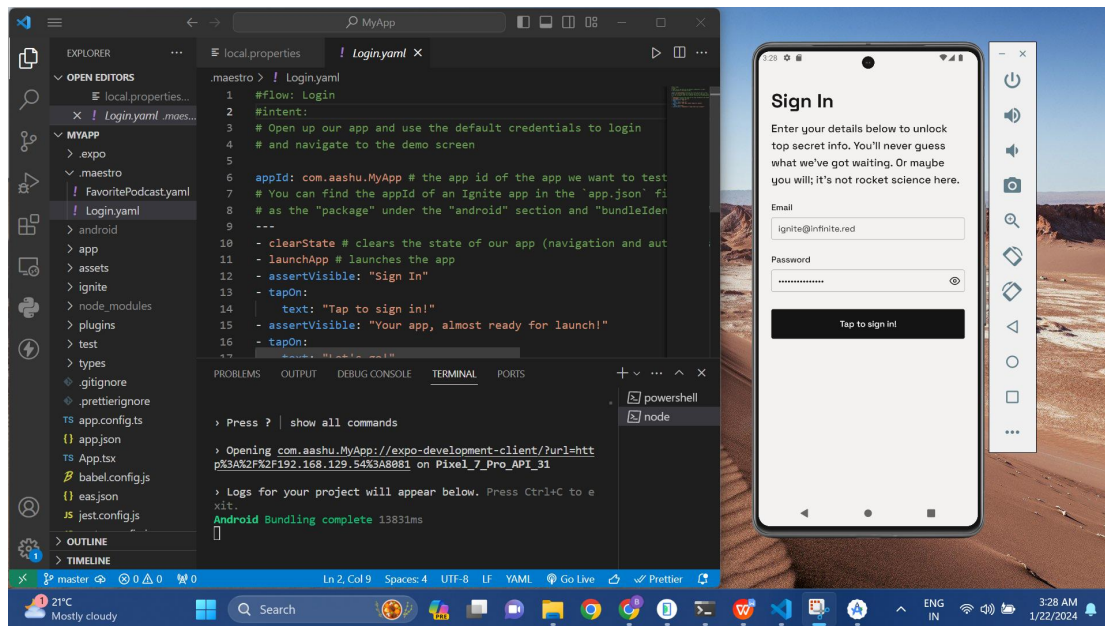
# Project

## End-to-End Testing of React Native Application using Maestro

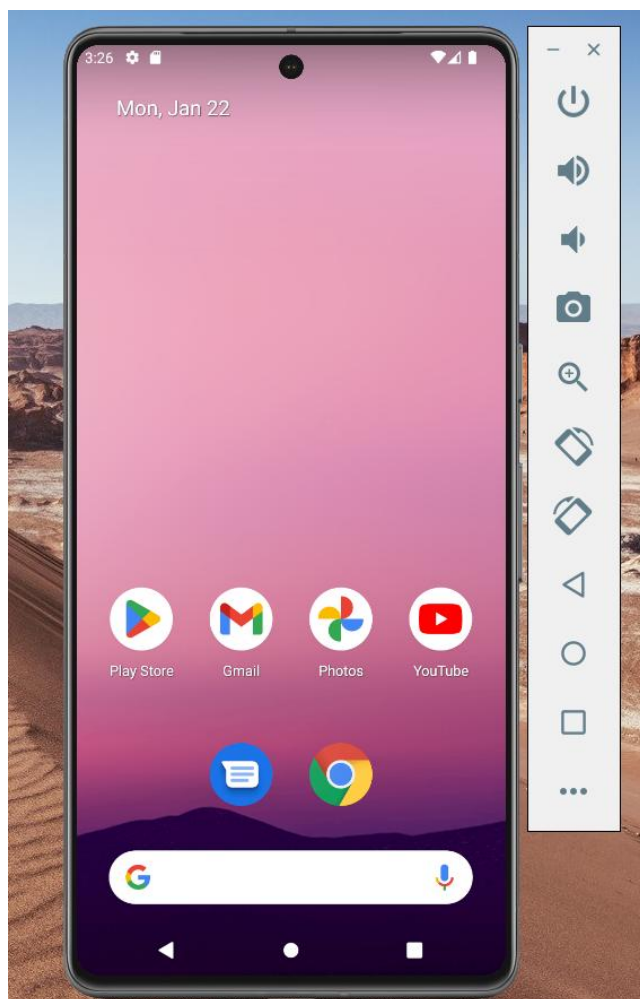
### Tech required

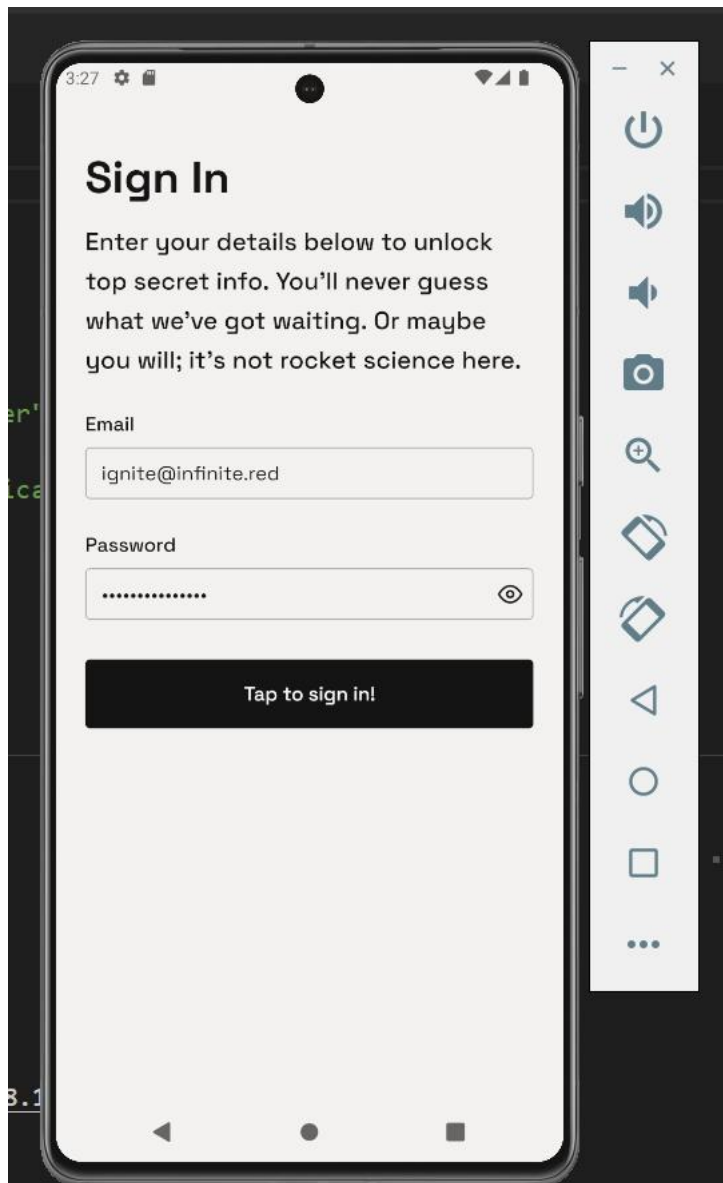
Nothing makes it into Ignite unless it's been proven on projects that Infinite Red works on. Ignite apps include the following rock-solid technical decisions out of the box:

Library	Category	Version	Description
React Native	Mobile Framework	v0.72	The best cross-platform mobile framework
React	UI Framework	v18	The most popular UI framework in the world
TypeScript	Language	v5	Static typechecking
React Navigation	Navigation	v6	Performant and consistent navigation framework
MobX-State-Tree	State Management	v5	Observable state tree
MobX-React-Lite	React Integration	v3	Re-render React performantly
Expo	SDK	v49	Allows (optional) Expo modules
Expo Font	Custom Fonts	v11	Import custom fonts
Expo Localization	Internationalization	v14	i18n support (including RTL!)
Expo Status Bar	Status Bar Library	v1	Status bar support
RN Reanimated	Animations	v3	Beautiful and performant animations
AsyncStorage	Persistence	v1	State persistence
apisauce	REST client	v2	Communicate with back-end
Reactotron RN	Inspector/Debugger	v3	JS debugging
Hermes	JS engine		Fine-tuned JS engine for RN
Jest	Test Runner	v26	Standard test runner for JS apps
Maestro	Testing Framework		Automate end-to-end UI testing
date-fns	Date library	v2	Excellent date library
FlashList	FlatList replacement	v1	A performant drop-in replacement for FlatList



## Step 2: Set Up the Environment





Using Powershell And Ubutnu:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\KIIT> $Env:ANDROID_ADB_SERVER_PORT = 5038
PS C:\Users\KIIT> $Env:ANDROID_ADB_SERVER_PORT = 5038
PS C:\Users\KIIT> & adb -a -P $Env:ANDROID_ADB_SERVER_PORT nodaemon server
01-22 02:34:00.030 24300 24304 I adb.exe : auth.cpp:416 adb_auth_init...
01-22 02:34:00.041 24300 24304 I adb.exe : auth.cpp:152 loaded new key from 'C:\Users\KIIT\.android\adbkey' with fingerprint 1B1D13E5ADDD590F23B6214E38EF45F18490DDA
01-22 02:34:00.042 24300 2916 I adb.exe : transport.cpp:335 emulator-5554: read thread spawning
01-22 02:34:00.042 24300 6008 I adb.exe : transport.cpp:307 emulator-5554: write thread spawning
01-22 02:34:00.046 24300 24304 I adb.exe : transport.cpp:1734 fetching keys for transport emulator-5554
01-22 02:34:00.046 24300 24304 I adb.exe : auth.cpp:468 Calling send_auth_response
01-22 02:34:00.059 24300 24304 I adb.exe : adb.cpp:180 emulator-5554: offline
```

The screenshot shows a Windows PowerShell terminal window and a code editor. The terminal window, titled 'ashu@BT1000122019: ~/samp', displays the following commands and output:

```
ashu@BT1000122019:~$ adb kill-server
cannot connect to daemon at tcp:5037: Connection refused
ashu@BT1000122019:~$ export ADB_SERVER_SOCKET=tcp:192.168.129.54:5038
ashu@BT1000122019:~$ adb devices
List of devices attached
emulator-5554    device

ashu@BT1000122019:~$ maestro --host 192.168.129.54 test android-flow.yaml

Flow path does not exist: /home/ashu/android-flow.yaml
ashu@BT1000122019:~$ cd samples/
ashu@BT1000122019:~/samples$ maestro --host 192.168.129.54 test android-flow.yaml
^C
ashu@BT1000122019:~/samples$ maestro --host 192.168.129.54 test android-flow.yaml
^C
ashu@BT1000122019:~/samples$ maestro --host 192.168.129.54 test MyApp.yaml
^C
ashu@BT1000122019:~/samples$ maestro --host 192.168.129.54 test android-flow.yaml
^C
ashu@BT1000122019:~/samples$ cd~
Command 'cd~' not found, did you mean:
command 'cdo' from deb cdo (2.0.4-1)
command 'cdi' from deb cdo (2.0.4-1)
command 'cdp' from deb irpas (0.10-9)
command 'cde' from deb cde (0.1+git9-g551e54d-1.2)
command 'cdb' from deb tinycdb (0.78build3)
command 'cdw' from deb cdw (0.8.1-2)
command 'cd5' from deb cd5 (0.1-4)
Try: sudo apt install <deb name>
```

The code editor, titled 'Login.yaml', shows the following YAML content:

```
maestro > ! Login.yaml
1 #flow: Login
2 #intent:
3 # Open up our app and use the default credentials to login
4 # and navigate to the demo screen
5
6 appId: com.aashu.MyApp # the app id of the app we want to test
7 # You can find the appId of an Ignite app in the 'app.json' file
8 # as the "package" under the "android" section and "bundleIdentifier" under the "ios" section
9 ---
10 - clearState # clears the state of our app (navigation and authentication)
11 - launchApp # launches the app
12 - assertVisible: "Sign In"
13 - tapOn:
14   text: "Tap to sign in!"
15 - assertVisible: "Your app, almost ready for launch!"
16 - tapOn:
17   text: "Let's go!"
18 - assertVisible: "Components to jump start your project!"
19
20
```

### Step 3: Write Maestro Tests

The screenshot shows a code editor with a file named 'Login.yaml'. The file content is as follows:

```
maestro > ! Login.yaml
1 #flow: Login
2 #intent:
3 # Open up our app and use the default credentials to login
4 # and navigate to the demo screen
5
6 appId: com.aashu.MyApp # the app id of the app we want to test
7 # You can find the appId of an Ignite app in the 'app.json' file
8 # as the "package" under the "android" section and "bundleIdentifier" under the "ios" section
9 ---
10 - clearState # clears the state of our app (navigation and authentication)
11 - launchApp # launches the app
12 - assertVisible: "Sign In"
13 - tapOn:
14   text: "Tap to sign in!"
15 - assertVisible: "Your app, almost ready for launch!"
16 - tapOn:
17   text: "Let's go!"
18 - assertVisible: "Components to jump start your project!"
19
20
```