

## Saral Frontend Setup:

### Tech Stack

- React Native [\(Link\)](#)
- Native Module
- JavaScript

### Suggested Team composition:

| Type               | Language / Skills                                                                                                   | Experience | Full-time people |
|--------------------|---------------------------------------------------------------------------------------------------------------------|------------|------------------|
| Frontend Developer | React native,native module,Flrebase,push-n otification,firebase analytics and crashlytics,github,android studio     | 3+ years   | 1                |
| Backend Developer  | NodeJs, JavaScript, Express.js,Postman, MongoDB Compas, SonarQube, Github, Jest Unit testing, Swagger documentation | 3+ years   | 1                |
| DevOps             | Docker, AWS, KUBERNETES,jenkins,S CM tools, Grafana, Prometheus,EFK stack, Terraform                                | 3+ years   | 1                |
| QA                 | UAT                                                                                                                 | 3+ years   | 1                |

### Software requirements and Prerequisites:

- NodeJS (Recommended version v16.9.1 LTS)
- NPM
- [OpenJDK 11](#)
- [Android Studio \(Android Studio Electric Eel | 2022.1.1 Patch 2 February 27, 2023\)](#)
- Vysor

- NDK old Version **20.0.5594570** needed to be compatible. Use command `sdkmanager 'ndk;20.0.5594570'` to switch to ndk **20.0.5594570**. (This command is handy on Mac or Linux)

## Other tools used

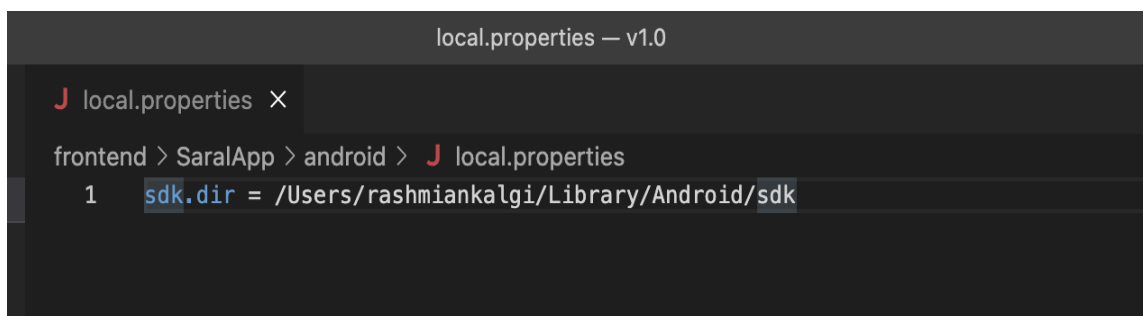
- [Microsoft VoTT\(Visual Object Tagging Tool\)](#) [VoTT Project Setup Video](#)
- [Jupyter Notebook](#) for transforming VoTT raw json to Saral ROI json. Refer to source repository path **specs\v1\README.md** file for more details. You may consider installing the extension in MS Visual Studio Code.
- PostMan or any other REST client tool to invoke/test Rest APIs

## Source Code references

- Open Terminal and clone source code git clone <https://github.com/Sunbird-Saral/Project-Saral.git>.
- Change Directory to Project-Saral/ folder and switch to release tag as per release notes. `git checkout tags/<tag_name>`
- `$FRONTEND_FOLDER = Project-Saral/v1.0/frontend`

## Installation and Setup

- [Setting up the React Native development environment](#) **Note:** Choose React Native CLI Quick Start tab for setup instructions.
- Download [NDK 20.0.5594570](#) and extract archive to **\$ANDROID\_HOME\ndk** folder.
- In Android Studio, navigate to File\Project Structure\SDK Location and set Android NDK location to **\$ANDROID\_HOME\ndk\android-ndk-r20** (extracted in the last step).
- Make sure `sdk.dir` paths are set properly in the file `local.properties`.



The screenshot shows the 'Project Structure' dialog in an IDE, specifically the 'SDK Location' tab. The title bar reads 'local.properties — v1.0'. The file explorer on the left shows the path 'frontend > SaralApp > android > local.properties'. The main text area shows the content of the file:

```
1  sdk.dir = /Users/rashmiankalgi/Library/Android/sdk
```

If the file doesn't exist, create in the given path.

frontend\SaralApp\android\local.properties.

- Register on [Google Firebase](#), add the Saral project on the firebase dashboard and download the corresponding **google-services.json**.

If the project exist use the google-services.json from Project settings.

- Place downloaded google-services.json under **frontend/SaralApp/android/app** folder.
- Once Vysor and AVD are configured as per the instructions, follow the below steps to bring up the application.
- Open a terminal in **frontend/SaralApp** and run npm i the command.
- cd android && ./gradlew clean && cd .. (use "gradlew clean" for windows machine)
- Make sure backend **BASE\_URL** is configured in **frontend/SaralApp/src/configs/config.js** file.
- npx react-native run-android
- Successful frontend deployment should show Saral OCR login screen on AVD or USB connected Android device

## Generate keystore file

- For generating keystore file run this command
  - **keytool -genkey -v -keystore my-release-key.keystore -alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000**

## Generating APK from source code

- open Terminal in **frontend/SaralApp/android** folder
- Run command. **./gradlew clean**
- APK Signing can be enabled using the below Gradle files. Make sure the Keystore file is to be used for signing placed in **frontend/SaralApp/android/app** folder.
- Change alias and password in **frontend/SaralApp/android/gradle.properties**

MYAPP\_RELEASE\_STORE\_FILE=my-upload-key.keystore

MYAPP\_RELEASE\_STORE\_PASSWORD=changeit

MYAPP\_RELEASE\_KEY\_ALIAS=hwrecog-key-alias

MYAPP\_RELEASE\_KEY\_PASSWORD=changeit

- Open Terminal in **frontend/SaralApp/android** folder and run command.  
**./gradlew assembleRelease**

- You can find the release apk file in  
Project-Sara1/v1.0/frontend/Sara1App/android/app/build/outputs/apk/release folder

## Generate AAB (App bundle) from source code

- open Terminal in **frontend/Sara1App/android** folder
- Run command. **./gradlew clean**
- APK Signing can be enabled using the below Gradle files. Make sure the Keystore file is to be used for signing placed in **frontend/Sara1App/android/app** folder.
- Change alias and password in **frontend/Sara1App/android/gradle.properties**

MYAPP\_RELEASE\_STORE\_FILE=my-upload-key.keystore

MYAPP\_RELEASE\_STORE\_PASSWORD=changeit

MYAPP\_RELEASE\_KEY\_ALIAS=hwrecog-key-alias

MYAPP\_RELEASE\_KEY\_PASSWORD=change it

- Open Terminal in **frontend/Sara1App/android** folder and run command  
**./gradlew assembleRelease**
- You can find the release apk file in  
Project-Sara1/v1.0/frontend/Sara1App/android/app/build/outputs/apk/releases/app-release.aab folder

## Debugging/running saral from android studio

- Open **frontend/Sara1App/android** folder from Android Studio.
- Run npx react-native start command from Terminal in **frontend/Sara1App** the directory.
- Run adb reverse tcp:8081 tcp:8081 from another terminal
- Now click on the Run or Debug button from Android studio.
- When you are debugging, have debug breakpoints in android code as needed for troubleshooting.

## Google Play Store App Publish

Refer to [Developer Program Policy](#) for more details. These policies can be changed by google time to time, so please refer latest policies before publishing App in playstore.

**Privacy Policy** for App to be listed in the playstore to be generated and publish this policy of the app as a URL. The same Privacy Policy URL needs to be added in the play store "Store Listing" section of your application before publishing the Application.

**Misleading Claims** Google team reviews the app content including logos , images , content to see if any misleading claims. For example "government affiliation or authorization to facilitate government services through your app". A [Advance Notice Form](#) to be submitted for any such affiliations or claims for google team to review it.

## Integrating firebase

### Create a Firebase Project

- Go to the [Firebase Console](#).
- Click "Add Project" and follow the on-screen instructions to create a new Firebase project.
- Add google.json file which is created by firebase in folder **frontend/saralapp/android**
- In saral app code all firebase event present in <https://github.com/Sunbird-Saral/Project-Saral/blob/main/v1.0/frontend/SaralApp/src/utlis/Analytics.js>

### sample example to add firebase event

```
export async function loginEvent(username) {  
  await analytics().logEvent("successfullLogin", {userName:username});  
}
```

```
9  
10  
11 export async function loginEvent(username) {  
12   await analytics().logEvent("successfullLogin", {userName:username});  
13 }  
14  
15 export async function SchoolDetailSubmit(username) {  
16   await analytics().logEvent("schoolDetailSubmit", {schoolDetailSubmituser:username});  
17 }  
18  
19  
20 export async function markAttendance(value) {  
21   await analytics().logEvent("attendance", {attendance:value});  
22 }  
23  
24 export async function markAttendanceNext(username) {  
25   await analytics().logEvent("attendanceNext",{attendanceNextUser:username});  
26 }  
27  
28 export async function SaveInDbEvent(username) {  
29   await analytics().logEvent("saveAllScanClick",{saveInDbuser:username});  
30 }  
31  
32 export async function onScanButtonClickEvent(username) {  
33   await analytics().logEvent("clickOnScanButton",{scanButtonClickuser:username});  
34 }  
35  
36 export async function EditMarksManually(username) {  
37   await analytics().logEvent("EditMarksManually",{EditMarksManually:username});  
38 }  
39  
40 export async function saveLocalAfterScan(username) {  
41   await analytics().logEvent("SaveInLocalStorage",{saveLocaldata:username});  
42 }
```