Problem Statement 1

In this task, you will create end-to-end tests for a React Native application using Maestro.

Here's a structured approach to guide you through the process:

Submitted by:
Veer Kumar (2005629) CSE

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

We use React Native, Android studio and ignite red boiler plate and Maestro for testing automation.Also, we use Visual Studio as a code editor tool.

STEPS FROM SETTING UP ENVIRONMENT TO WRITING TESTING CODE

Step1:

You need to install and setup the environement Of React-Native
https://reactnative.dev/docs/0.72/environment-setup

Install Node via Chocolatey a popula Manager for Windows
https://chocolatey.org/

Open an Administrator Command Prompt (right click Command Prompt and select "Run as Administrator"), then run the following command:

Run this command

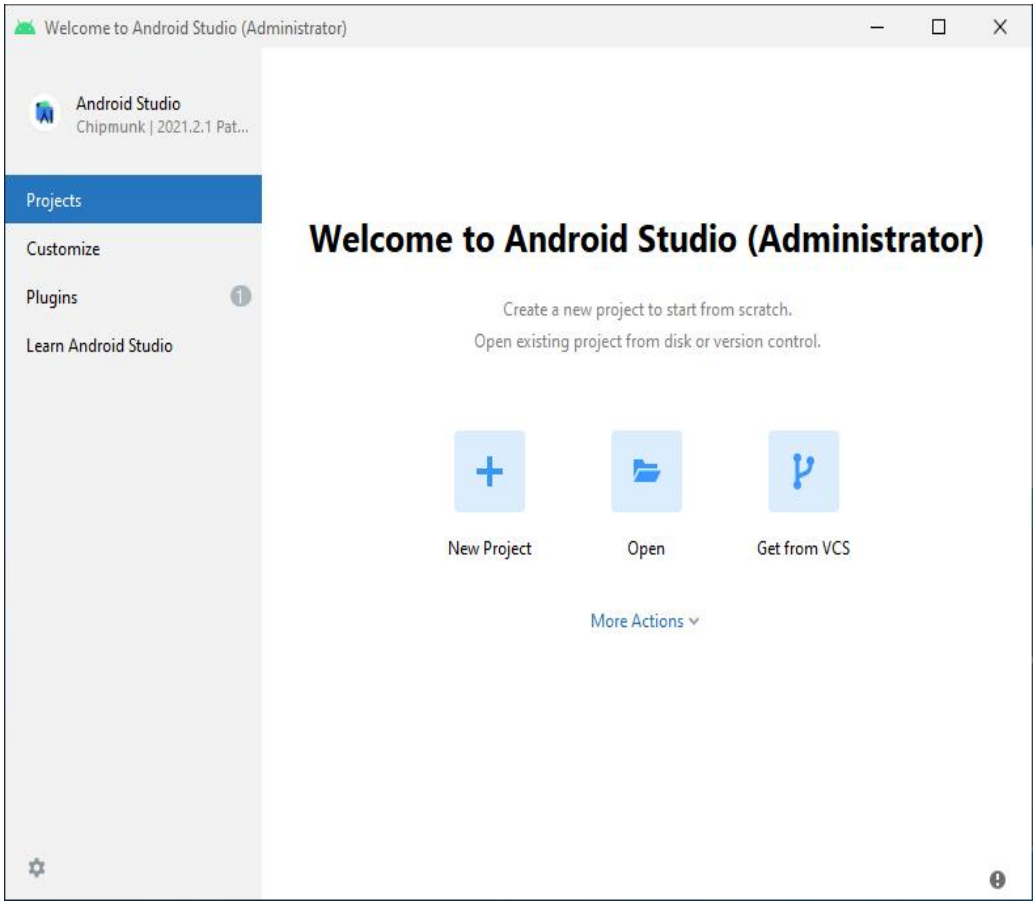
- choco install -y nodejs-lts microsoft-openjdk11

Step2:

Android development environment

Download and Install Android Studio and setup the SDK.
https://developer.android.com/studio

2. Install the Android SDK


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Pat...
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio (Administrator)
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions

Select the SDK manager and choose

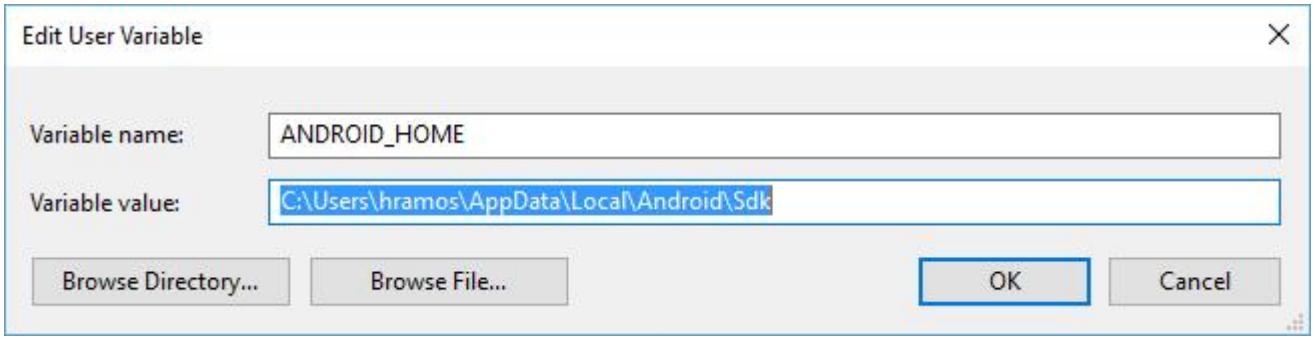
- Android SDK Platform 33
- Intel x86 Atom_64 System Image or Google APIs Intel x86 Atom System Image

Finally, click "Apply" to download and install the Android SDK and related build tools.

3. Configure the ANDROID_HOME environment variable

The React Native tools require some environment variables to be set up in order to build apps with native code.

1. Open the **Windows Control Panel.**
2. Click on **User Accounts**, then click **User Accounts** again
3. Click on **Change my environment variables**
4. Click on **New...** to create a new ANDROID_HOME user variable that points to the path to your Android SDK:



Location will be %LOCALAPPDATA%\Android\Sdk

Open a new Command Prompt window to ensure the new environment variable is loaded before proceeding to the next step.

1. Open powershell
2. Copy and paste **Get-ChildItem -Path Env:** into powershell
3. Verify ANDROID_HOME has been added

4. Add platform-tools to Path

1. Open the **Windows Control Panel.**
2. Click on **User Accounts**, then click **User Accounts** again
3. Click on **Change my environment variables**
4. Select the **Path** variable.
5. Click **Edit.**
6. Click **New** and add the path to platform-tools to the list.

STEP 3: Creating a new React Native project with Ignite on Terminal or Powershell

To start off, you can create a new Ignite project using the following command:

- **npx ignite-cli new YourProjectName**

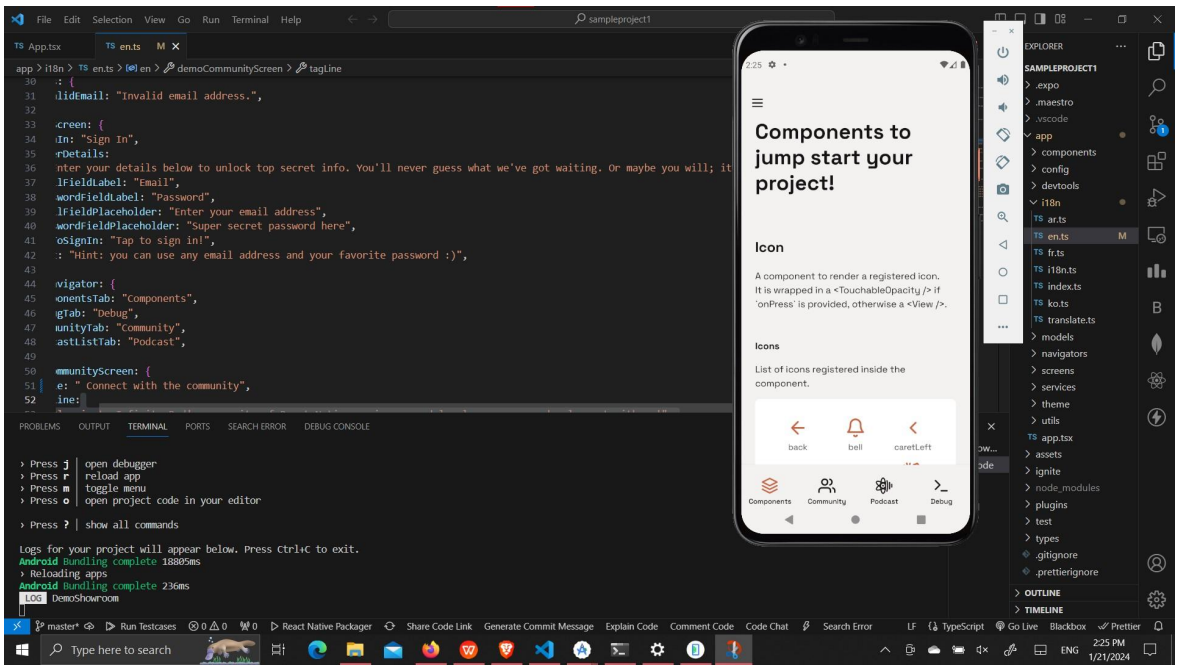
This will generate a new React Native project with all the functionality.

- **npx ignite-cli new HealthTracker --expo**

Once You Create the new project after that you Open Your Powershell on RUN on Admin

- Cd yourprojectname
- npm run android

Then Your android studio will start



Step 4:

Setting Up Maestro in Ignite

MAESTRO INSTALLATION

We're going to start by installing Maestro via the terminal. To do this, we'll need to run the following command:

Command :-

```
curl -Ls "https://get.maestro.mobile.dev" | Bash
```

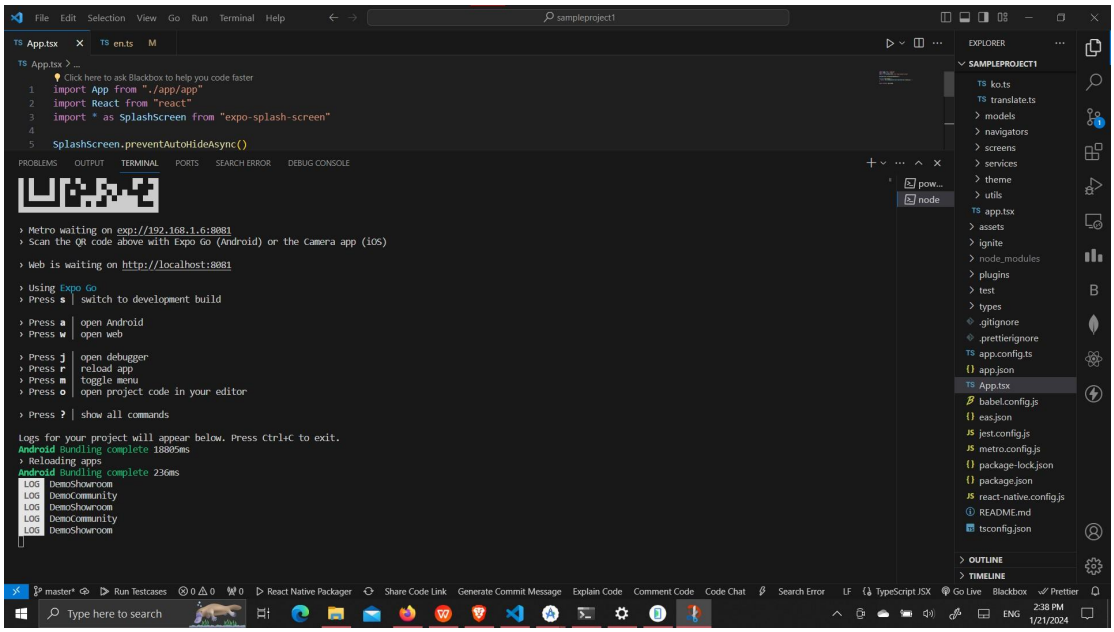
Creating our first Maestro Flow

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

appId: com.maestroapp # 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: "Sign In"
- tapOn:
text: "Tap to sign in!"
- assertVisible: "Your app, almost ready for launch!"
- tapOn:
```

```
text: "Let's go!"
- assertVisible: "Components to jump start your project!"
```

Here, is terminal snap where the android studio connects and reload the all stuffs.
It connects at the port of http://localhost:8081



http://localhost:8081

Let's set you up to use Android in your freshly installed WSL2

On UBUNTU for windows

- Download the Android command line tools zip file from Android official site.

Use the following instructions to set up Android command lines correctly in your WSL2.

Open WSL2 terminal.
Create a new directory in your home directory.

- **~ $ mkdir Android**
- **~ $ cd Android**

Now, Unzip the Android command line tools zip file in the android directory using this command: unzip ~<command_line_zip_filename>.zip

- $ mkdir latest
- $ mv cmdline-tools/* latest/
- $ mv latest/ cmdline-tools/

- Fire up your Android emulator on Windows.
- Once the Android emulator is up and running, open a PowerShell prompt.
- Run this command in PowerShell

adb -a -P 5037 nodaemon server

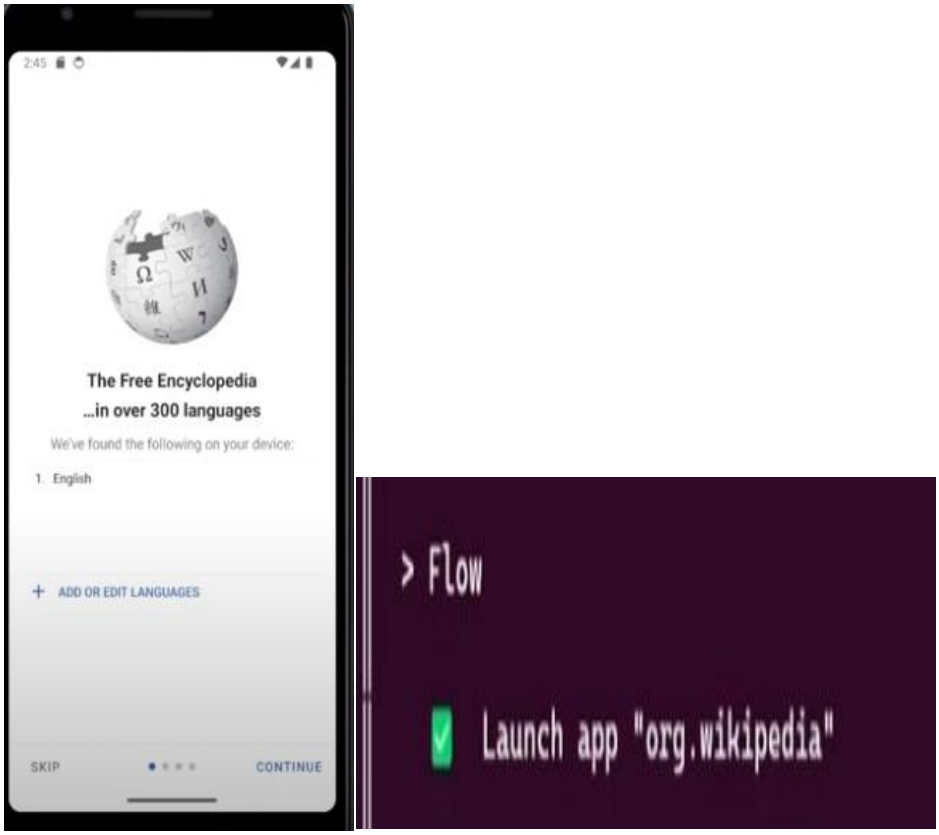
Now open your WSL2 terminal and run these commands.

- adb kill-server
- Export ADB_SERVER_SOCKET=tcp:<WINDOWS_IPV4_ADDR>:5037
- adb devices
- You should see your connected emulator successfully now.

Step 5: Ready to start Using Maestro?

You can run Maestro commands in WSL2 terminal with --host flag. eg.

- **maestro --host <WINDOWS_IPV4_ADDR> test flow.yaml**
- **maestro --host <WINDOWS_IPV4_ADDR> studio**



Successfully run Maestro Testing