QUICKSTART

This document helps new testers and developers to quickly understand the procedures for setting up the environment; different components in app and finally, running the app

Contents

- Frontend
- Backend
- Running

Frontend

1.1 Environment Setup

Follow this Installation process for running the frontend-Client part of the code in your Operating System. It is recommended to use Ubuntu or any Debian system but not necessary.

1.1.1 Node JS:

You have to install a node in your local machine for installing all the dependencies.

1.1.1.1 For Windows:

- Go to Node Js downloads page and download MSI or any other package according to your system requirements(32 bit /64 bit)

- Once Downloaded, go to the Downloaded location and open the file; Follow the instructions on the screen. Once it is done, you have to **restart your PC**.

- Verify the installation by typing the following commands in the command prompt or PowerShell.

```
> node -v
> npm -v
```

Both the commands should display the currently running version of node in your Local Machine. If it didn't, there must be a problem with the installation.

1.1.1.2 For Linux (Ubuntu OR Debian):

Run the following command in the terminal and check the installation by running the commands 'node -v' and 'npm -v'

```
sudo apt install nodejs npm
```

1.1.2 Expo CLI:

Open your terminal or command prompt and run the command in your terminal and wait for the installation process to complete.

```
npm install --global expo-cli.
```

-{Linux users might have to use sudo in case of issues.}

Run `expo --version` to verify the installation.

NOTE: Android users can use the expo app without signing up. But IOS users should be signed in to expo on both their PCs and mobiles. You can register for the expo on this site expo.io .

Once you created an account you can login via command `expo login` in your command prompt, and you can login in your iphone via GUI provided in the expo app.

1.2 Files:

Note: It is Better to have an idea on react concepts.

- **App.js:**
  The central component in the code which collects all the individual screens in our app and render them through a stack Navigator

- **Constants.js:**
  Contains all the environment variables for our app. All the changes should be performed only in this file. (host and port changes)

- **Home.js:**

The home screen component for the app is implemented in this file. All the functionalities of the homepage like copy, delete, scan, etc are initiated from here.

- **Detected_Images.js and ImageView.js:**
  Both these files are responsible for rendering the screen containing the list of detected items in a scrollView.

- **Input_order.js**
  This file is responsible for rendering the screen which takes the order input(item and quantity) from the user.

- **Settings.js**
  This file is responsible for the “editable option functionality” which allows the user to change the ability of the user to edit the detected quantities.

- **Colours.js**
  A file for storing all the color codes used in the app.

- **newSummary.js**
  This File is responsible for displaying the intermediate detections by using the data given by the backend model. The functionalities like delete, save, AddImage are also implemented here.

- **orderInfo.js**
  This file is responsible for displaying the order details screen which shows up on clicking a particular order from the home Screen. Please note that the new scan functionality is implemented here in the app.

- **app.json, package.json, package-lock.json, babel.config.js, .expo**
  All these files are responsible for compiling and monitoring the working of the app which are managed by the expo. Never delete these files.

- **Camera.js**
  Displays Camera with Recording and Picturing option. To start Recording single tap the circle in the bottom and do the same to stop the recording. Long press the circle to capture a picture.

- **Processing.js**
  Send the captured picture or video to the host mentioned in **Constants.js** to the route /imagedetect or /videodetect depending on the MIME type.

- **Detections.js**
  Acts like a navigation page between the Camera and Processing page.

- **Network.js**
  Checks the availability of the Internet while sending the request to the server.

Backend

2.1 Environment Setup

Follow this Installation process only if you want to run the code in your local machine(localhost). You can skip this if you are using an AWS server and go to 3.3.

2.1.1 Python3 installation :

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.8
python --version
```

2.1.2 Virtual Environment Installation :

```
sudo apt update
```

```
sudo apt install python3-venv
```

2.1.3 pip and pip3 Installation :

```
sudo apt update
sudo apt install python3-pip
pip3 --version
```

2.1.4 Python packages installation :

Create a virtual environment by using the command.

```
python3 -m venv “environmentname”
```

Without double quotes , and activate the environment by using the command in the directory containing the environment folder.

```
source environmentname/bin/activate
```

Once the environment is activated, run the following commands one by one.

```
pip install Flask
pip install pymongo
pip3 install numpy
pip install opencv-python
pip install Pillow
pip install sk-video
pip install matplotlib
pip install --no-cache-dir torchvision
pip install yacs
pip3 install -U scikit-learn scipy matplotlib
pip install scikit-image
pip install Flask-Cors
sudo apt-get install python3-opencv
```

You can deactivate the environment by using the command “`deactivate`”

2.1.5 MongoDB installation :

Follow the MongoDB official documentation here to install MongoDB in your localhost
And at last, don't forget to run the command

```
sudo systemctl enable mongod
```

To test whether you have installed it correctly or not try to run

```
sudo mongo
```

2.2 Files

This section gives some insight into some important locations in the backend folder

- **data/demo** : Location of the folder where one needs to insert class images in the format of **data/demo/class-image-1/img1.png**, **data/demo/class-image-1/img2.png**, **data/demo/class-image-2/img1.png**, etc.

- **app.py :** Main Flask server file that contains all routes of the server

- **demo.py :** A demo file for explaining the working of model, inputs class images and aile images, and outputs 0.png in the root folder (ie backend), but this has nothing to do with **app.py** that contain routes that process the model, for this purpose we designed two special files which will be discussed below.

- **demo_image.py** : a modified version of demo.py which contains a function for processing image files.

- **demo_video.py** : a modified version of demo.py which contains a function for processing video files.

- **osd/utils/visualization.py** : This contains some utility functions needed for above mentioned *demo* files to run. We suggest not to edit it, but sometimes if you need to know about how the detected image is showing labels, ids, percentages, or need to change the labels this is the place to do it.

Running

3.1 Frontend

Navigate to the frontend directory and run the command, to install all the dependencies for the app UI.

```
npm install
```

This will create **node_modules** folder in the frontend directory, this contains the code for all the dependencies, above command, need to be executed if the frontend folder doesn't contain **node_modules** folder, Generally while exporting the project across the devices, the developer removes the **node_modules** folder and guides to execute **npm install,** this fetches the dependency names from **package.json** and places it in **node_modules** folder which is required to run the code and bellow commands. So executing the **npm install** once at the start of the project is enough, next time you can directly proceed with the below commands

Once all the dependencies are installed you can start the metro bundler by running the command

```
expo start
```

In a couple of seconds, this opens up the page shown in the below diagram.

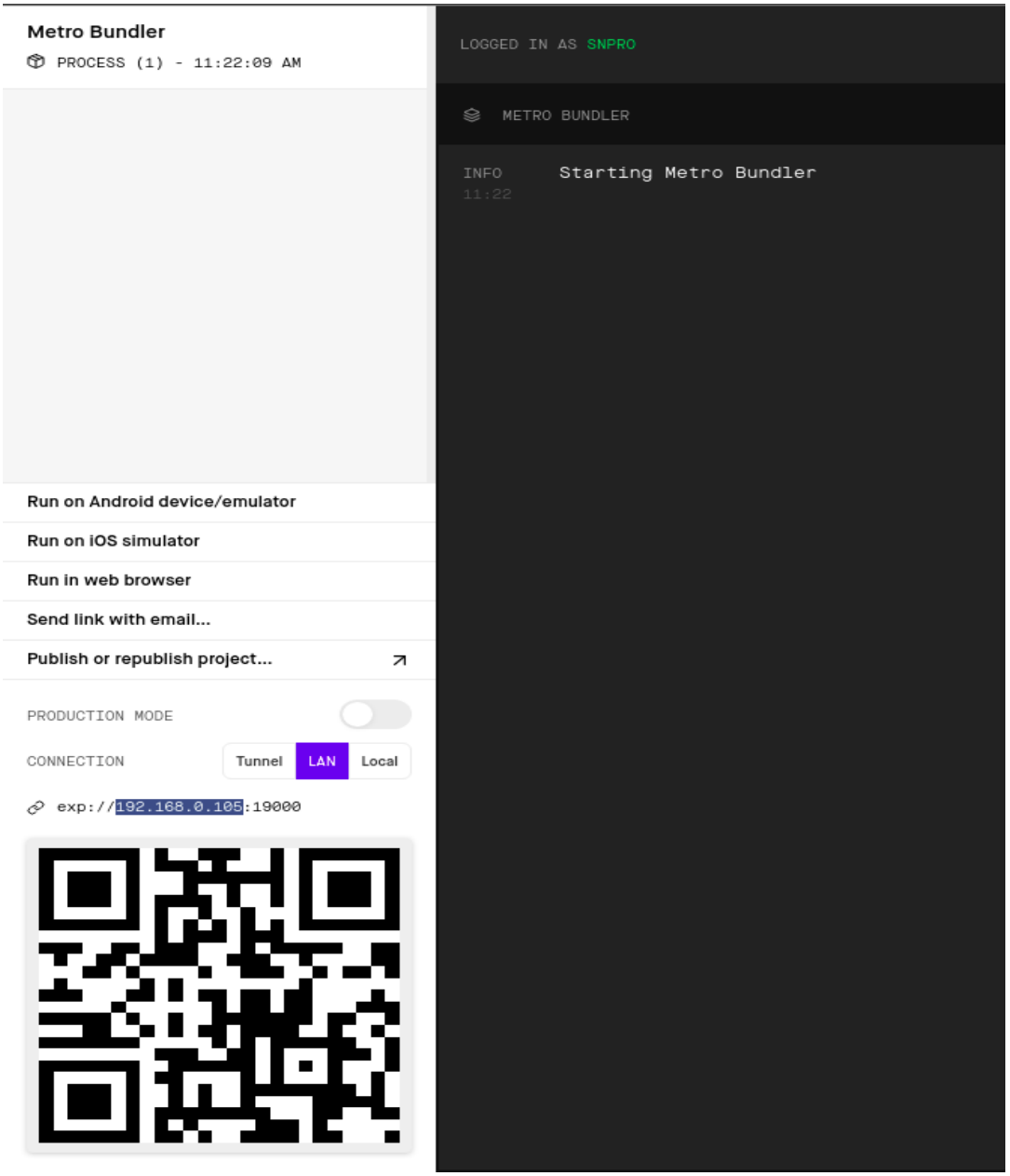

Figure1:Bundler

You can see the address 192.165.0.105 where your bundler is running. Make a note of this address which is the wifi ip address. This address will be different in your machine.

Now install expo go in your android or iphone for play store or app store. Connect the phone to the same wifi as the computer and scan the QR code from the expo app. You are good to go!

3.2 Running the Backend Local Machine:

This applies only when you want to run the backend in your localhost

- Navigate to the backend directory and activate the virtual environment

```
source <location_to_env>/bin/activate
```

- Run the below command to execute the server code in localhost 127.0.0.1 and at port 5000

```
    flask run
```

- But as you are executing and testing the app through your mobile, you cannot directly access the localhost of your laptop/pc. So follow the below steps
    - Connect both your mobile and laptop/pc to the same network
    - Once you have started the metro bundler in the frontend, you can see the host in which the expo app is running at the left bottom above the QRcode in the metro bundler, as we have explained above in **Fig1**. Copy that host
    - Now run the command `flask run -h <copiedhost>`

That's it, now the backend server is running on the desired host (that host is nothing but the IPv4 of your network)

- Now open **Constants.js**, you can see some variables header, host, port. Set the host to the **<copiedhost>** and port to 5000.

3.3 Running the Backend from Aws:

If you need to connect the frontend to the deployed backend in AWS, not to the localhost backend you need to specify the host as the public IPv4 of the deployed Amazon Ec2 and port as 80.