

APNAFOOD

Guide to use different tools and technologies.

Table of Contents

-Technology Stack

1. Github
2. MongoDB Atlas
3. Contabo Remote Server
4. DLT
5. React Native and Android Studio Setup

TECH STACK – Apna Food

MOBILE APP



App Hosting



CLOUD PLATFORM



Server/service
s hosting



SMS – DLT : CISCO



WEBSITE



Code hosting / development



Web Framework



Payment Gateway-DBS



1) GITHUB

Login details -

Email - info@mithranjali.org.in

Password - Orayiram@2020

After sign in using the above credentials, we can see a repository called 'apnaFood'.

The 'apnaFood' repository is the codebase for the mobile app.

The repository has different branches for frontend screens and backend api. Switching to any of the branches gives access to the code.

mithranjali / apnaFood Private

<> Code Issues Pull requests 1 Actions Projects Security Insights Settings

backend 12 branches 0 tags Go to file Add file <> Code

This branch is 45 commits ahead of main. Contribute

sripryamaturi	error fixed	a1524e0 on May 1	59 commits
.env	take order		2 months ago
.gitignore	added unique user validation		3 months ago
README.md	Initial commit		4 months ago
apnaFood.py	error fixed		2 months ago

README.md

apnaFood

2) MongoDB Atlas

The cloud database we are using is MongoDB Atlas.

Login details -

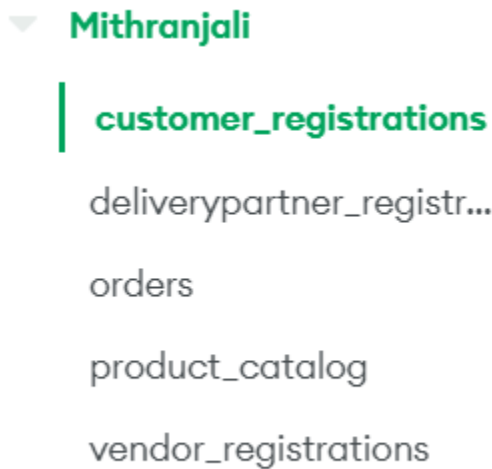
Email - info@mithranjali.org.in

Password - Orayiram@2020

Upon login, we can find cluster 0.

Cluster0 Connect View Monitoring Browse Collections ...

Click on browse collections to find our database and check all the collections.



These are the collections one can find under the database 'Mithranjali'. Click on any of the collections to check the data inside it.

Mithranjali.vendor_registrations

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 3.27KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns ⓘ Aggregation Search Indexes Charts ●

INSERT DOCUMENT

Filter ⓘ Type a query: { field: 'value' } Reset Apply More Options ▶

QUERY RESULTS: 1-6 OF 6

```
_id: ObjectId('63f74b1909783f15d01ce490')
username: "sripriya"
email: "sripriyamaturi8@gmail.com"
mobile: "919701044584"
password: "abcd"
Organization Details: Object
Bank Details: Object
Address Details: Object
Status: "Rejected"
Stage: "Sub District"
```

```
_id: ObjectId('63fcc26fc72dce54a3327cb5')
username: "Prasanna"
```

In a similar way, new collections can be created and existing collections can be modified.

The above data can be accessed via code, by connecting to the Mithranjali database. The following lines of code perform the database connection task.

```
client =
pymong
```

```
o.MongoClient("mongodb+srv://sripriya:"+urllib.parse.quote("Orayiram@2020")+"@cluster0.once1vv.mongodb.net/?retryWrites=true&w=majority",
server_api=ServerApi('1'), connect = False)
mydb = client.Mithranjali
```

3) Contabo Remote Server

The application backend is hosted on a remote server.

Here are the details of the server (ip address) -

www.apnafood.org.in

130.185.118.141

To connect to the server remotely, we use SSH.

The command is - ssh root@apnafood.org.in

Enter the password - Orayiram@2020

```
C:\Users\mahes>ssh root@apnafood.org.in
root@apnafood.org.in's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

https://ubuntu.com/pro

      .--' \   .--' \
     ( (  \ .---./ ) )
      '---/o   o\---'
         {=   ^   =}
          >   -   <
-----" " \-----" "-----
/                               \
\  www.apnafood.org.in         /
\  130.185.118.141             \
\-----" " \-----" "-----
      ____)( ) (____ jgs
      (((__ ) (____))
Last login: Mon Jun 12 18:50:26 2023 from 49.37.144.108
root@www:~# |
```

On logged in, we can get into the apnaFood folder and get access to the backend FAST api.

The backend api is hosted. Using nginx we redirect all the api calls to the fastAPI running on a specific port.

4) DLT

The DLT is used to send SMS otp for user phone number verification.

To do this, the first step is to create a template of the message to be sent on the BSNL dlt platform.

<https://www.ucc-bsnl.co.in/>

Once the template is approved on the bsnl dlt, we can utilize the template and send SMS via textlocal.

ID	Template Name	Category	Sender names	Status	
<input type="checkbox"/> 1407167568077970014	ApnaFood_OTP	Service Implicit	MTRJLI	Active	

The above shown is our active template.

An api key needs to be created in the textlocal platform to use in the code - to send sms in the specific approved template.

Using this, the code snippet to send SMS looks like this -

```
otp=randint(1000,9999)
params = {'apikey':
'NTA2NDZhNjgzNDVhNGYzNTZiMzE2YTczNDQ2YzYxNzk=', 'numbers': phnnum,
'message' : 'Welcome to apnaFood by MITHRANJALI FOUNDATION.\nYour OTP for
registration is ' + str(otp) + '.', 'sender': 'MTRJLI'}
f = urllib.request.urlopen('https://api.textlocal.in/send/?'+
urllib.parse.urlencode(params))
return {'otp' : otp}
```

5) React Native and Android Studio Setup

For the React Native projects, we can either go with an Expo Go or a React Native CLI. The one used for this project specifically is React Native CLI and steps followed for installation and configurations.

> Install Node.js on local system.

You will need Node, the React Native command line interface, a JDK, and Android Studio.

Recommended Versions of the technologies used:

Node: >14

Java Development Kit: >11

Android Studio: >

If you're using the latest version of Java Development Kit, you'll need to change the Gradle version of your project so it can recognize the JDK. You can do that by going to {project root folder}\android\gradle\wrapper\gradle-wrapper.properties and changing the distributionUrl value to upgrade the Gradle version. You can check out [here the latest releases of Gradle](#).

> Install Android Studio

Download and install Android Studio. While on Android Studio installation wizard, make sure the boxes next to all of the following items are checked:

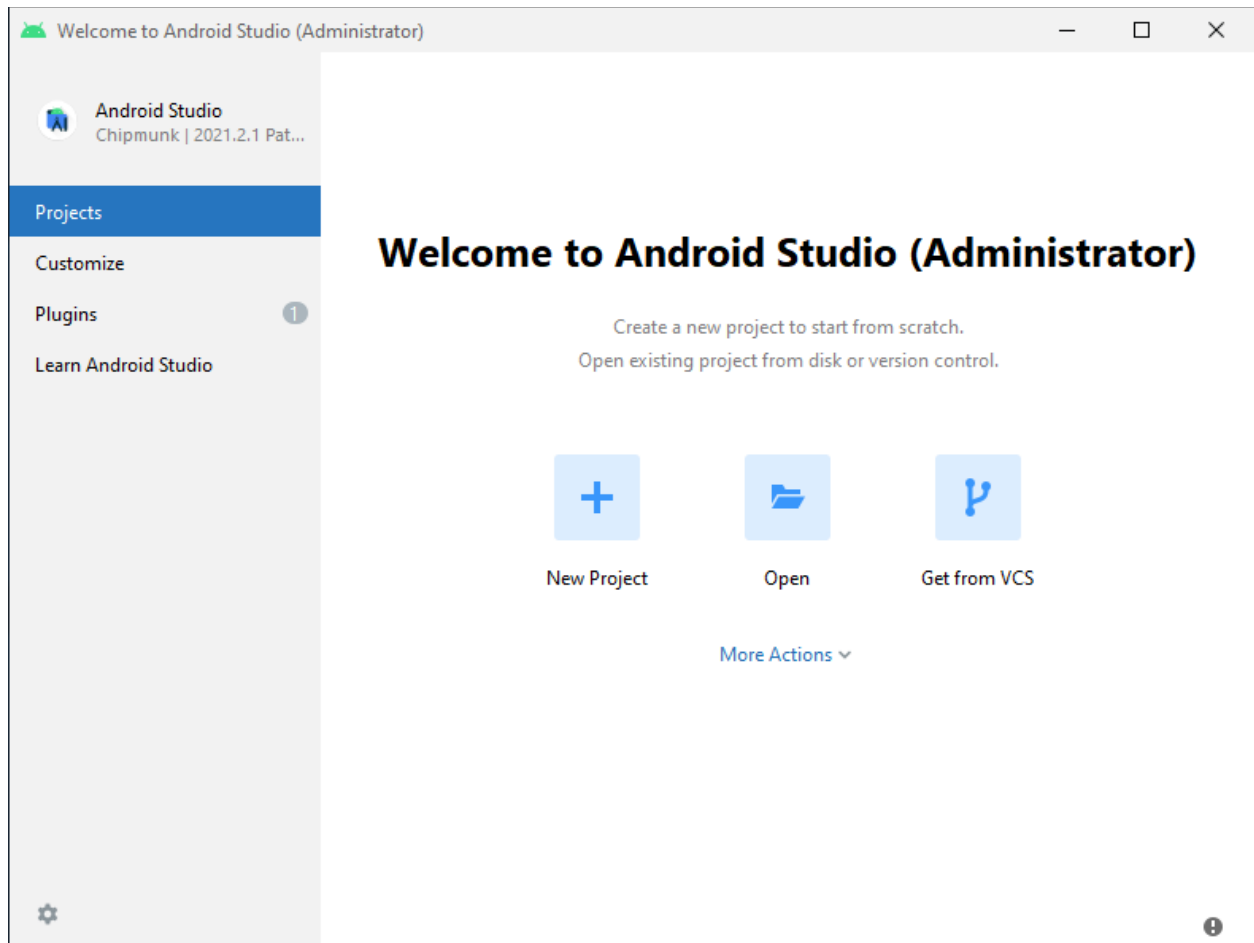
- **Android SDK**
- **Android SDK Platform**
- **Android Virtual Device**

Then, click "Next" to install all of these components.

> Install the Android SDK

Android Studio installs the latest Android SDK by default. Building a React Native app with native code, however, requires the **Android 13 (Tiramisu)** SDK in particular. Additional Android SDKs can be installed through the SDK Manager in Android Studio.

To do that, open Android Studio, click on "More Actions" button and select "SDK Manager".



The SDK Manager can also be found within the Android Studio "Preferences" dialog, under Appearance & Behavior → System Settings → Android SDK.

Select the "SDK Platforms" tab from within the SDK Manager, then check the box next to "Show Package Details" in the bottom right corner. Look for and expand the **Android 13 (Tiramisu)** entry, then make sure the following items are checked:

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

Next, select the "SDK Tools" tab and check the box next to "Show Package Details" here as well. Look for and expand the **Android SDK Build-Tools** entry, then make sure that **33.0.0** is selected.

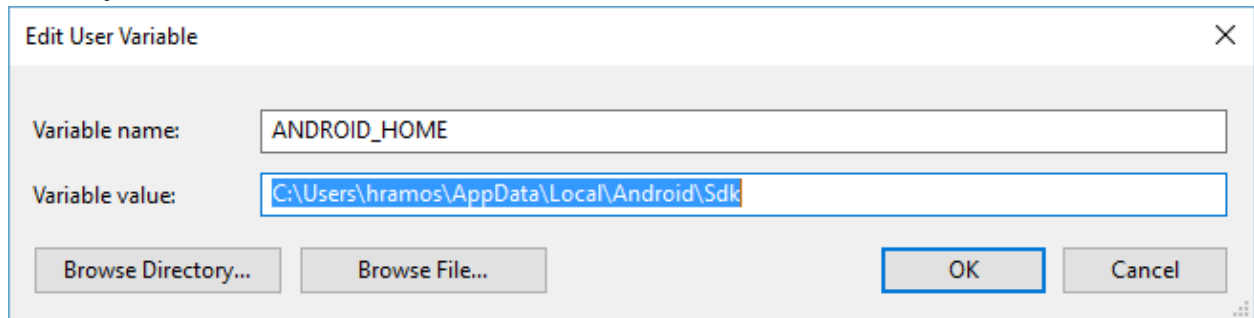
Finally, click "Apply" to download and install the Android SDK and related build tools.

> **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:



Edit User Variable

Variable name:

Variable value:

The SDK is installed, by default, at the following location:

```
%LOCALAPPDATA%\Android\Sdk
```

> 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.

The default location for this folder is:

```
%LOCALAPPDATA%\Android\Sdk\platform-tools
```

Preparing the Android device

You will need an Android device to run your React Native Android app. This can be either a physical Android device, or more commonly, you can use an Android Virtual Device which allows you to emulate an Android device on your computer.

Either way, you will need to prepare the device to run Android apps for development.

Using a physical device

If you have a physical Android device, you can use it for development in place of an AVD by plugging it in to your computer using a USB cable and following the instructions [here](#).

Using a virtual device

If you use Android Studio to open [./mithranjali/android](#), you can see the list of available Android Virtual Devices (AVDs) by opening the "AVD Manager" from within Android Studio. Look for an icon that looks like this:



Running React Native application

Step 1: Start Metro

First, you will need to start Metro, the JavaScript bundler that ships with React Native. Metro "takes in an entry file and various options, and returns a single JavaScript file that includes all your code and its dependencies."—[Metro Docs](#)

To start Metro, run `npx react-native start` inside your React Native project folder:

```
npx react-native start
```

`react-native start` starts Metro Bundler.

If you use the Yarn package manager, you can use `yarn` instead of `npx` when running React Native commands inside an existing project.

If you're familiar with web development, Metro is a lot like webpack—for React Native apps. Unlike Kotlin or Java, JavaScript isn't compiled—and neither is React Native. Bundling isn't the same as compiling, but it can help improve startup performance and translate some platform-specific JavaScript into more widely supported JavaScript.

Step 2: Start your application

Let Metro Bundler run in its own terminal. Open a new terminal inside your React Native project folder. Run the following:

```
npx react-native run-android
```

If everything is set up correctly, you should see your new app running in your Android emulator shortly.



`npx react-native run-android` is one way to run your app - you can also run it directly from within Android Studio.

Modifying your app

Now that you have successfully run the app, let's modify it.

- Open `App.js` in your text editor of choice and edit some lines.
- Press the `R` key twice or select `Reload` from the Developer Menu (`Ctrl + M`) to see your changes