1. **OBJECTIVE**
2. **COMPONENTS**
   1. **HARDWARE**
      1. **Client**
      2. **Server**
   2. **SOFTWARE**
      1. **Client**
      2. **Server**
   3. **TESTING**
3. **FLOW CHART**
4. **ALGORITHM**
5. **SETUP**
   1. **CLIENT – WINDOWS**
   2. **CLIENT – RASPBERRY PI**
   3. **SERVER**
   4. **LABELME**
6. **CODE**
   1. **CLIENT CODE**
   2. **SERVER CODE**
7. **EXECUTION STEPS**
   1. **PARKING SPOT DETECTION CODE**
   2. **START SERVER**
   3. **CHECK DATA IN DATABASE**

1. **OBJECTIVE**

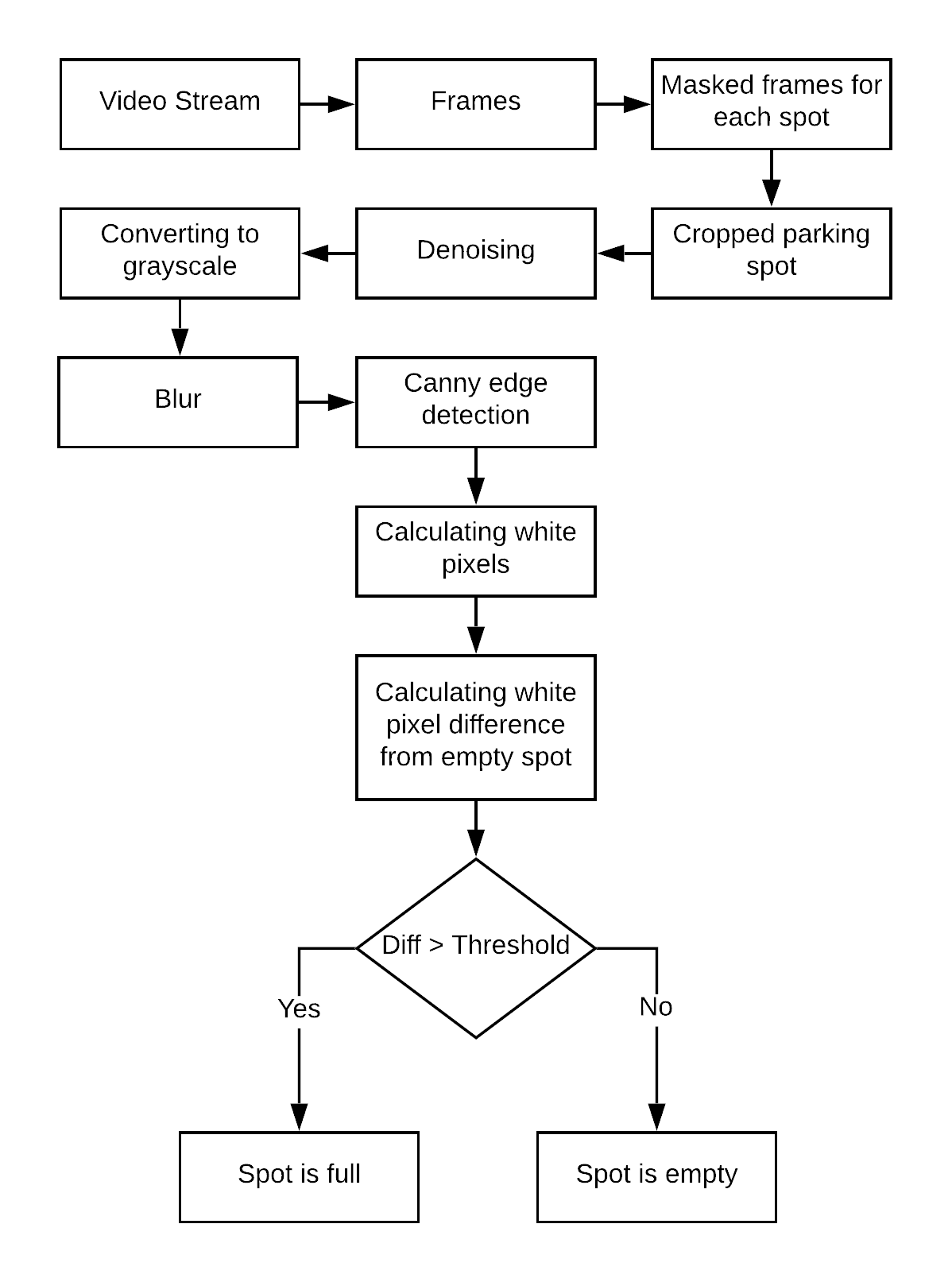
Finding an empty parking spot is difficult. Companies invest million dollars to automate this by installing an Infrared sensor at each parking spot to detect if it’s occupied by a vehicle or not. Cost of the this increases as each parking spot will require an IR sensor, drilling a hole to fit the sensor, a controller to transfer the data to server, and a power supply.

Our solution is bit different. We’ll use camera and image processing to detect the parking spot is empty or occupied. In our approach, we are using few IP cameras to cover all the parking spot view, Raspberry Pi/NUC to take video feed from the cameras and do image processing.

1. **COMPONENTS**
   1. **HARDWARE**
      1. **Client**

* Raspberry Pi
* Intel NUC
* IP Camera
* Router / Wifi Dongle
  + 1. **Server**
    - Windows Machine with minimum i5 processor and 8 GB Ram.
    - Internet connectivity
  1. **SOFTWARE**
     1. **Client**
* **OpenCV** – It is a computer vision library.
* **Labelme** – It is a graphical image annotation tool
* **Anaconda Navigator** – It is a GUI to manage and create environments and install OpenCV and Labelme.

1. **Server**
   * **Ngrok** – Ngrok creates a global link which can be used in our client code to share the results to the server. Ngrok will port forward the request to out server’s local host.
   * **VS Code** – Code editor.
   * **Node JS** – Javascript runtime environment.
   * **Mongo DB** – NoSql data base.
   * **Video Compressor & Trimmer –** Windows app to downscale the video quality to 480p.
   1. **TESTING**
      1. Laptop (Teamviewer, Labelme, Python, OpenCV installed)
      2. One Raspberry PI with camera and Raspberry PI power adapter
      3. One Raspberry PI and power adapter
      4. One NUC and power adapter
      5. 3 Power banks
      6. 4 LAN Cable with 1 long cable
      7. Many typing strips
      8. One router with power adapter and antenna
      9. Portable keyboard and trackpad
      10. Portable monitor
      11. Power extender cable
      12. Verizon hotspot
      13. Stepper / Ladder
2. **BELOW IS THE FLOW CHART OF THE ALGORITHM TO DETECT IF PARKING SPOT IS FULL OR EMPTY.**

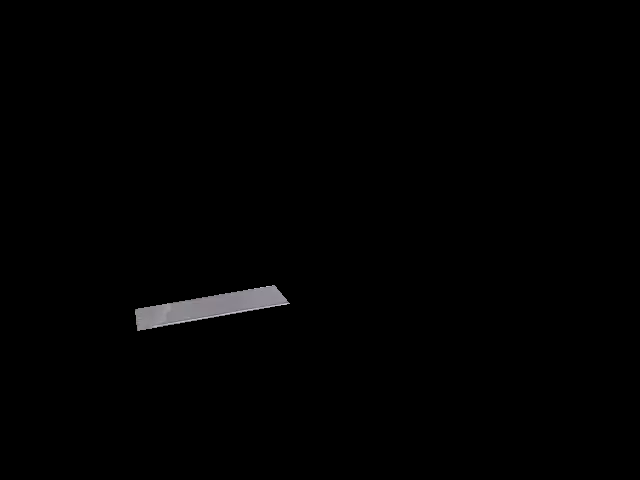
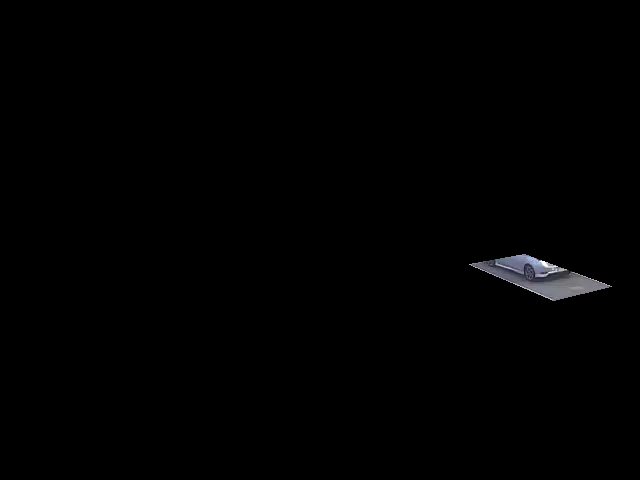


1. **ALGORITHM**

**This is the one frame taken from the video stream.**



**Frame is then masked for each parking spot.**

**Masked frame is cropped for each parking spot.**

**It is then denoised using an OpenCV function “fastNlMeansDenoisingColored”.**

**This color denoised image is converted to gray scale.**

**It is then blurred using OpenCV function “GaussianBlur”.**

**Canny edge detection for each spot is done using OpenCV function “Canny”.**

**White pixel of the spot is calculated which is then subtracted with white pixels of the empty spot.**

Current condition of spot 1 Current condition of spot 2

Empty condition of spot 1 Empty condition of spot 2

**White pixels of current condition spot – white pixels of empty condition spot**

For spot 1 For spot 2

330 – 329 = 1 575 – 302 = 273

**If the difference is greater than the threshold ie. 10; then spot is full; else spot is empty.**

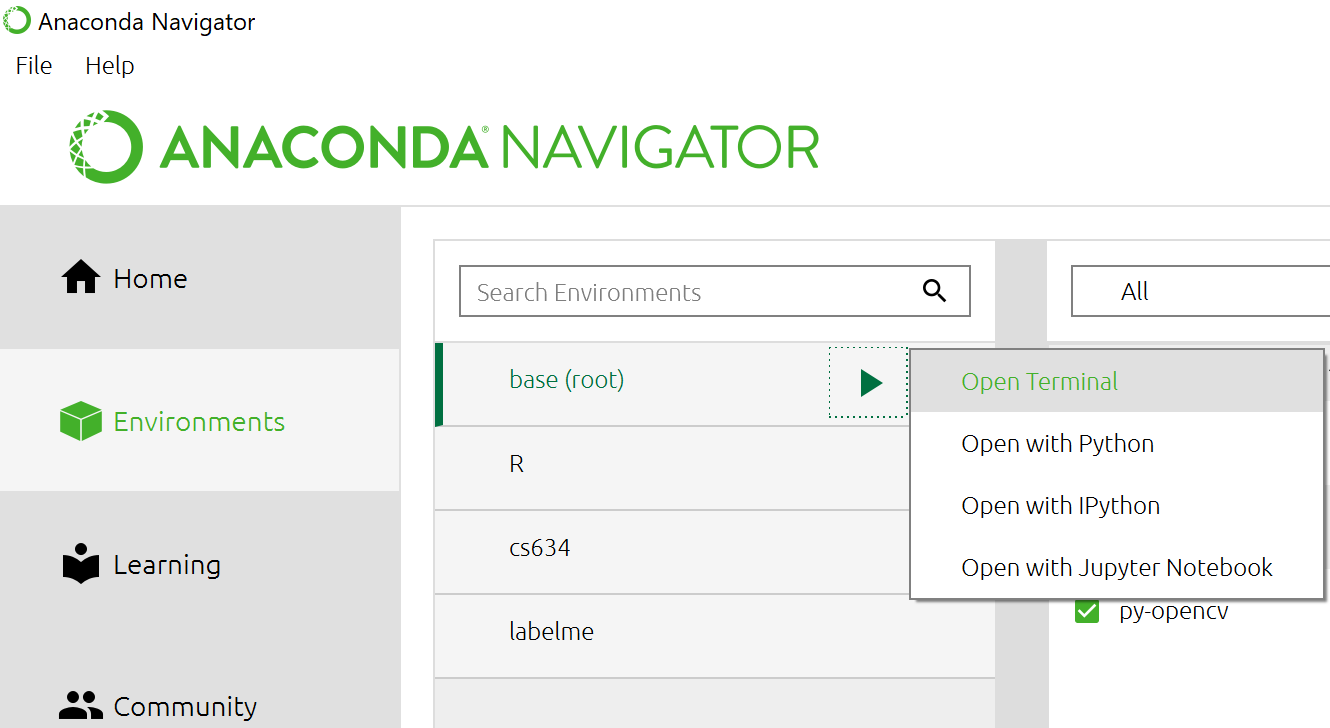
1 < 10 = Spot is empty 273 > 10 = Spot is full

1. **SETUP**
   1. **Client – Windows**

**1) Install Anaconda with Python 3.7+**

<https://www.anaconda.com/download/#windows>

**2) Open Anaconda Navigator. Go to Environments > base(root) > Open Terminal**



**3) In terminal execute below commands,**

conda update -n base -c defaults conda (This will update the packages list with latest versions for the base environment)

conda install -c conda-forge opencv (This will install OpenCV and its dependent packages)

pip install opencv-contrib-python (If the above command doesn’t work, use this)

conda install -c anaconda pillow (adds image processing capabilities to your Python interpreter)

pip install datetime (To add timestamp to the images)

pip install schedule (To install scheduler)

pip install requests (To send HTTP request to the server)

**4) Add path to environment variables (Use the path according to your installation)**

C:\Users\Nitz Mistry\Anaconda3

C:\Users\Nitz Mistry\Anaconda3\Scripts

C:\Users\Nitz Mistry\Anaconda3\Library\bin

C:\Users\Nitz Mistry\Anaconda3\Library\mingw-w64\bin

**5) Open cmd and execute below commands to confirm your installation**

python

>>>import numpy

>>>numpy.version.version

>>>import cv2

>>>cv2.\_\_version\_\_

* 1. **Client - Raspberry Pi**

**1) In terminal, execute below steps**

wget https://github.com/jjhelmus/berryconda/releases/download/v2.0.0/Berryconda3-2.0.0-Linux-armv7l.sh

chmod +x Berryconda3-2.0.0-Linux-armv7l.sh

./Berryconda3-2.0.0-Linux-armv7l.sh

conda create --name venv python=3.6.1 -y

source activate venv

conda install nump

conda install opencv

python

import numpy

numpy.version.version

import cv2

cv2.\_\_version\_\_

* 1. **Server**

**1) Install VS Code**

<https://code.visualstudio.com/>

**2) Install Node**

<https://nodejs.org/en/download/>

**3) Install Mongo DB**

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

**4) Install ngrok and configure according to the steps provided on website.**

<https://ngrok.com/download>

**5) Add ngrok.exe path in environment variable.**

**6) Extract Server.zip**

**7) Open Server folder in VS Code**

**8) Open terminal in VS Code and execute below commands in it**

export-startlayout -path C:\Users\sy423\Desktop\Server\layout.xml (Change you path accordingly)

npm init

npm install express --save

npm install nodemon --save-dev

npm install body-parser --save

npm install mongodb --save

npm install ejs --save

npm install -g multer --save

npm install multer --save

npm run dev

**9) Open another terminal in VS Code and execute below command**

ngrok http 3000

**10) This will start the server and create a link which you must use in the code. Server will send the results to this link and ngrok will port forward the results to our server’s localhost.**

**11) Open another terminal and execute below commands**

mongo (this will switch to mongo client)

use parkingspot (It’s the database)

db.createCollection("spotdata") (Collection is the table)

db.spotdata.find().pretty() (To check any existing document/rows available)

* 1. **Install Labelme on Windows**

**1) Open terminal from anaconda navigator and execute below commands**

conda create --name=labelme python=3.6 (Change python version according to the python available on system)

conda activate labelme

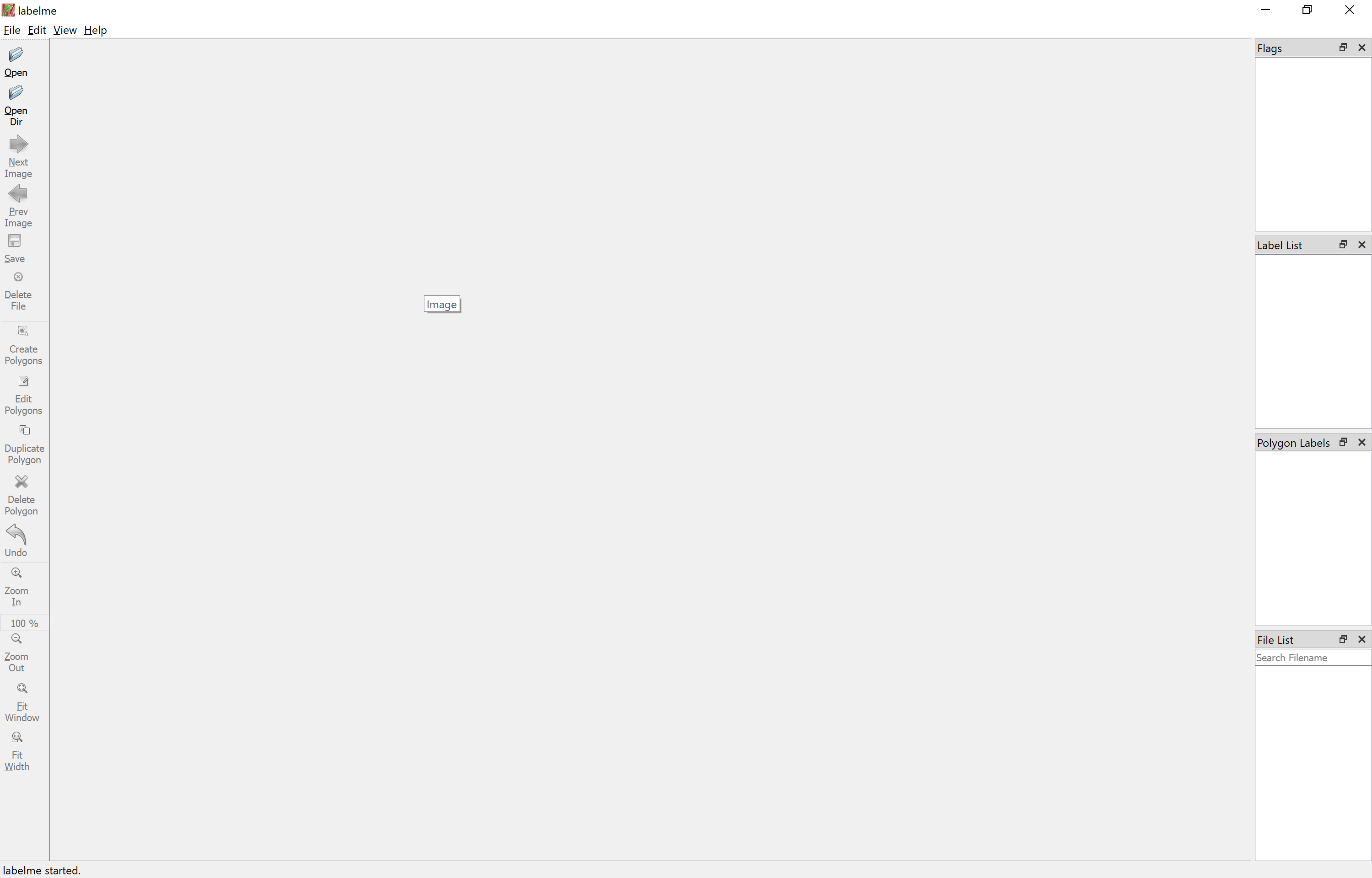
pip install pyqt5

pip install labelme

**2) Once Labelme is installed, Open Anaconda Navigator > Environments > labelme > Open Terminal**



**3) Type labelme**



Using Labelme is covered in the section 7 Execution steps.

**6. CODE**

1. **CLIENT CODE**

Live\_stream.py can be executed on NUC (Windows) and Raspberry Pi. In every 40 seconds, it connects with the live camera with the IP address provided, process and shares result with server and drops the camera connection.

****

Video\_file.py can be execute on NUC (Windows) and Raspberry Pi. Video\_file.py is same as Live\_stream.py but it connects with the video provided and shows the output using imshow.

****

1. **SERVER CODE**

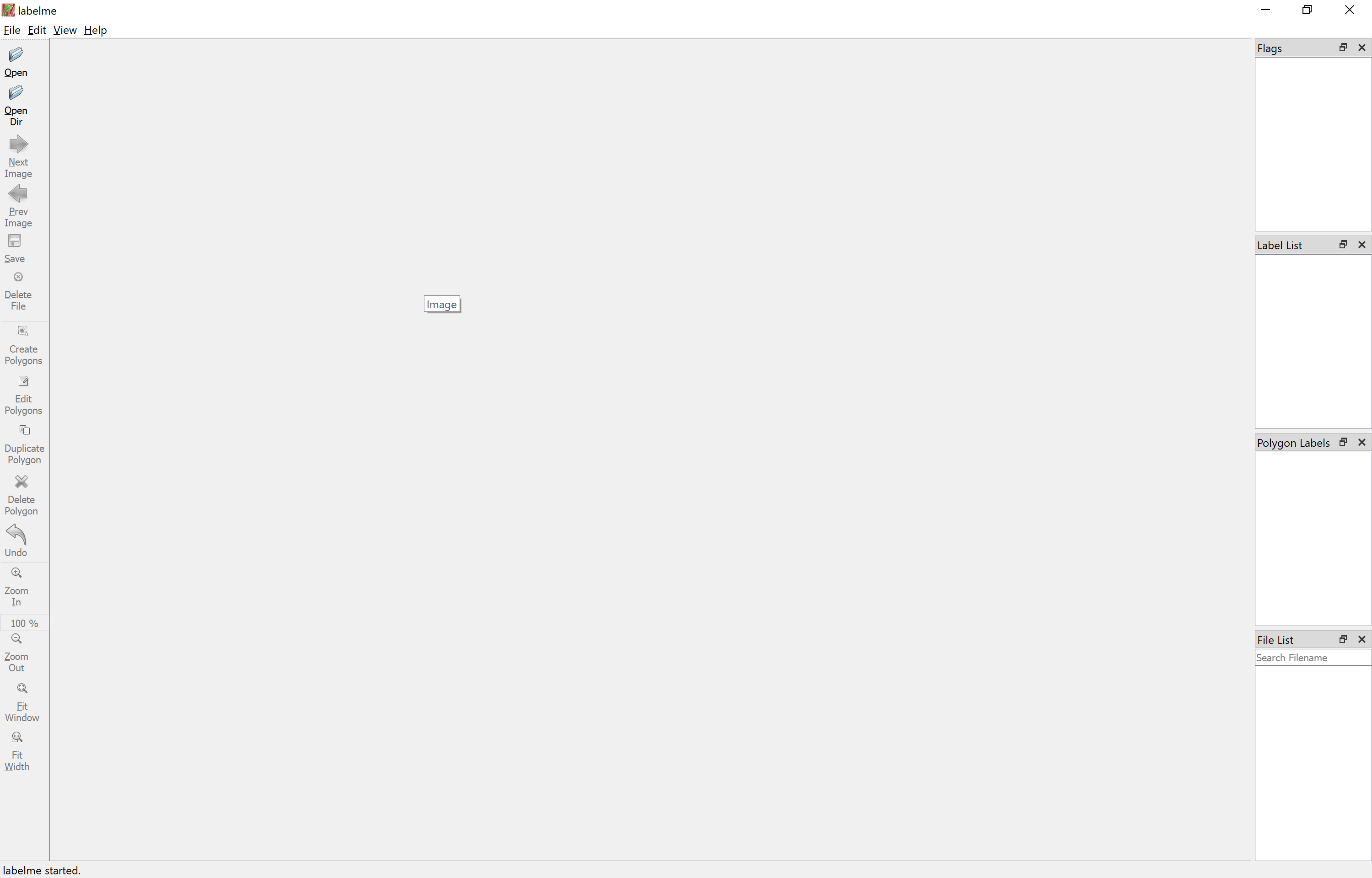
****

1. **EXECUTION STEPS**
   1. **PARKING SPOT DETECTION CODE**

Getting parking spot co-ordinates:

1) Execute “Live\_stream.py” to take a picture of the parking location and save it to local drive.

2) Open terminal/cmd and type “labelme”.



3) Click “Open” and select the parking location picture.

4) Click on “Create Polygon” and draw a polygon for each parking spot and give a label for each parking spot.



5) Click on File > Save as and select JSON in save as type.

6) Pass this JSON file path in below line of “AvgWhitePixels.py”

with open('C:\\Users\\Nitz Mistry\\Pictures\\Test\_15\_7\\snapshot.json') as empty\_spot\_data\_json:

7) Execute “AveWhitePixels.py” with “python AveWhitePixels.py” to calculate initial white pixels for empty parking spot. This will create a file named “spot\_data.json” in the path provided in “AveWhitePixels.py”

8) Pass the path of JSON files created in step 5 and 7 in “Parking.py”.

9) Run “Live\_stream.py” with “python Live\_stream.py”.

**NOTE:** Step 1-8 are configuration steps to get the parking spot coordinates and white pixel value when the spot is empty and must be executed only once.

1. **START SERVER**

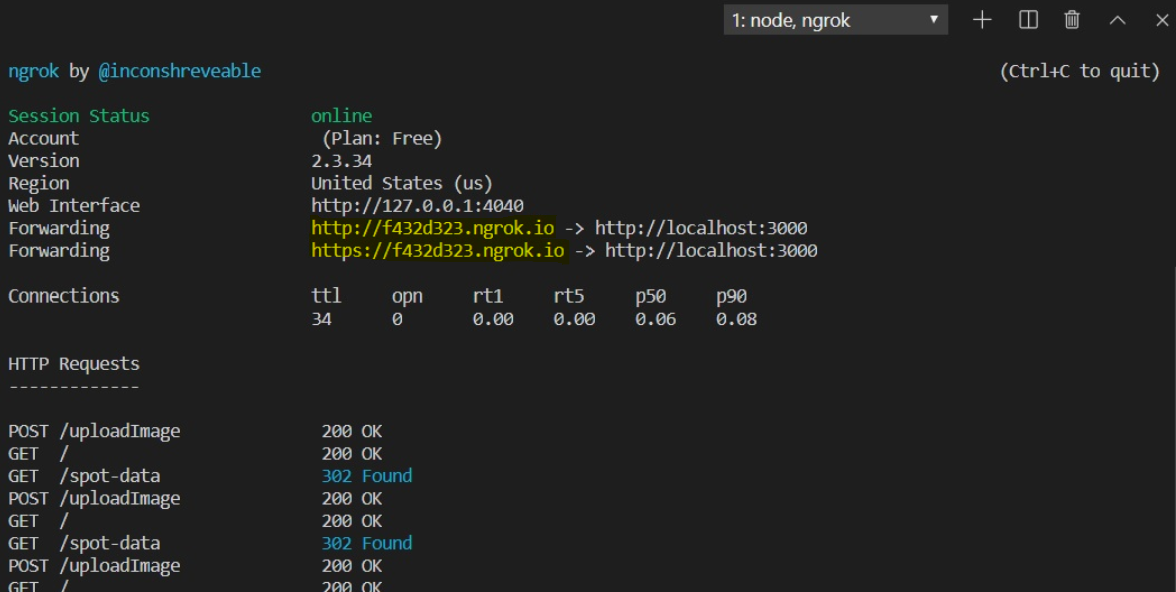
**1) Open terminal and execute below code**

npm run dev

**2) Open another terminal in VS Code and execute below command**

ngrok http 3000

**3) This will start the server and create a link (highlighted below) which you must use in the code. Server will send the results to this link and ngrok will port forward the results to our server’s localhost.**



1. **CHECK DATA IN DATABASE**

**1) Open another terminal and execute below commands**

mongo (this will switch to mongo client)

use parkingspot (It’s the database)

db.createCollection("spotdata") (Collection is the table)

db.spotdata.find().pretty() (To check any existing document/rows available)