Configuration Manual

Research Project

MSc. Cybersecurity

Christopher Tochukwu

Student ID: x20257457

School of Computing

National College of Ireland

Configuration Manual

Introduction

- **The hardware requirements and software setups, procedures** our record-keeping app using Flutter, Solidity, and Truffle**, as well as the whole project implementation, are all covered in detail in this configuration. The project's goal was to use blockchain technologies in medical health**
- **The technical specifications and procedures listed below lead to the project's outcomes.**
- System Configuration

Figure 1: System Configuration

This project was carried out using a MacBook Pro with a 1.4 GHz Quad-Core Intel Core i5 processor, 8GB of RAM, and a 1TB SSD running macOS Ventura.

Environment Setup

The software setup needed to run the project includes:

- Flutter
- Android Studio
- Xcode
- JDK
- Ganache
- Nodejs

For the implementation of this project, Flutter was chosen as the Cross Platform Mobile FrameWork. And Android Studio was used as our development Ide .Xcode was used to enable flutter support for iOS and to also provide us with IOS Simulators to test our apps on the laptop. JDK had to be installed as it is a requirement for flutter to work on Android studio

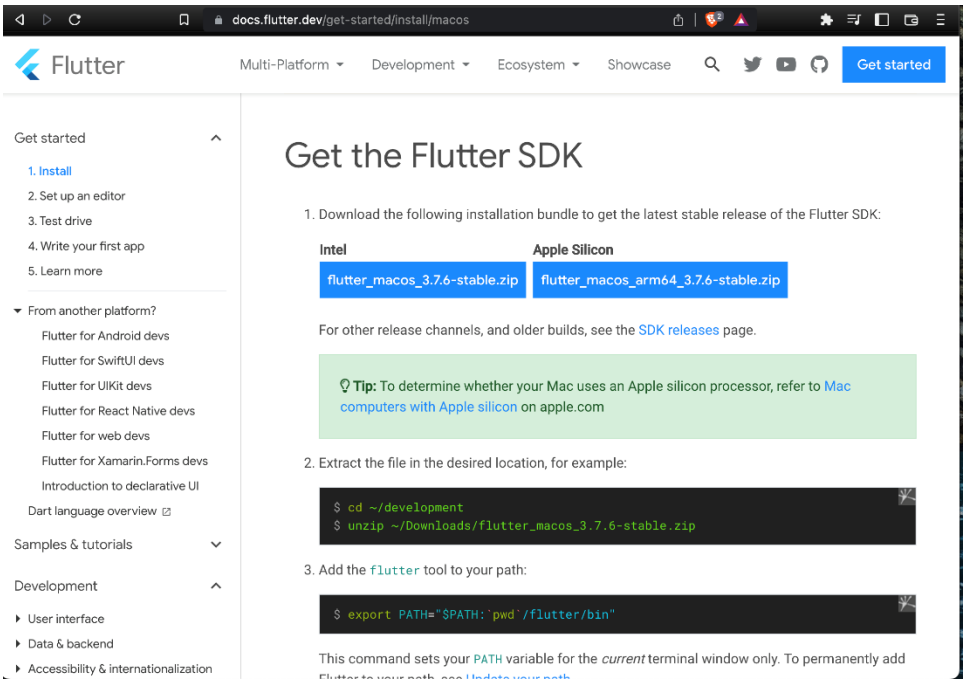
Ganache is a personal blockchain for ethereum development, which allows developers to test and deploy our smart contracts locally. It provides a local Ethereal network that can be used for testing and development purposes, without the need to deploy contracts to a a live network. Ganache provides a simple user interface for creating and managing blockchain accounts.

Nodejs has to be installed on our laptop to manage installation of truffle which compiles and migrates our solidity contracts

The latest version of Flutter that I downloaded and installed from the official website flutter.dev

Figure 2: Flutter download

Visit flutter.dev to download the flutter sdk as shown in the figure below



Download the sdk and follow instructions on that same page on how to configure the sdk path

1. Extract the file in the desired location, for example:

   cd ~/development
   unzip ~/Downloads/flutter_macos_3.7.6-stable.zip

2. Add the flutter tool to your path:
3. Content copy

   $ export PATH="$PATH:`pwd`/flutter/bin"

4. This command sets your PATH variable for the *current* terminal window only. To permanently add Flutter to your path, see Update your path

Run flutter doctor
Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the -v flag):

$ flutter doctor

This command checks your environment and displays a report to the terminal window. The Dart SDK is bundled with Flutter; it is not necessary to install Dart separately. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

[-] Android toolchain - develop for Android devices
  • Android SDK at /Users/obiwan/Library/Android/sdk
  ✗ **Android SDK is missing command line tools; download from https://goo.gl/XxQghQ**
  • Try re-installing or updating your Android SDK,
    visit https://docs.flutter.dev/setup/#android-setup for detailed instructions.

The following sections describe how to perform these tasks and finish the setup process.

Once you have installed any missing dependencies, run the flutter doctor command again to verify that you've set everything up correctly.

Update your path
You can update your PATH variable for the current session at the command line, as shown in Get the Flutter SDK. You'll probably want to update this variable permanently, so you can run flutter commands in any terminal session.

The steps for modifying this variable permanently for all terminal sessions are machine-specific. Typically you add a line to a file that is executed whenever you open a new window. For example:

5. Determine the path of your clone of the Flutter SDK. You need this in Step 3.
6. Open (or create) the rc file for your shell. Typing echo $SHELL in your Terminal tells you which shell you're using. If you're using Bash, edit $HOME/.bash_profile or $HOME/.bashrc. If you're using Z shell, edit $HOME/.zshrc. If you're using a different shell, the file path and filename will be different on your machine.
7. Add the following line and change [PATH_OF_FLUTTER_GIT_DIRECTORY] to be the path of your clone of the Flutter git repo:

   $ export PATH="$PATH:[PATH_OF_FLUTTER_GIT_DIRECTORY]/bin"

8. Run source $HOME/.<rc file> to refresh the current window, or open a new terminal window to automatically source the file.
9. Verify that the flutter/bin directory is now in your PATH by running:

   $ echo $PATH

10. Verify that the flutter command is available by running:
11. $ which flutter

Figure 3: IOS Setup

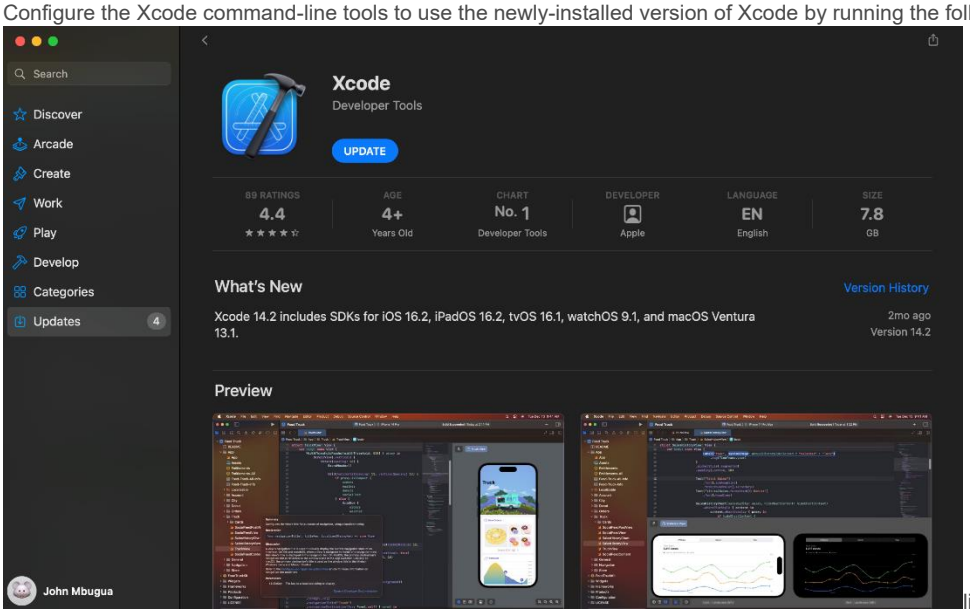
Upon the completion of installation, Flutter is ready but we cant still run any code. We need to make new changes

Install Xcode

To develop Flutter apps for iOS, you need a Mac with Xcode installed.

Install the latest stable version of Xcode (using web download or the Mac App Store).

Configure the Xcode command-line tools to use the newly-installed version of Xcode by running the following from the command line:




```
$ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
$ sudo xcodebuild -runFirstLaunch
```

This is the correct path for most cases, when you want to use the latest version of Xcode. If you need to use a different version, specify that path instead.

Make sure the Xcode license agreement is signed by either opening Xcode once and confirming or running sudo xcodebuild -license from the command line.

Versions older than the latest stable version may still work, but are not recommended for Flutter development.

With Xcode, you'll be able to run Flutter apps on an iOS device or on the simulator.

Set up the iOS simulator

To prepare to run and test your Flutter app on the iOS simulator, follow these steps:

On your Mac, find the Simulator via Spotlight or by using the following command:

```
$ open -a Simulator
```

Make sure your simulator is using a 64-bit device (iPhone 5s or later). You can check the device by viewing the settings in the simulator's **Hardware > Device** or **File > Open Simulator** menus.

Depending on your development machine's screen size, simulated high-screen-density iOS devices might overflow your screen. Grab the corner of the simulator and drag it to change the scale. You can also use the **Window > Physical Size** or **Window > Pixel Accurate** options if your computer's resolution is high enough.

Figure 4: Android Studio Setup

Flutter relies on a full installation of Android Studio to supply its Android platform dependencies. However, you can write your Flutter apps in a number of editors.

Install Android Studio

Download and install Android Studio.

Start Android Studio, and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.

Run flutter doctor to confirm that Flutter has located your installation of Android Studio. If Flutter cannot locate it, run flutter config --android-studio-dir <directory> to set the directory that Android Studio is installed to.

Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 4.1 (API level 16) or higher.

Enable Developer options and USB debugging on your device. Detailed instructions are available in the Android documentation.

Windows-only: Install the Google USB Driver.

Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.

In the terminal, run the flutter devices command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your adb tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the ANDROID_SDK_ROOT environment variable to that installation directory.

Set up the Android emulator

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

Enable VM acceleration on your machine.

Launch Android Studio, click the AVD Manager icon, and select Create Virtual Device…

In older versions of Android Studio, you should instead launch Android Studio > Tools > Android > AVD Manager and select Create Virtual Device…. (The Android submenu is only present when inside an Android project.)

If you do not have a project open, you can choose Configure > AVD Manager and select Create Virtual Device…

Choose a device definition and select Next.

Select one or more system images for the Android versions you want to emulate, and select Next. An x86 or x86_64 image is recommended.

Under Emulated Performance, select Hardware - GLES 2.0 to enable hardware acceleration.

Verify the AVD configuration is correct, and select Finish.

For details on the above steps, see Managing AVDs.

In Android Virtual Device Manager, click Run in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

Agree to Android Licenses

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

Make sure that you have a version of Java 11 installed and that your JAVA_HOME environment variable is set to the JDK's folder.

Android Studio versions 2.2 and higher come with a JDK, so this should already be done.

Open an elevated console window and run the following command to begin signing licenses.

$ flutter doctor --android-licenses

Review the terms of each license carefully before agreeing to them.

Once you are done agreeing with licenses, run flutter doctor again to confirm that you are ready to use Flutter.

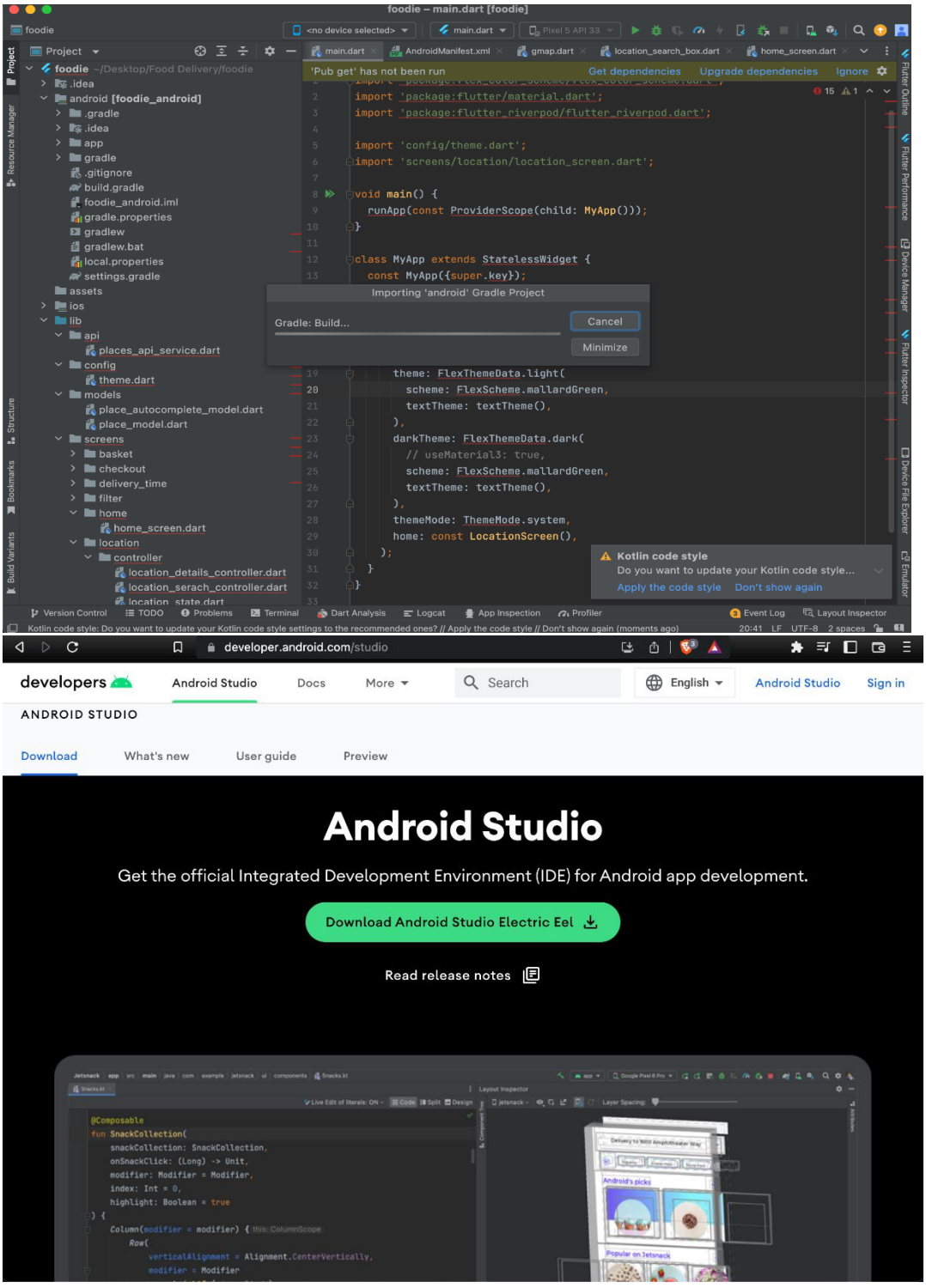
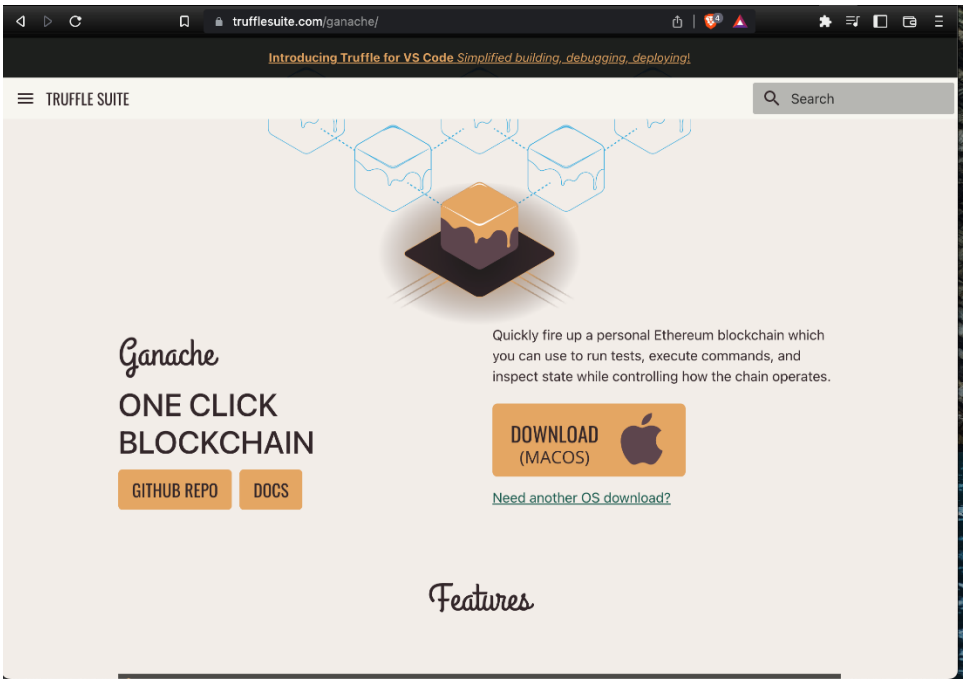


Figure 5: Install Ganache



Visit https://trufflesuite.com/ganache/ and install ganache

Figure 6: Install Nodejs

Visit https://nodejs.org/en/ to download nodejs which will be necessary for installing truffle

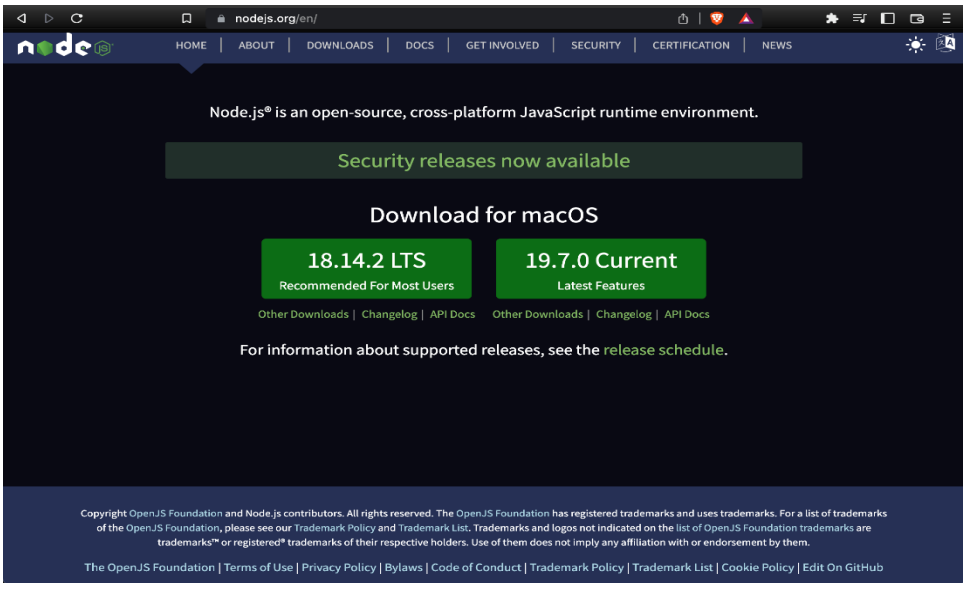
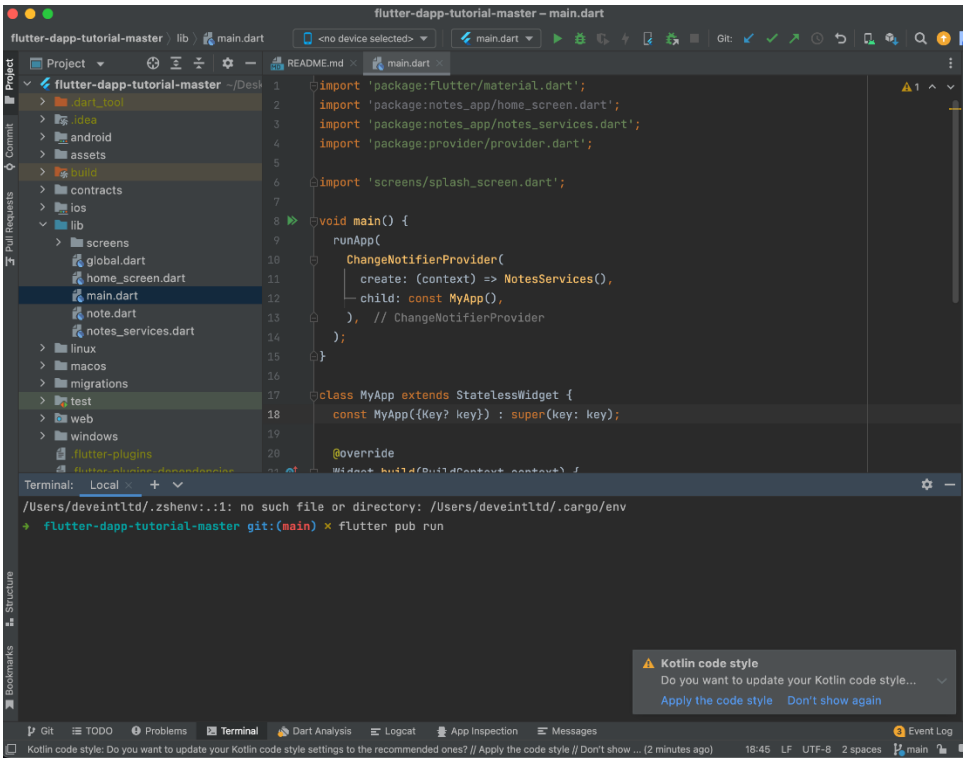


Figure 7: Open the attached extracted zipped file

Open the extracted folder in android studio and run the following command



flutter pub run

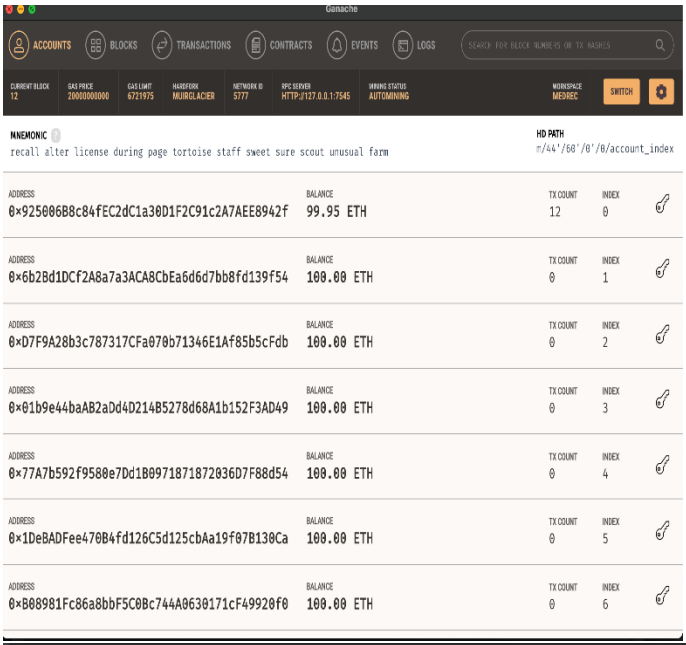
Run open -a Simulator

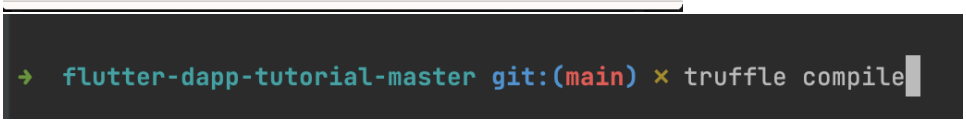
To open a simulator

Figure 8: Install Truffle

In your android studio terminal type

Sudo npm install truffle





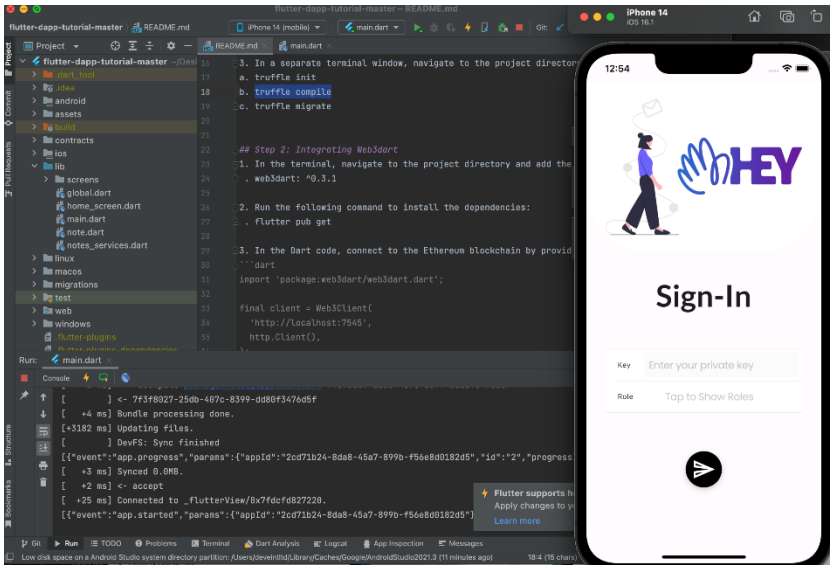
Start Ganache to create a local blockchain network by running the following command in the terminal:

a. truffle init
b. truffle compile
c. truffle migrate

Figure 9: Run your flutter code

On your terminal type

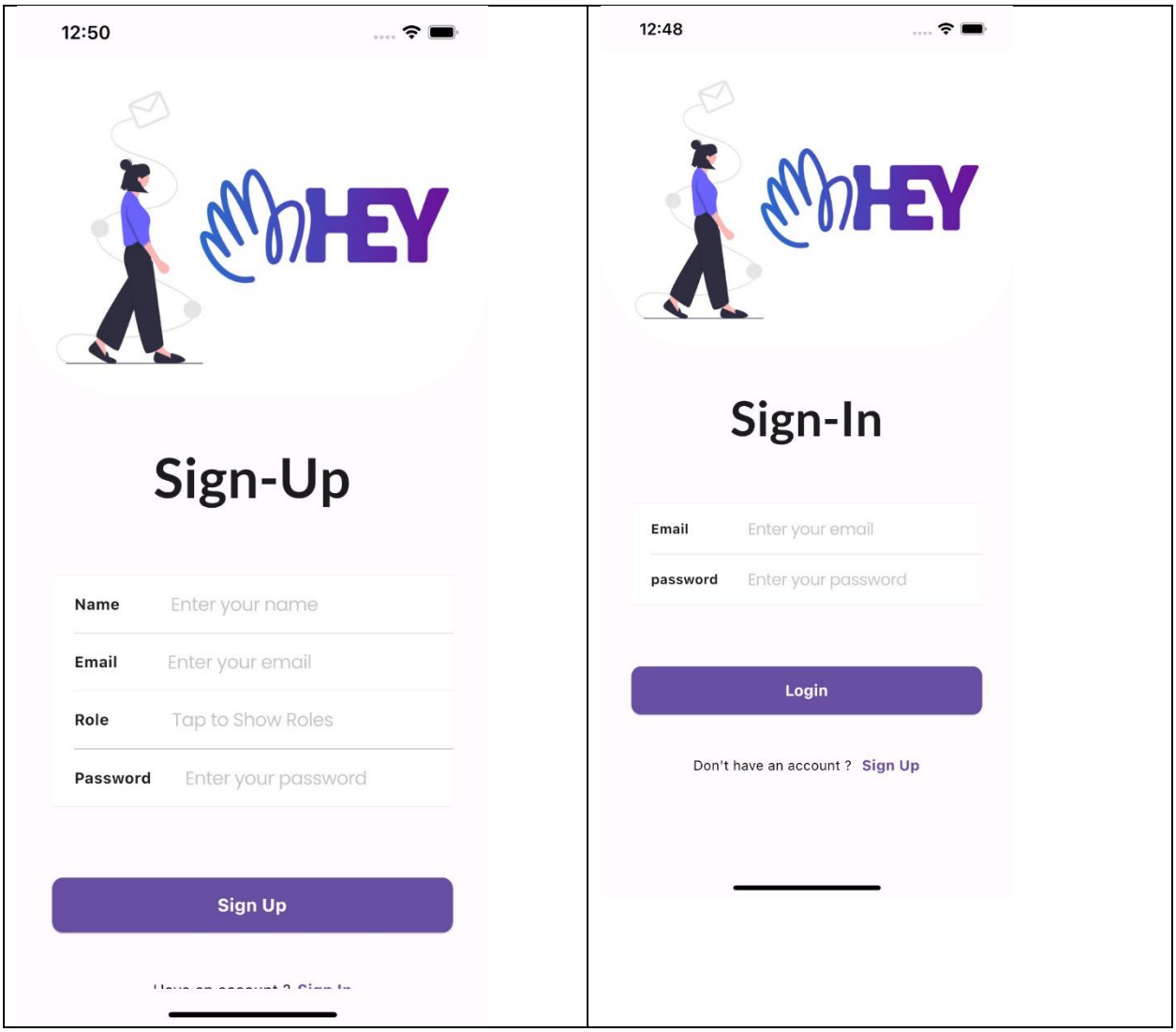
flutter run

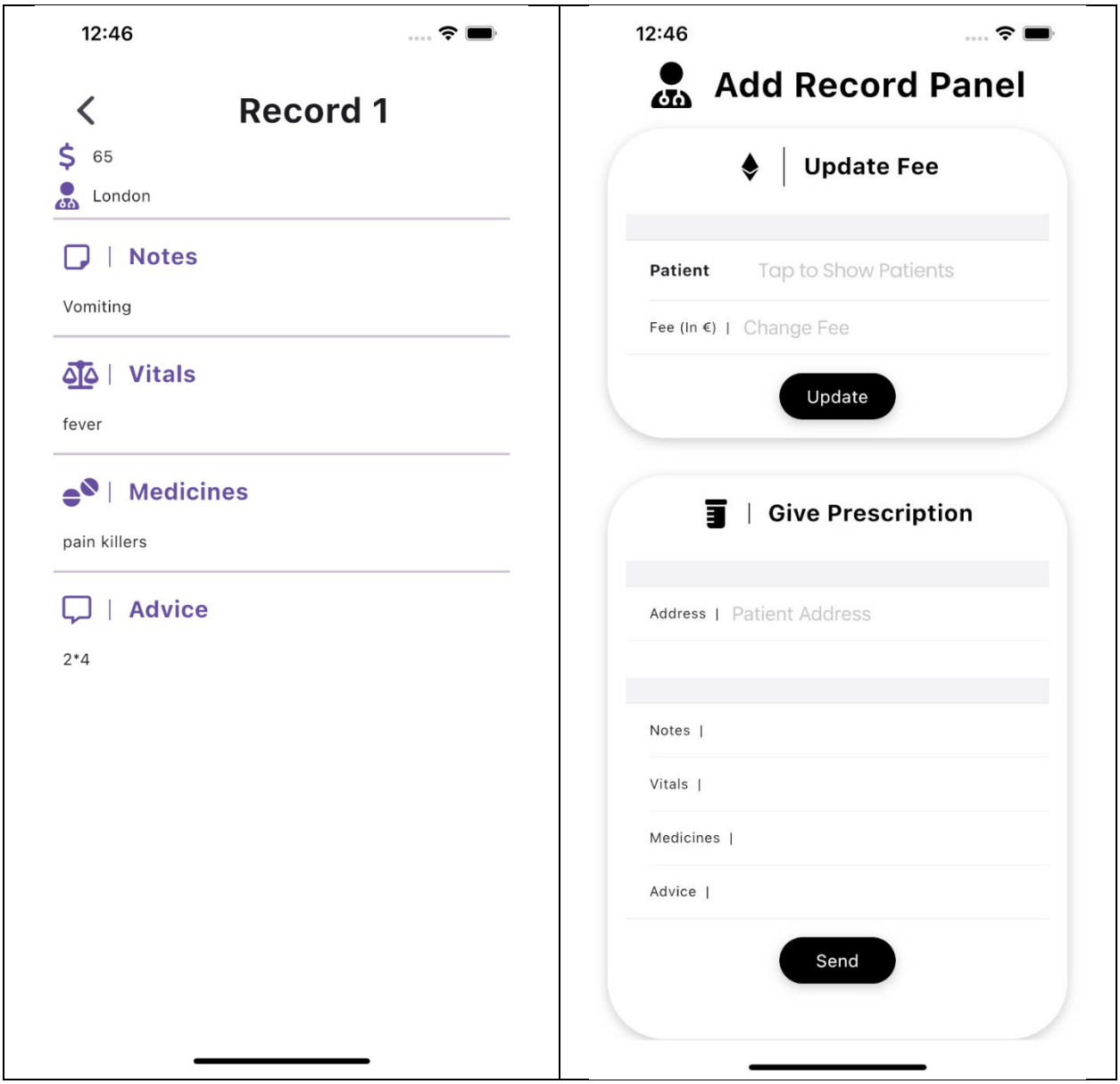


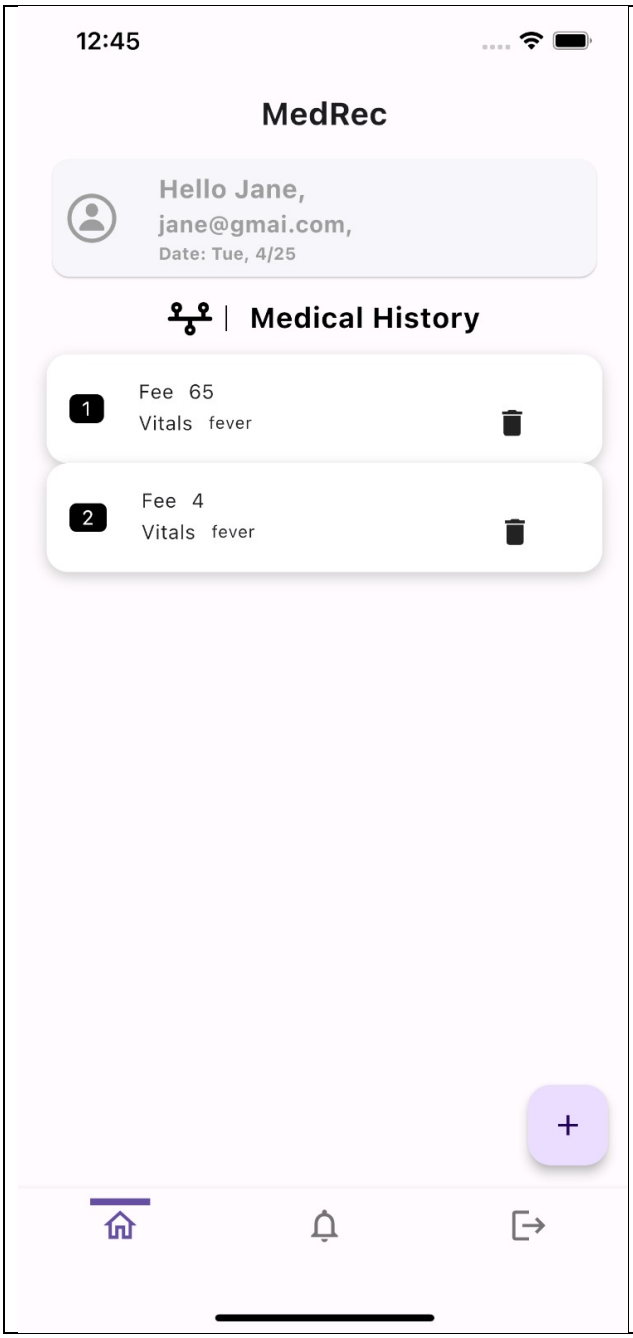
Use your key gotten from truffle suite and proceed to select your role

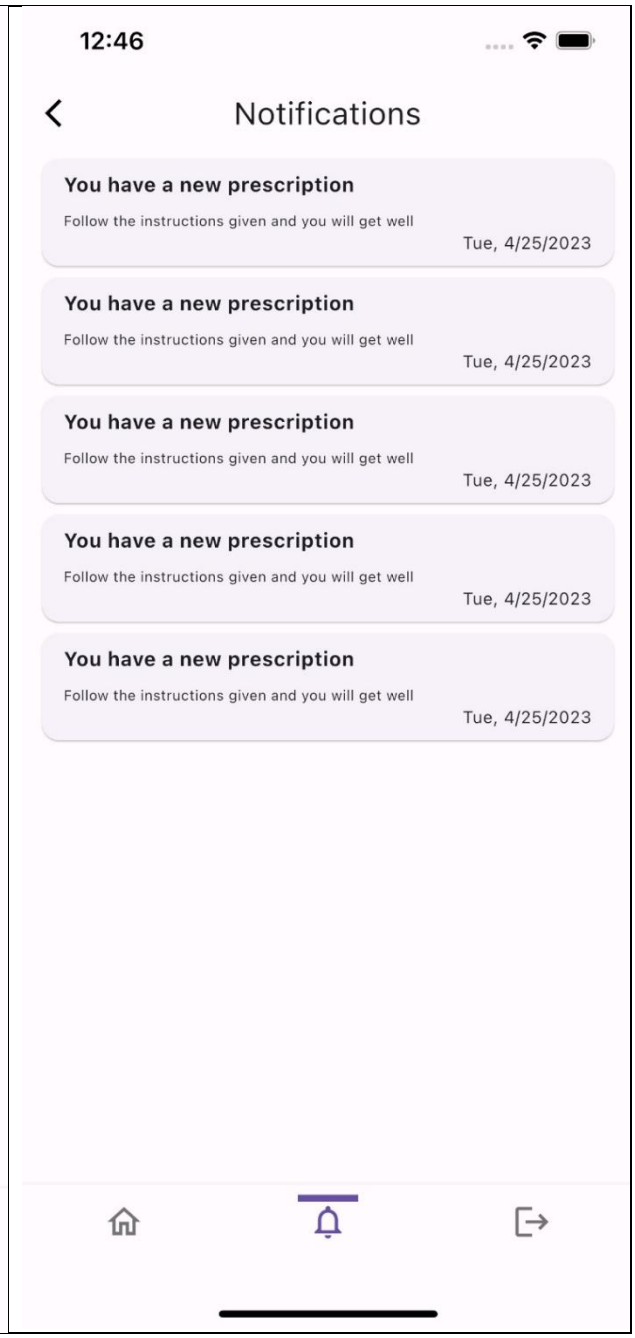
Proceed to the patient screen which as for now is empty. You need to add records when signed in as a doctor and the patient will manage to access the information in real-time.

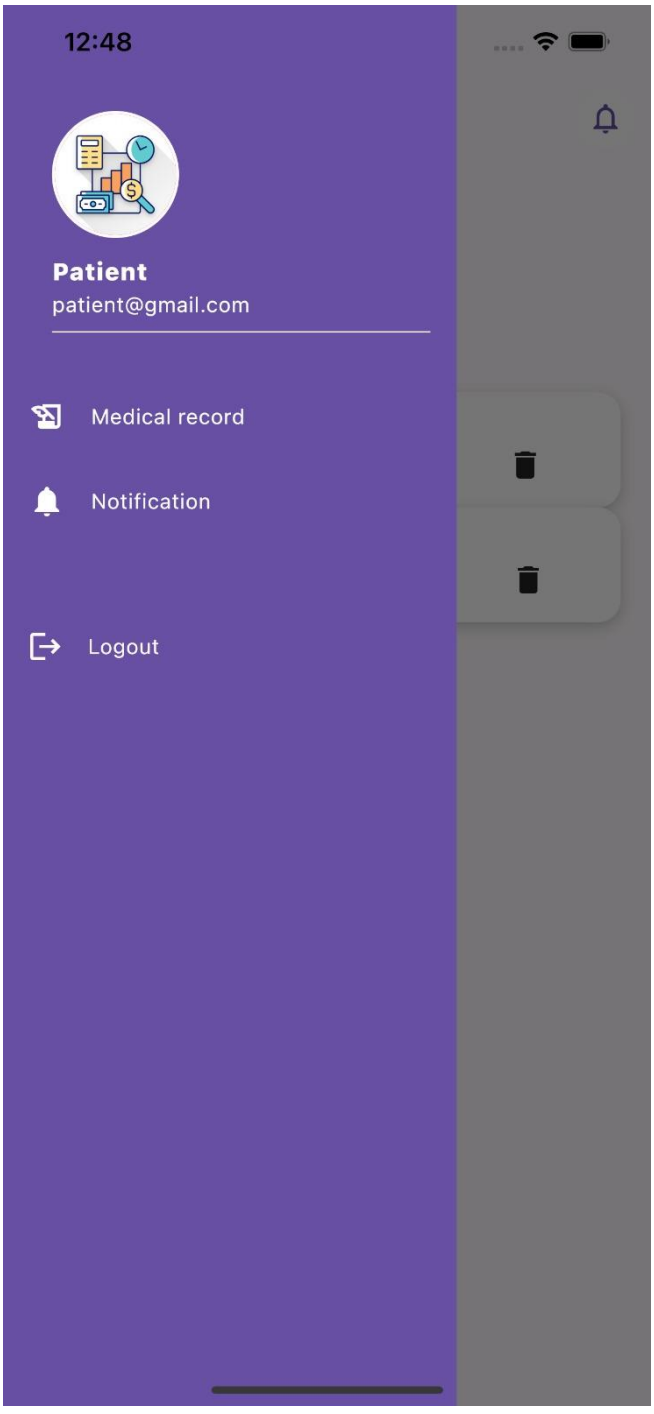
As shown in the figures below.




12:46
Record 1
65
London
Notes
Vomiting
Vitals
fever
Medicines
pain killers
Advice
2*4
12:46
Add Record Panel
Update Fee
Patient
Tap to Show Patients
Fee (In €) |
Change Fee
Update
Give Prescription
Address |
Patient Address
Notes |
Vitals |
Medicines |
Advice |
Send


12:45
MedRec
Hello Jane,
jane@gmai.com,
Date: Tue, 4/25
Medical History
1
Fee 65
Vitals fever
2
Fee 4
Vitals fever


12:46
Notifications
You have a new prescription
Follow the instructions given and you will get well
Tue, 4/25/2023
You have a new prescription
Follow the instructions given and you will get well
Tue, 4/25/2023
You have a new prescription
Follow the instructions given and you will get well
Tue, 4/25/2023
You have a new prescription
Follow the instructions given and you will get well
Tue, 4/25/2023
You have a new prescription
Follow the instructions given and you will get well
Tue, 4/25/2023



Github Link to the Project

Here is the github link to the project link