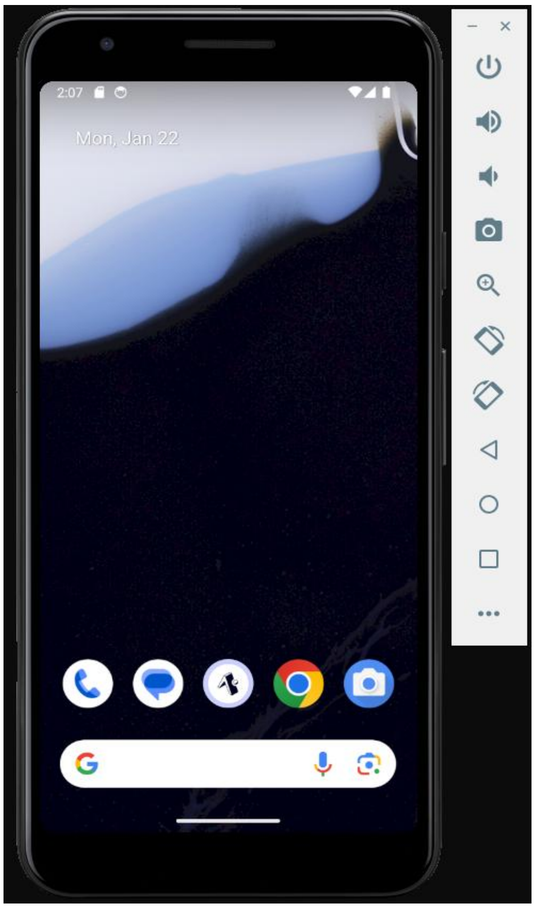
Maestro Test Automation Documentation

Step 1: Environment Setup

1.1 Install Android Studio

-> Download Android Studio App from browser
-> After installation, open Android Studio
-> Click on ‘More actions’
-> Launch ‘Pixel_3a’
-> Android mobile will pop up
-> Add ‘Android/Sdk’ and ‘Android/Sdk/platform-tools’ path to your environment variable



1.2 Connect ADB devices

‘adb’ devices is a command used in Android development and debugging to check the list of Android devices connected to your computer via the Android Debug Bridge (ADB). ADB is a versatile command-line tool that allows developers to interact with Android devices for various purposes, such as installing and debugging applications, transferring files, and accessing the device shell.

Commands:

Open Windows Powershell

- Check adb version

```
  adb --version
```

- Install WSL

```
 wsl --install
```

- Install Ubuntu

```
 wsl --install -d Ubuntu
 wsl --update
```

Open Ubuntu

- Sudo Command

```
  sudo apt-get update
  sudo apt update
```

-Download latest command line tools for linux from browser

- Install Java

    sudo apt install openjdk-11-jdk

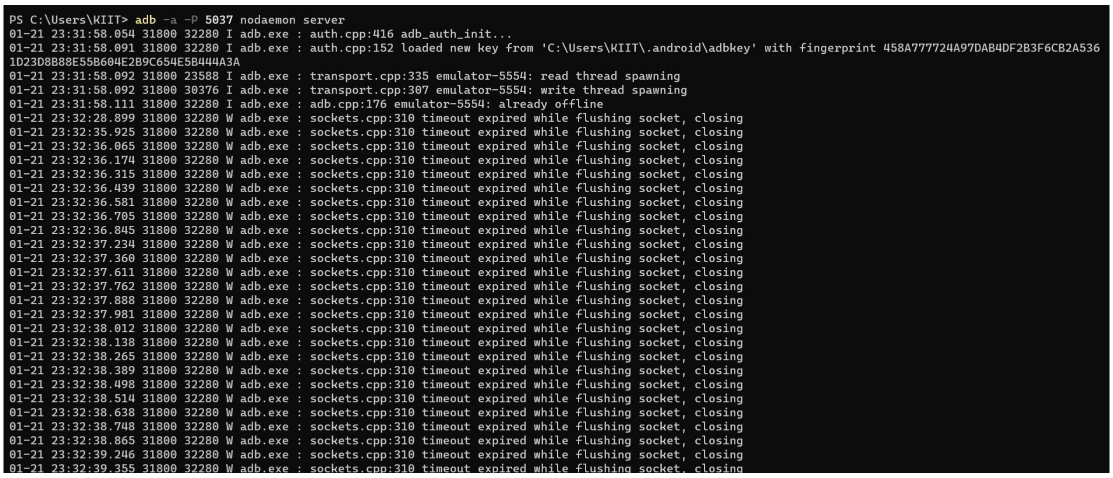
- Follow the process given in the doc for installing and upgrading Maestro CLI
  https://maestro.mobile.dev/getting-started/installing-maestro/windows

- Ensure Maestro CLI is installed successfully by running:

  maestro --version

-Start the adb server in window host in Windows Powershell

    adb kill server
    adb -a -P 5037 nodaemon server



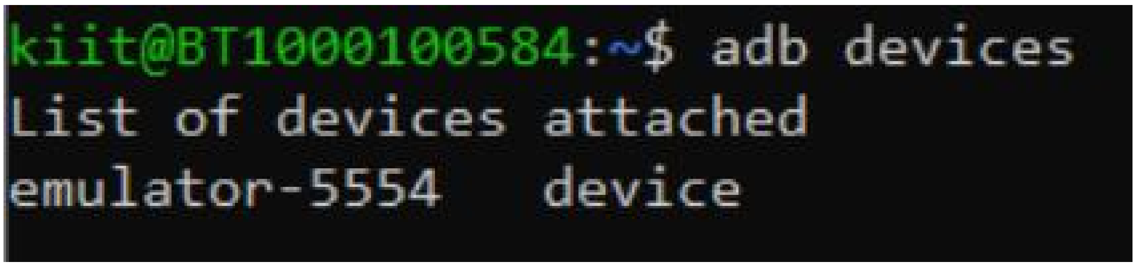
```
PS C:\Users\KIIT> adb -a -P 5037 nodaemon server
01-21 23:31:58.054 31800 32280 I adb.exe : auth.cpp:416 adb_auth_init...
01-21 23:31:58.091 31800 32280 I adb.exe : auth.cpp:152 loaded new key from 'C:\Users\KIIT\.android\adbkey' with fingerprint 458A777724A97DAB4DF2B3F6CB2A536
1D23D8B88E55B604E2B9C654E5B444A3A
01-21 23:31:58.092 31800 23588 I adb.exe : transport.cpp:335 emulator-5554: read thread spawning
01-21 23:31:58.092 31800 30376 I adb.exe : transport.cpp:307 emulator-5554: write thread spawning
01-21 23:31:58.111 31800 32280 I adb.exe : adb.cpp:176 emulator-5554: already offline
01-21 23:32:28.899 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:35.925 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.065 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.174 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.315 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.439 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.581 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.705 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:36.845 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.234 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.360 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.611 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.762 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.888 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:37.981 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.012 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.138 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.265 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.389 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.498 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.514 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.638 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.748 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:38.865 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:39.246 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
01-21 23:32:39.355 31800 32280 W adb.exe : sockets.cpp:310 timeout expired while flushing socket, closing
```

  -adb connection

  export ADB_SERVER_SOCKET=tcp:<WINDOWS_IPv4_ADDR>:5037

-Check connected adb devices

  adb devices



```
kiit@BT1000100584:~$ adb devices
List of devices attached
emulator-5554    device
```

1.3 Setup of React Native development environment

-> Select any preferable IDE
-> Download the required dependencies
-> Check app.json and package.json file

app.json file

```
{
  "name": "PizzaApp",
  "displayName": "PizzaApp",
  "expo": {
    "name": "PizzaApp",
    "slug": "PizzaApp",
    "scheme": "pizzaapp",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/images/app-icon-all.png",
    "splash": {
      "image": "./assets/images/splash-logo-all.png",
      "resizeMode": "contain",
      "backgroundColor": "#191015"
    },
    "updates": {
      "fallbackToCacheTimeout": 0
    },
    "jsEngine": "hermes",
    "assetBundlePatterns": [
      "**/*"
    ],
    "android": {
      "icon": "./assets/images/app-icon-android-legacy.png",
      "package": "com.pizzaapp",
      "adaptiveIcon": {
        "foregroundImage": "./assets/images/app-icon-android-adaptive-foreground.png",
        "backgroundImage": "./assets/images/app-icon-android-adaptive-background.png"
      },
      "splash": {
        "image": "./assets/images/splash-logo-android-universal.png",
        "resizeMode": "contain",
        "backgroundColor": "#191015"
      }
    },
    "ios": {
      "icon": "./assets/images/app-icon-ios.png",
      "supportsTablet": true,
      "bundleIdentifier": "com.pizzaapp",
      "splash": {
        "image": "./assets/images/splash-logo-ios-mobile.png",
        "tabletImage": "./assets/images/splash-logo-ios-tablet.png",
        "resizeMode": "contain",
        "backgroundColor": "#191015"
      }
    },
    "web": {
      "favicon": "./assets/images/app-icon-web-favicon.png",
      "splash": {
        "image": "./assets/images/splash-logo-web.png",
        "resizeMode": "contain",
        "backgroundColor": "#191015"
      },
      "bundler": "metro"
    },
    "plugins": [
      "expo-localization",
      [
        "expo-build-properties",
        {
          "ios": {
            "newArchEnabled": false
          },
          "android": {
            "newArchEnabled": false
          }
        }
      ]
    ],
    "experiments": {
      "tsconfigPaths": true
    }
  },
  "ignite": {
    "version": "9.4.0"
  }
}
```

Step 2: Maestro Test cases

1.1 Scenarios

-> Login into the App

```
#flow: Login
#intent:
# Open up our app and use the default credentials to login
# and navigate to the demo screen

appId: com.pizzaapp # the app id of the app we want to test
# You can find the appId of an Ignite app in the `app.json` file
# as the "package" under the "android" section and "bundleIdentifier" under the "ios" section
---
- clearState # clears the state of our app (navigation and authentication)
- launchApp # launches the app
- assertVisible: "PizzaApp"
- tapOn:
    text: "PizzaApp"
- assertVisible: "Sign In"
- tapOn:
    text: "Tap to sign in!"
- tapOn:
    text: "Continue"
- assertVisible: "Your app, almost ready for launch!"
- tapOn:
    text: "Let's go!"
- assertVisible: "Components to jump start your project!"
```

-> Select Favourite Podcast

```
# flow: run the login flow and then navigate to the demo podcast list screen, favorite a
podcast, and then switch the list to only be favorites.

appId: com.pizzaapp
env:
  TITLE: "RNR 277 - Expo Launch Party"
  FAVORITES_TEXT: "Only Show Favorites"
---
- tapOn: "Podcast"
- scrollUntilVisible:
    element:
      text: ${FAVORITES_TEXT}
    direction: UP
    timeout: 50000
    speed: 80
    visibilityPercentage: 0
- assertVisible: "React Native Radio episodes"
- tapOn:
    text: ${FAVORITES_TEXT}
- assertVisible: "This looks a bit empty"
- tapOn:
    text: ${FAVORITES_TEXT}
    # https://maestro.mobile.dev/troubleshooting/known-issues#android-accidental-double-tap
    retryTapIfNoChange: false
- scrollUntilVisible:
    element:
      text: ${TITLE}
    direction: DOWN
    timeout: 50000
    speed: 50
    visibilityPercentage: 100
- longPressOn: ${TITLE}
- scrollUntilVisible:
    element:
      text: ${FAVORITES_TEXT}
    direction: UP
    timeout: 50000
    speed: 40
    visibilityPercentage: 100
- tapOn:
    text: ${FAVORITES_TEXT}
- assertVisible: ${TITLE}
```

-> Go ToButton

```
# flow: run the login flow and then go to menu bar click on button

appId: com.pizzaapp
env:
  TITLE: "Button"
---
- runFlow: Login.yaml
- tapOn:
    point: "7%,6%"
- assertVisible: "Button"
- tapOn: ${TITLE}
- assertVisible: "Presets"
```

-> Logout From the Application

```
# flow: run the login flow and then navigate to the Debug screen and click on the Logout
Button.

appId: com.pizzaapp
env:
  TITLE: "Log Out"
---
- runFlow: Login.yaml
- tapOn: "Debug"
- scrollUntilVisible:
    element:
      text: ${TITLE}
    direction: DOWN
    timeout: 50000
    speed: 80
    visibilityPercentage: 0
- assertVisible: "Log Out"
- tapOn:
    text: ${TITLE}
- assertVisible: "Sign In"
```

-> Send Message

```
# flow: run the login flow and then navigate to the demo podcast list screen, favorite a
podcast, and then switch the list to only be favorites.

appId: com.pizzaapp
env:
  TITLE: "Send us a message"
---
- runFlow: Login.yaml
- tapOn: "Community"
- scrollUntilVisible:
    element:
      text: ${TITLE}
    direction: DOWN
    timeout: 50000
    speed: 80
```

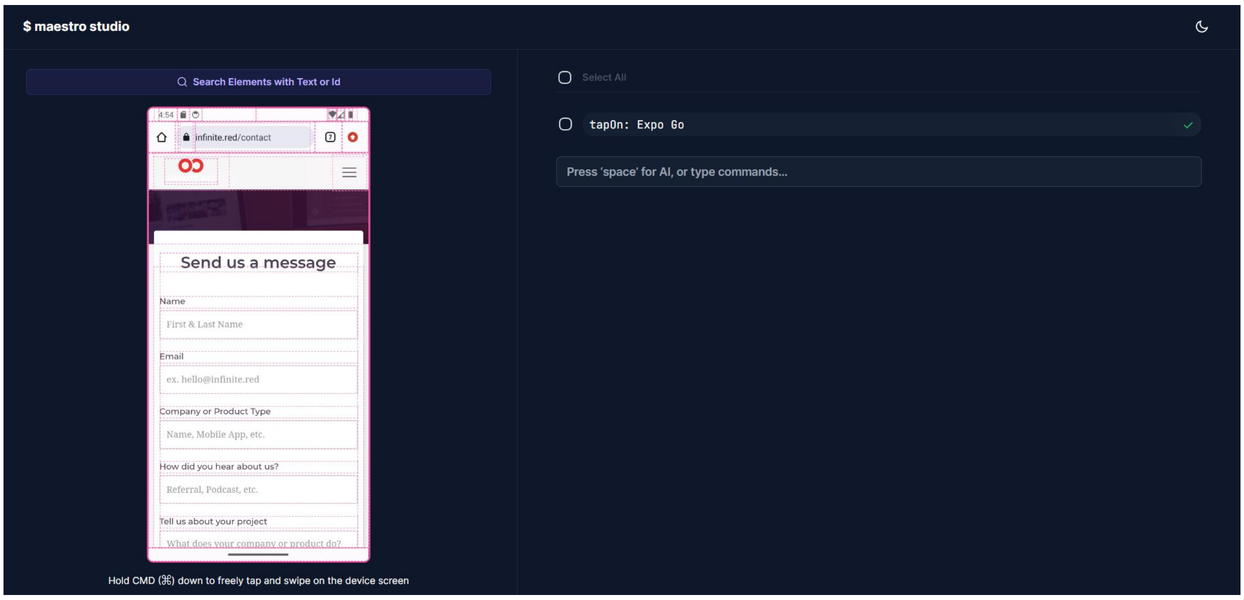
```yaml
    visibilityPercentage: 100
- assertVisible: ${TITLE}
- tapOn: ${TITLE}
- assertVisible: ${TITLE}
- scrollUntilVisible:
    element:
      id: "name"
    direction: DOWN
    speed: 20
- tapOn:
    id: "name"
- inputText: "Monsoon Maurya"
- scrollUntilVisible:
    element:
      id: "Email-5"
    direction: DOWN
    speed: 20
- tapOn:
    id: "Email-5"
- inputText: "abc@gmail.com"
- scrollUntilVisible:
    element:
      id: "Company"
    direction: DOWN
    speed: 20
- tapOn:
    id: "Company"
- inputText: "Expo Go"
- scrollUntilVisible:
    element:
      id: "Referral"
    direction: DOWN
    speed: 20
- tapOn:
    id: "Referral"
- inputText: "Podcast"
- scrollUntilVisible:
    element:
      id: "Message-2"
    direction: DOWN
    speed: 20
- tapOn:
    id: "Message-2"
- inputText: "This is an assesment for UI Automation Testing"
- tapOn: "Send Message"
- scrollUntilVisible:
    element:
      text: "Message sent!"
    direction: UP
    timeout: 50000
    speed: 40
    visibilityPercentage: 100
- back
```

1.2 Command To Run Test Cases

```
cp /mnt/path/to/file/<maestro_file> .
maestro --host <IP_Address> test <maestro_file_>
```

1.3 Command to Run Maestro Studio

```
maestro --host <WINDOWS_IPV4_ADDR> studio
```



$ maestro studio

Search Elements with Text or Id

Select All

```
tapOn: Expo Go
```

Press 'space' for AI, or type commands...

Send us a message

Name

First & Last Name

Email

ex. hello@infinite.red

Company or Product Type

Name, Mobile App, etc.

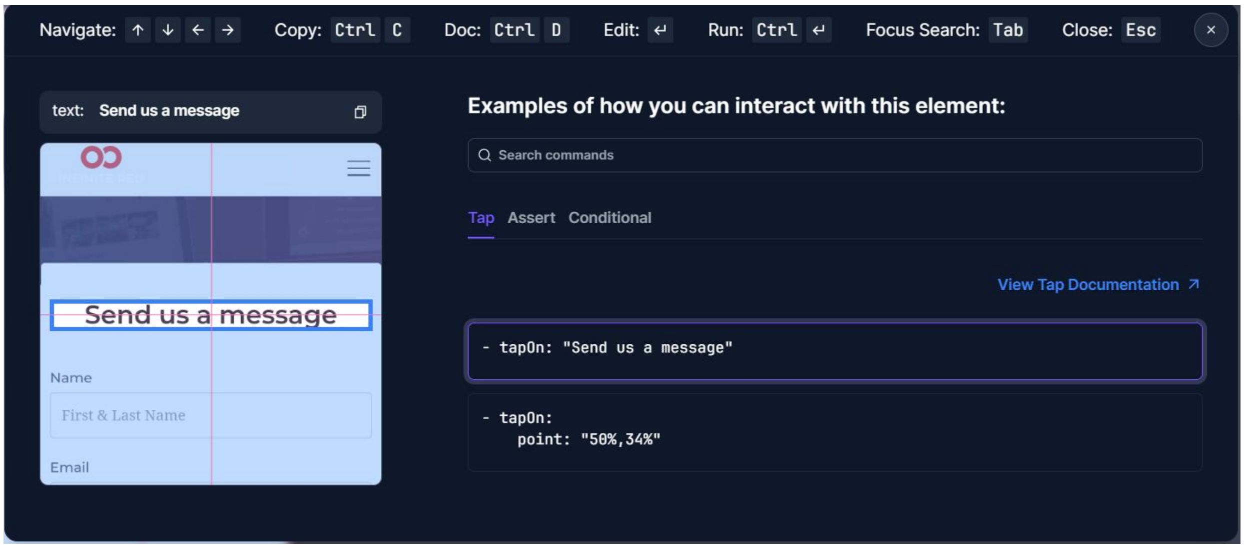
How did you hear about us?

Referral, Podcast, etc.

Tell us about your project

What does your company or product do?

Hold CMD (⌘) down to freely tap and swipe on the device screen



Navigate: ↑ ↓ ← → Copy: Ctrl C Doc: Ctrl D Edit: ↵ Run: Ctrl ↵ Focus Search: Tab Close: Esc

text: Send us a message

Send us a message

Name

First & Last Name

Email

Examples of how you can interact with this element:

Search commands

Tap Assert Conditional

View Tap Documentation ↗

```
- tapOn: "Send us a message"
```

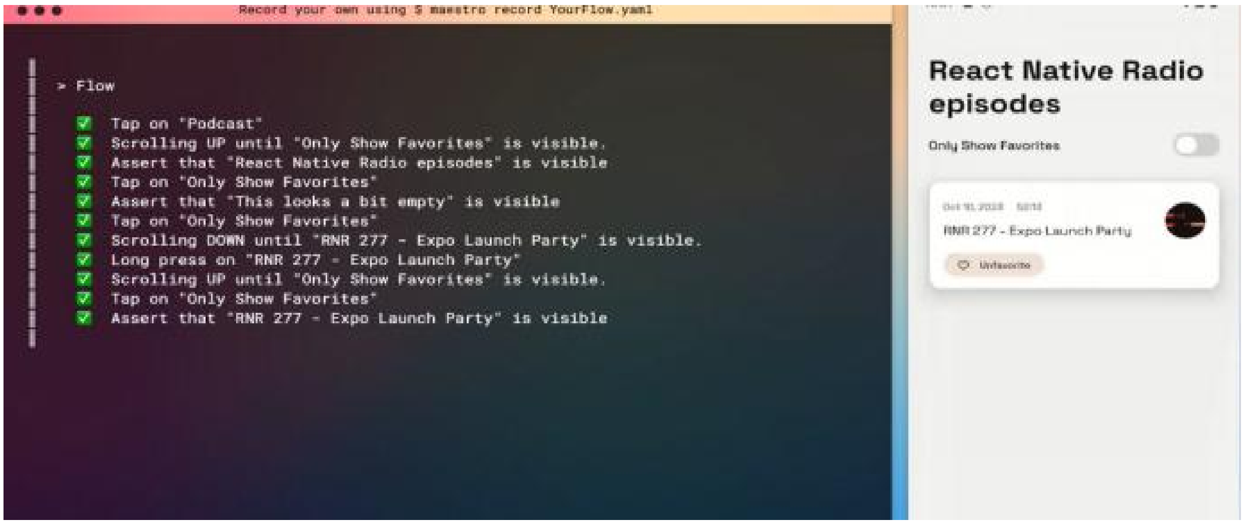
```
- tapOn:
    point: "50%,34%"
```

1.4 Command to Run Project on VsCode IDE

```
yarn android
```

Step 3: Maestro Test cases Report

-> Favourite Podcast Report

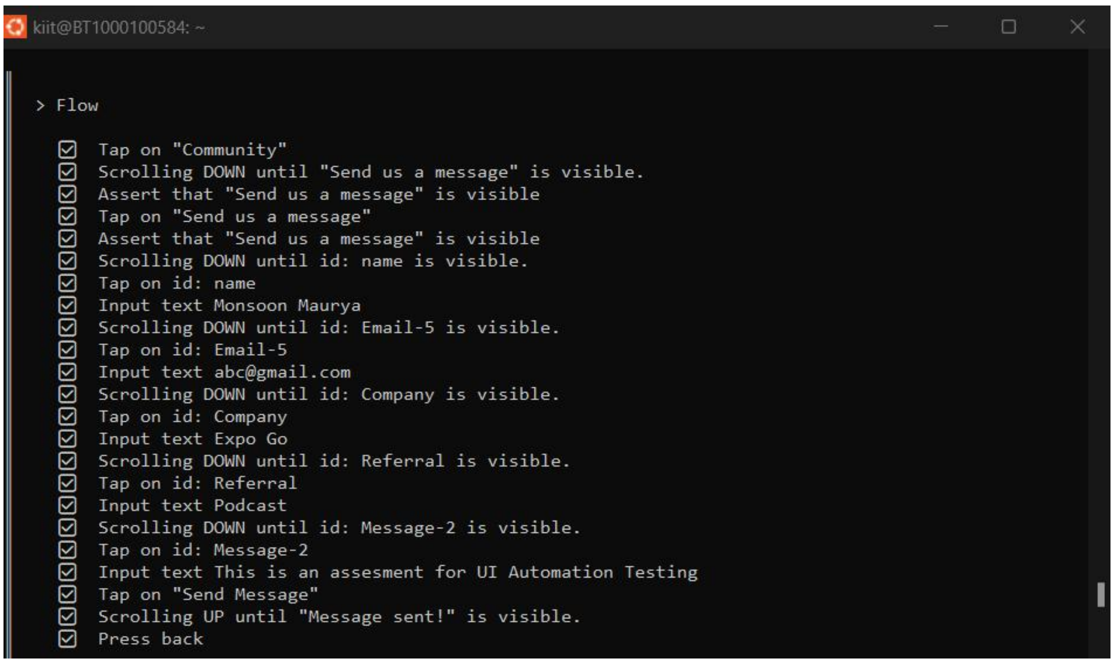


```
Record your own using $ maestro record YourFlow.yaml

 > Flow

   ✅ Tap on "Podcast"
   ✅ Scrolling UP until "Only Show Favorites" is visible.
   ✅ Assert that "React Native Radio episodes" is visible
   ✅ Tap on "Only Show Favorites"
   ✅ Assert that "This looks a bit empty" is visible
   ✅ Tap on "Only Show Favorites"
   ✅ Scrolling DOWN until "RNR 277 - Expo Launch Party" is visible.
   ✅ Long press on "RNR 277 - Expo Launch Party"
   ✅ Scrolling UP until "Only Show Favorites" is visible.
   ✅ Tap on "Only Show Favorites"
   ✅ Assert that "RNR 277 - Expo Launch Party" is visible
```

-> Send Message Report



```
kiit@BT1000100584: ~

 > Flow

   ☑ Tap on "Community"
   ☑ Scrolling DOWN until "Send us a message" is visible.
   ☑ Assert that "Send us a message" is visible
   ☑ Tap on "Send us a message"
   ☑ Assert that "Send us a message" is visible
   ☑ Scrolling DOWN until id: name is visible.
   ☑ Tap on id: name
   ☑ Input text Monsoon Maurya
   ☑ Scrolling DOWN until id: Email-5 is visible.
   ☑ Tap on id: Email-5
   ☑ Input text abc@gmail.com
   ☑ Scrolling DOWN until id: Company is visible.
   ☑ Tap on id: Company
   ☑ Input text Expo Go
   ☑ Scrolling DOWN until id: Referral is visible.
   ☑ Tap on id: Referral
   ☑ Input text Podcast
   ☑ Scrolling DOWN until id: Message-2 is visible.
   ☑ Tap on id: Message-2
   ☑ Input text This is an assesment for UI Automation Testing
   ☑ Tap on "Send Message"
   ☑ Scrolling UP until "Message sent!" is visible.
   ☑ Press back
```