# *Green Area Detector for Satellite Imagery*

****

**Introduction:**

Trees are vital. As the biggest plants on the planet, they give us oxygen, store carbon, stabilize the soil and give life to the world’s wildlife. They also provide us with the materials for tools and shelter. Green Area Detector is supposed to be a concept to fight Air Pollution and lack of plantation, forestation in any part of the world. It is designed to let you know about the surroundings and place where you live. Green area detector provides green and non-green part in any area of the world with percentage of green area.

**Use Cases:**

* To get the idea of plantation in any city or area.
* To provide awareness about the tree plantation.
* Making people to plant more trees and care them.
* Recommendations according to percentage of plantation.

**Methodology:**

We are using Google Earth Engine api for the Satellite imagery of Sentinel and Copernicus S2-R, further applies numpy and javascript to extract the green and non-green area in Python. It takes coordinates as user input with the help of drawings made on Leaflet maps in application. For application we have used Flask with python, js, html and css.

**Documentation:**

We’ve used Flask for the application, Google Earth Engine for Landsat-8 and Copernicus Satellite for satellite imagery, NDVI calculations through javascript and python, Numpy for mathematical calculations of Latitude and Longitudes, Leaflet maps for the user interface and toolbar for drawings using javascript. Overall we have used Python, Javascript, Html and Css with Flask.

**Commands for Installation of Packages:**

I’ve used Anaconda 3.7 with all built-in packages and Pycharm for this project. You need to install python to your pc and then install these packages in Flask application.

* Download Anaconda

<https://www.anaconda.com/distribution/>

* Download PyCharm

<https://www.jetbrains.com/pycharm/>

* Download Pip

<https://pip.pypa.io/en/stable/installing/>

* pip install numpy
* pip install flask
* pip install opencv-python
* pip install ee
* pip install DateTime

1. First of all create account on <https://earthengine.google.com/> earth engine, check your mail for account verification, it may take a few time.
2. Run the below command from a command-line to download/install the Python API client

pip install google-api-python-client

1. Run the below command from a command-line to ensure you have the proper crypto libraries installed

python -c "from oauth2client import crypt"

1. If running this command results in an error message, you will need to download and install the proper crypto libraries. This can be accomplished by running the below command.

pip install pyCrypto

* if running that fails, download VS community ed, download the build tools and install
* Windows 10 SDK, Visual C++ Tools for CMake, Core Tools for Testing
* Then run this https://github.com/pycrypto/pycrypto/issues/218#issuecomment-434551388

1. Run the below command from a command-line to download/install the Earth Engine Python library

pip install earthengine-api

1. Run the below command from a command-line to initialize the API and verify your account.

python -c "import ee; ee.Initialize()"

1. This will result in an error message due to the fact that Google still needs to verify your account with Earth Engine and it currently does not have the proper credentials. Therefore, run:

earthengine authenticate

1. This will open your default web-browser (ensure that you’re currently logged into your Google account) and provide you with a unique key to verify your account. Copy and paste the key into the terminal when prompted for the key.
2. Run python so that you’re utilizing the Python Command Line Interface (CLI) and run the following commands to ensure that the Earth Engine Python API is properly installed.

python

>>> import ee

>>> ee.Initialize()

1. If it is still showing errors, try to add these new modules and follow the same process again.

pip3.6 install --user pwhich (keep your version instead of 3.6)

1. You can specify which Python version to use for your virtualenv using the --python option. So, to create a new Python 3.6 virtualenv, run this command:

$ mkvirtualenv my-virtualenv --python=python3.6 (keep your version instead of 3.6)

1. Once you're in a virtualenv, to install packages you can just use pip with no Python version number or --user flag:

(my-virtualenv) $ pip install pwhich

1. We recommend that in any Python 2.7 virtualenv you create, you install the following security fix packages:

(my-virtualenv) $ pip install urllib3[secure] pyopenssl ndg-httpsclient pyasn1

There are 04 files in total:

* Index.html
* Results.html
* Test.html
* App.py

**Home Page (index.html)**

It is the first page of application it has navigation bar, manual to how it works, about smart city lab and contact portion for the user. On clicking ‘Get Started’ it will take you to next page of maps.

**Map Page (test.html + script)**

In this page we are using leaflet maps using javascript and html css to design it, it takes coordinates with the help of drawing. For drawings we are using toolbar which extract the coordinates and forward those coordinaes to python code. Submit button will not enable until you’ll select coordinates with the help of rectangle and python. We keep map open for all around the world but application will show the position of Karachi in start, you can go anywhere though.

**Result Page (results.html)**

This page is for the display of results using textareas and image file. It will display the Total Area (sq. km), Green Area (sq. km), Non-green Area (sq. km) and Percentage. You have to click Process button for the processing. Gif will appear till the code process and output image will be updated in last with values. Code() function is being called from app.py and f1() function is for GIF.

**Python Code (app.py)**

It is the main file of application where we are using ee, numpy, flask and opencv. Function code() is supposed to process the satellite imagery using ee. We are using bands on the basis of ndvi, RGB for color filtration and masking for last green colors on the map images. NDVI is being calculated with the help of Red and Near Infrared bands. Output image will be saved in ‘Static’ folder and displayed on last page of the application. Calculations are purely mathematical based and for python we have used numpy for the calculations.

Keep index.html, test.html and result.html in templates folder and run app.py as main file. Also give path of output image to static folder.

**Calculation of NDVI:**

Vegetation reflects light in the near-infrared (NIR) part of the electromagnetic spectrum and absorbs light in the red part. NDVI uses this to create a single value roughly reflecting the photosynthetic activity occurring at a pixel. The calculation is

(NIR - red) / (NIR + red)

This results in a number between 1 and -1, where pixels with high photosynthetic activity have a high NDVI. Earth Engine uses this way to calculate Net Vegetation Index from Satellite Imagery.

**Satellite Imagery:**

For Satellite imagery we are using Sentinel-2 because it uses 10m resolution but we can also use Landsat-8 with 30m resolution or any other satellite using Earth Engine. If you change the satellite imagery make sure you also add the bands accordingly and before extracting data using filterdate check if that satellite provides data or not.

**Data Collection:**

For data collection we have used .filterDate function from Earth Engine with Start and Ending date. We have collected monthly data of Karachi for past 7 years. We have collected total area, green area, nongreen area and percentage, but we can also collect output images in future as well.

**Piece of code for Monthly data**

**Add this piece of code to the code() function in order to collect data**

from datetime import date

from dateutil.rrule import rrule,MONTHLY

import datetime

dt = datetime.date(2013, 4, 1) #add the starting date according to your satellite, year, month, day

q = dt.strftime("%Y-%m-%d")

a = 1

while a<91: #set while condition for number of months

df = dt + datetime.timedelta(weeks = a\*4) #set number of days or weeks you want to extract

w = df.strftime("%Y-%m-%d")

print(w)

a = a + 1

#Define the area

area = ee.Geometry.Polygon(polygon)

#define the image

coll = ee.ImageCollection("YOUR SATELLITE").filterDate("" +q+ "","" +w+ "" )

**Piece of code for CSV files:**

**Add this piece of code to the last of code() function in order to collect data**

import csv

with open('Area.csv','a', newline="") as f:

thewriter = csv.writer(f)

thewriter.writerow([‘Date’,’Total Area’,’Green Area’,’Nongreen Area’,’Percentage’, ‘Coordinates’])

thewriter.writerow([w,total,green,nongreen,percent, polygon])

#add variables you want to add in csv in last line

**Note:**

Before starting this project make sure you have Google Earth Engine permission for EE editor and authentication for using api with Python. Apply for api and it may take time for permission approval.

**Code:**

Each part of code has comments in it so it will be easier to understand. Project is also deployed on Python Anywhere to use.

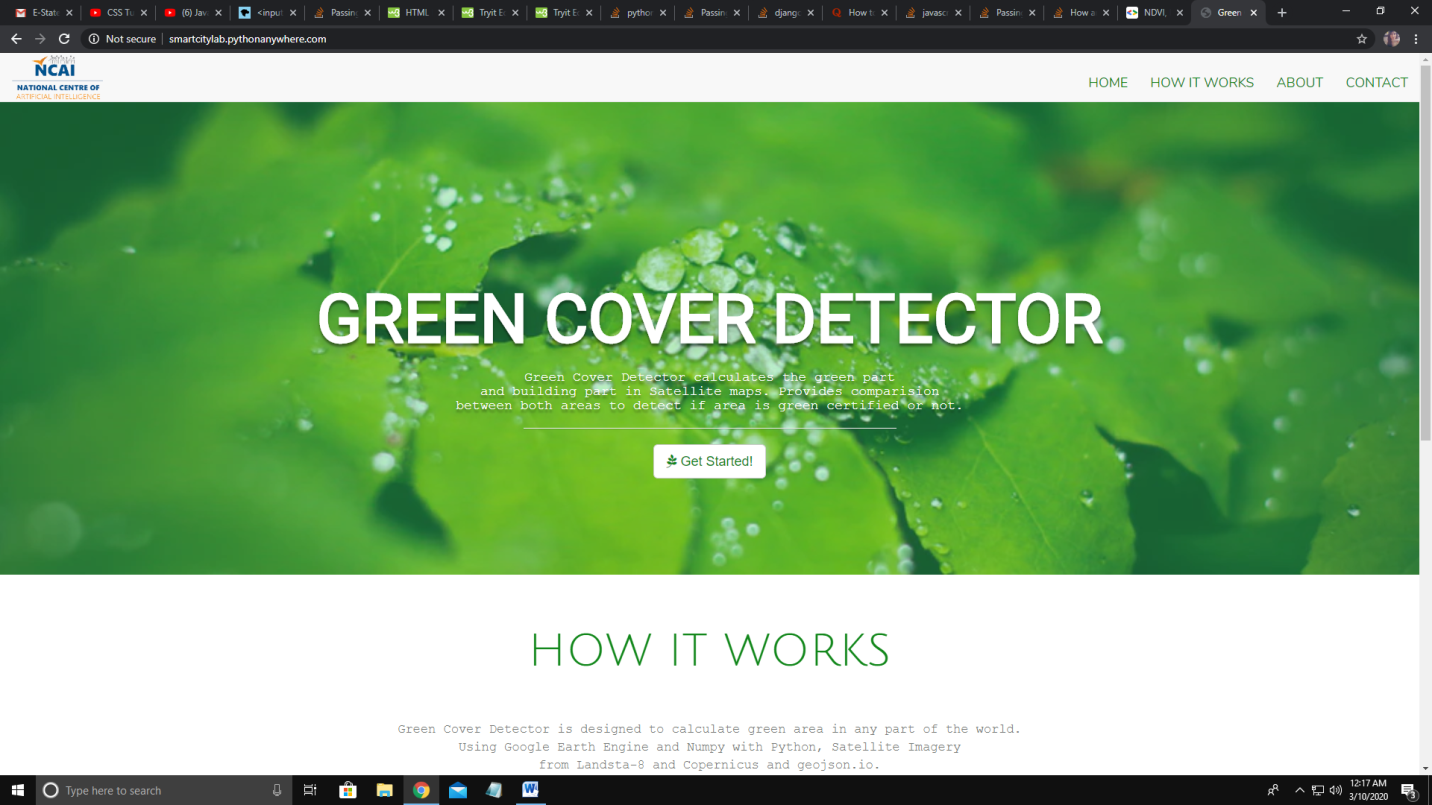
<http://smartcitylab.pythonanywhere.com/>

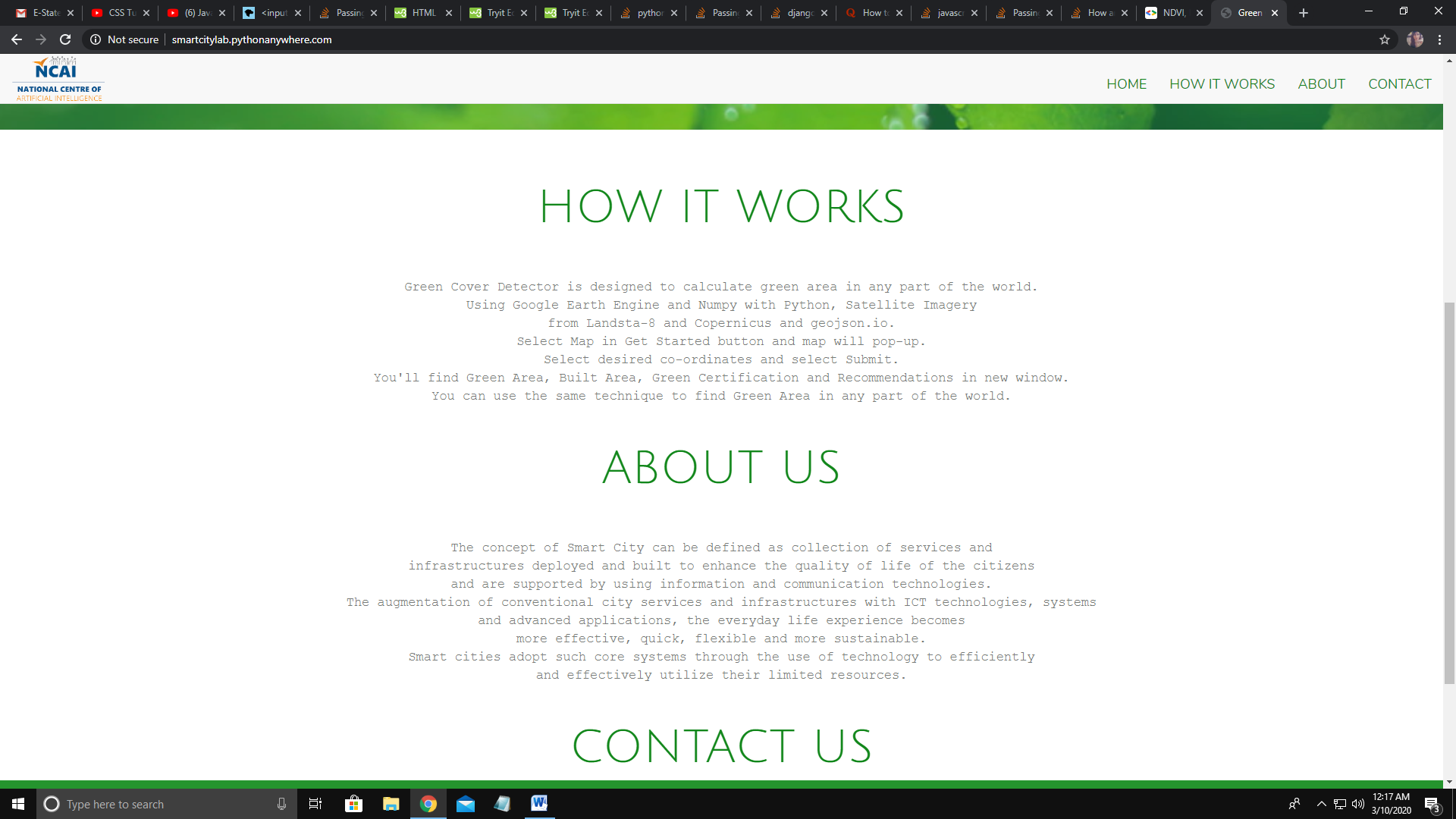
**Future Work:**

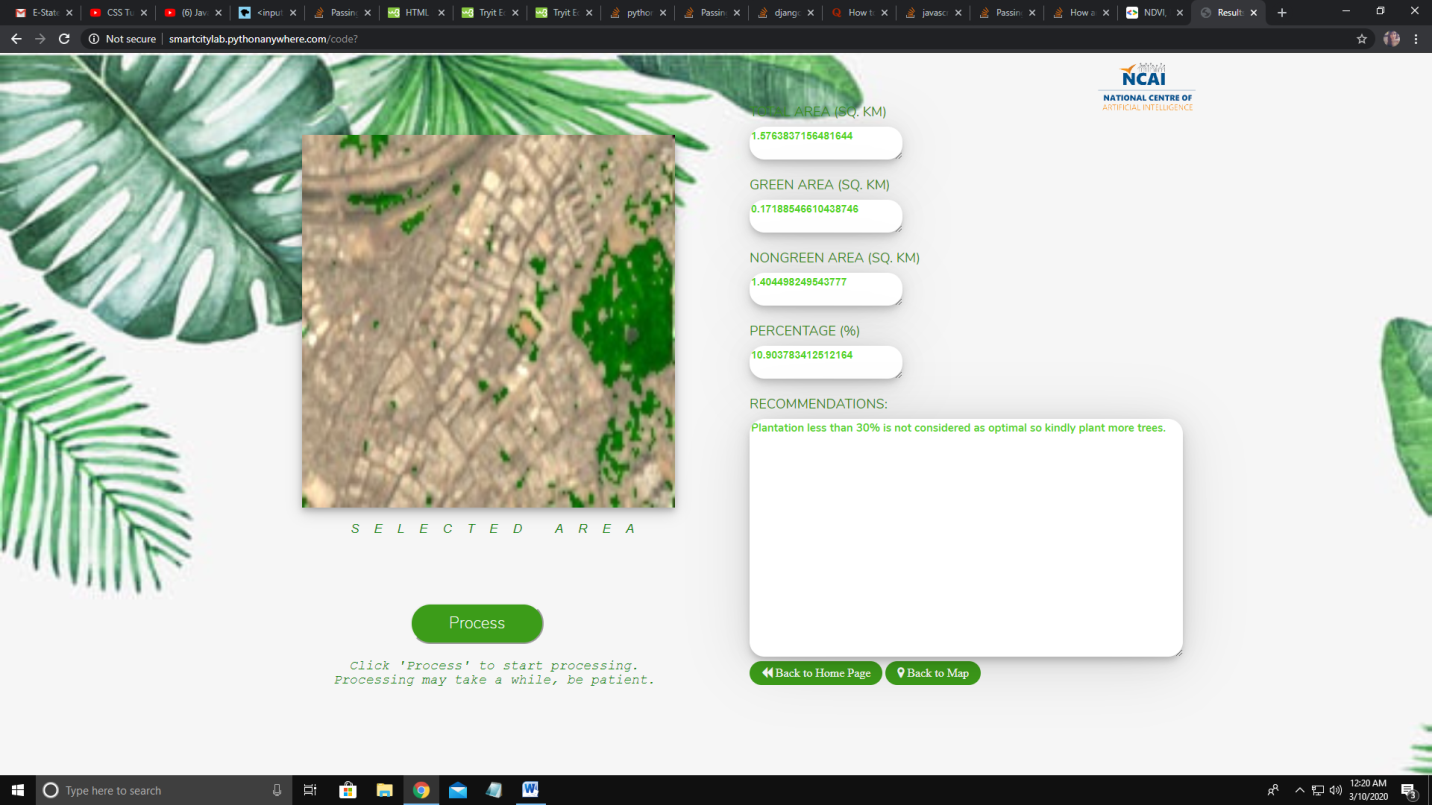
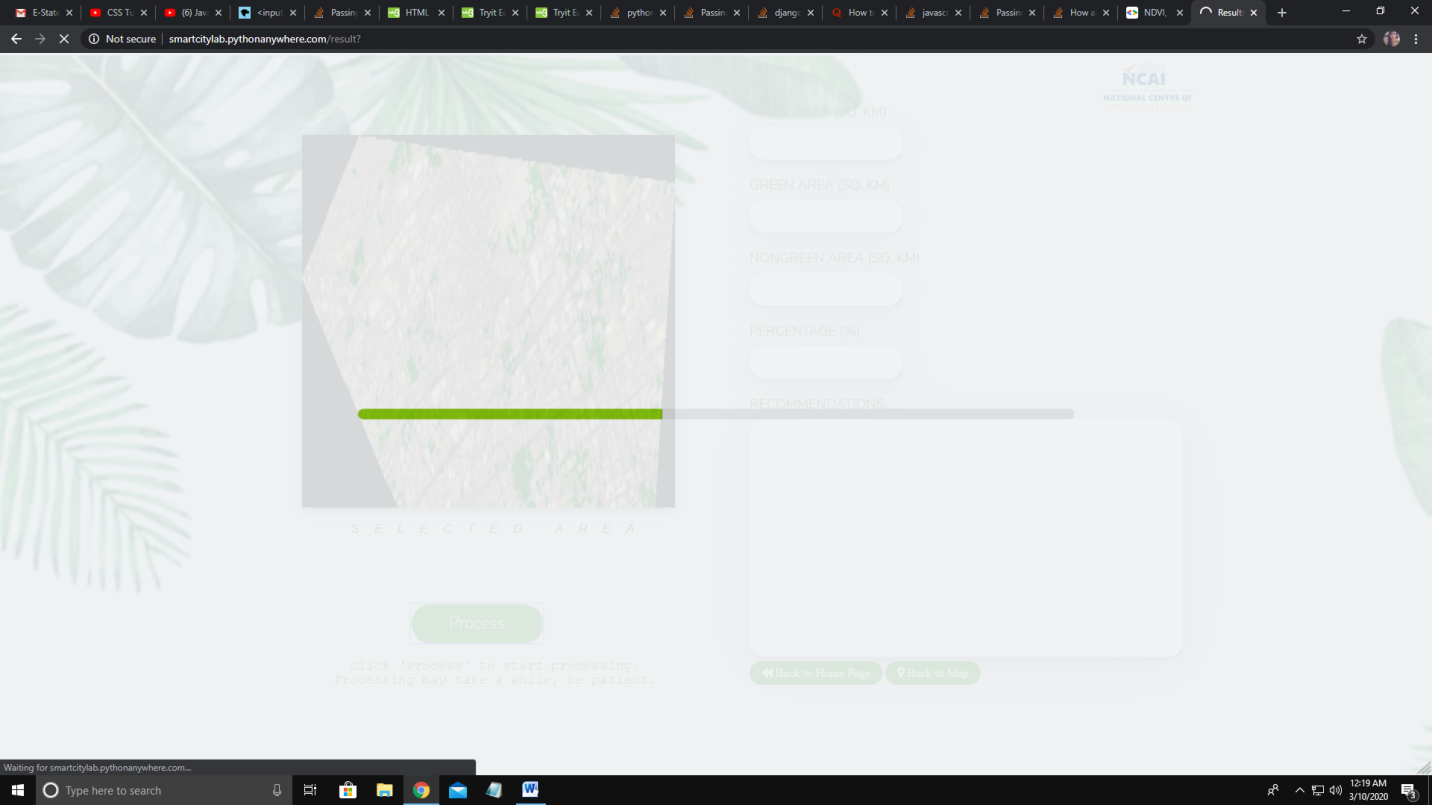
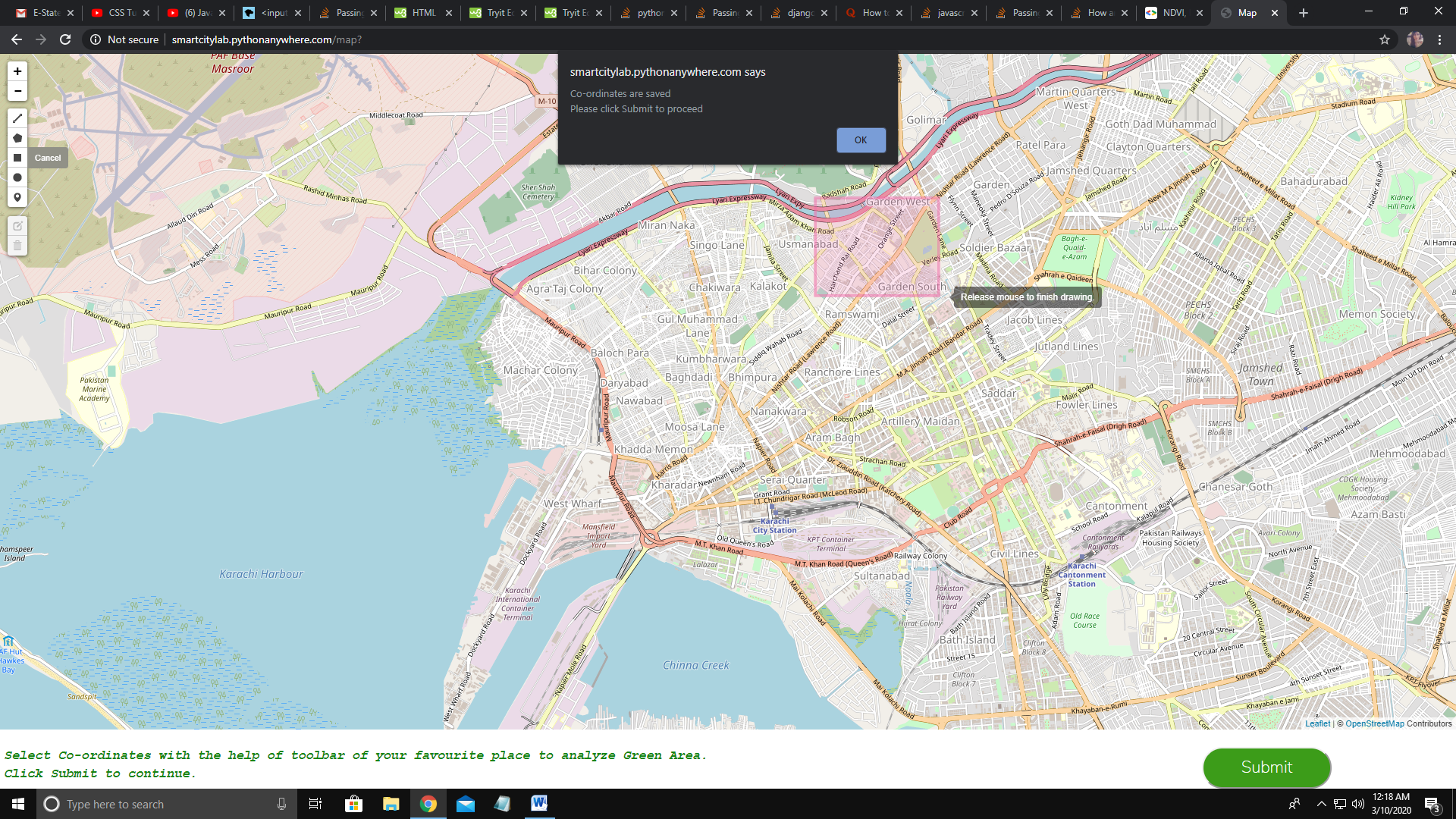
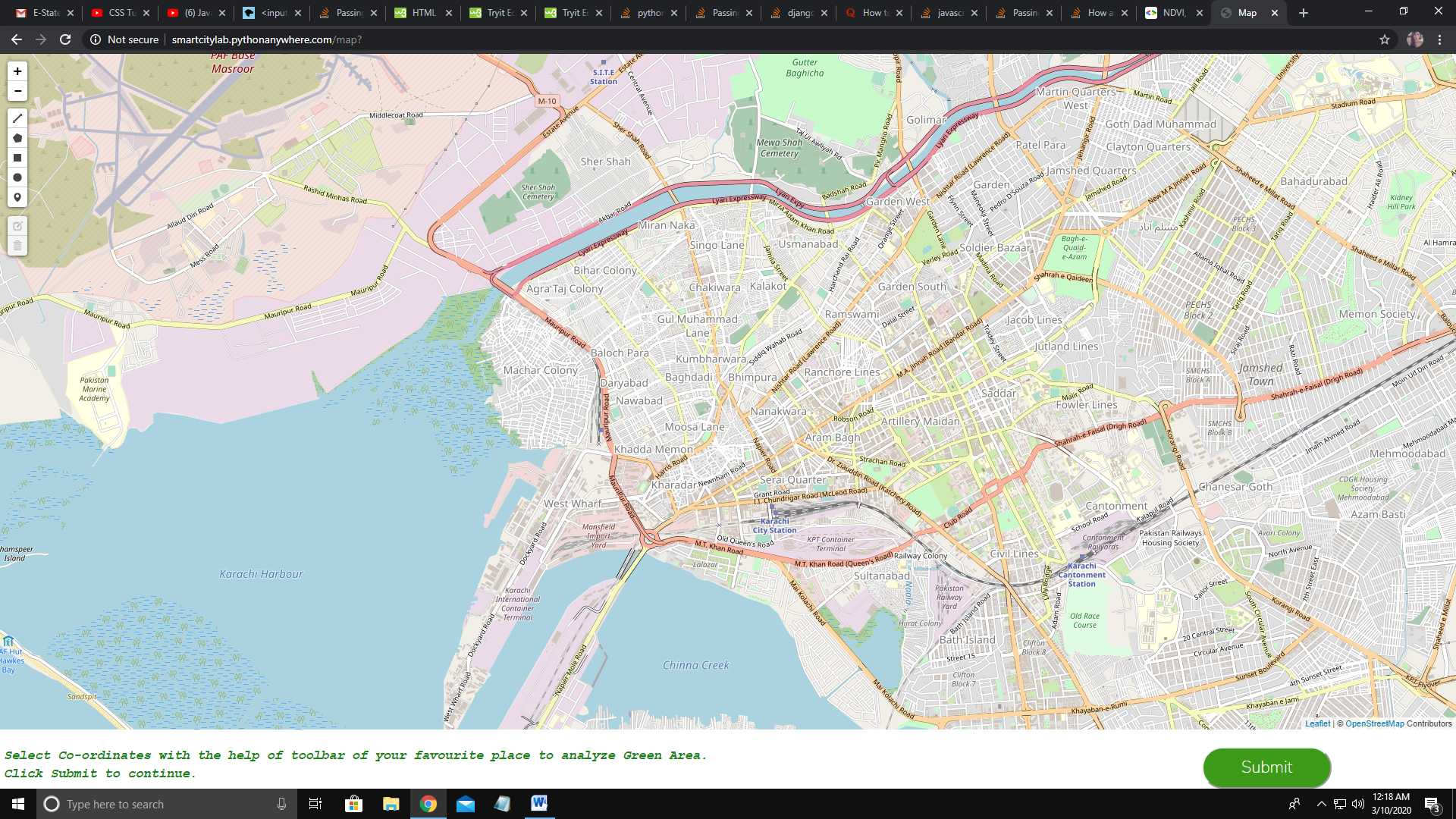
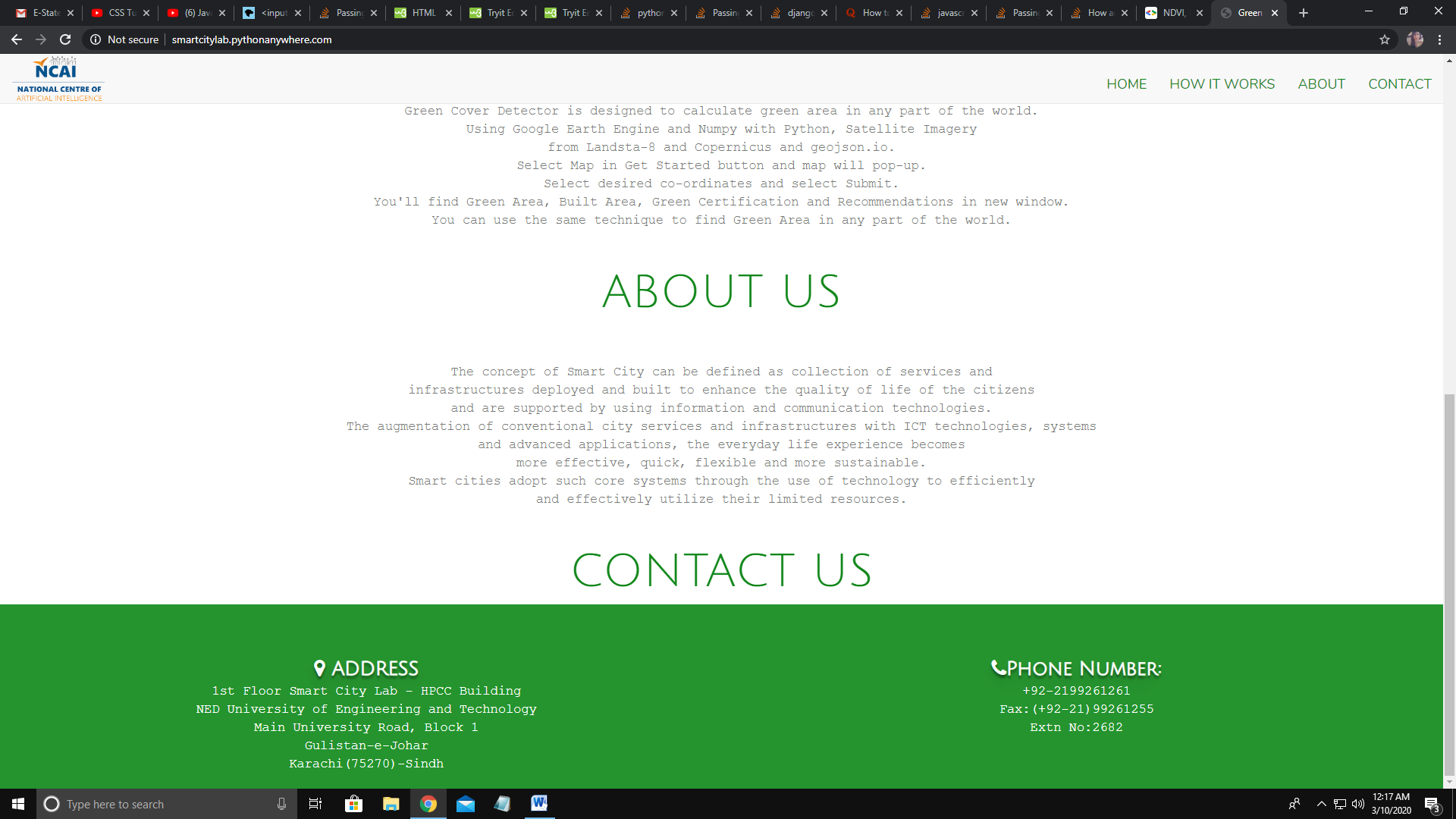
* Automation of Data Collection process.
* Addition of Satellite map instead of Leaflet map or option for both views.
* Addition of loop for image outputs with other data.
* Masking of Green Area on Output Image with clear pixels.
* Time lapse video of change in greenery.
* Better responsiveness of web application.

**Results:**

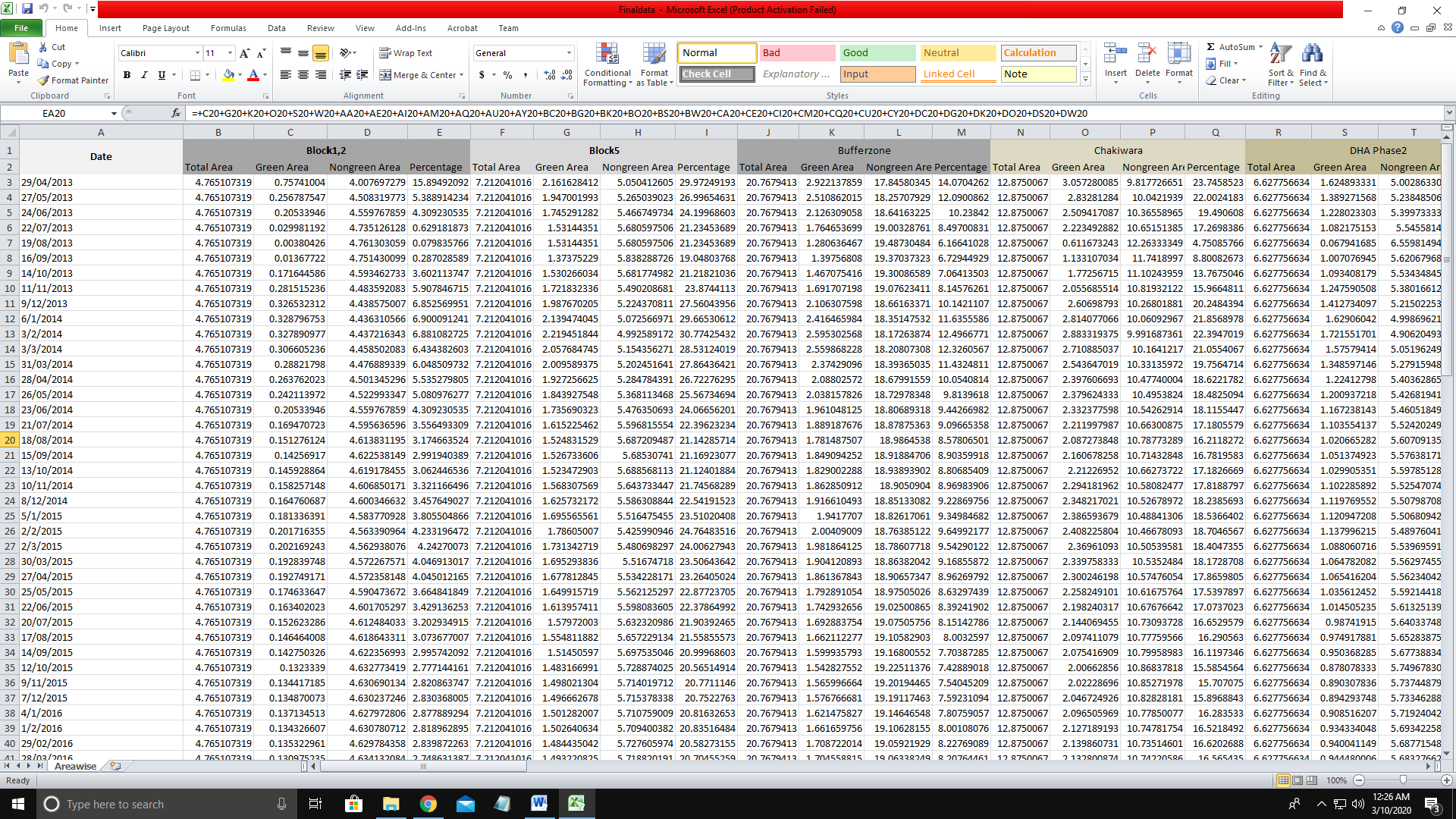
It takes coordinates as user input with the help of drawings, process it and show the total area, green area, non-green or build area and percentage of green area. Output image with green color masking of detected green area.

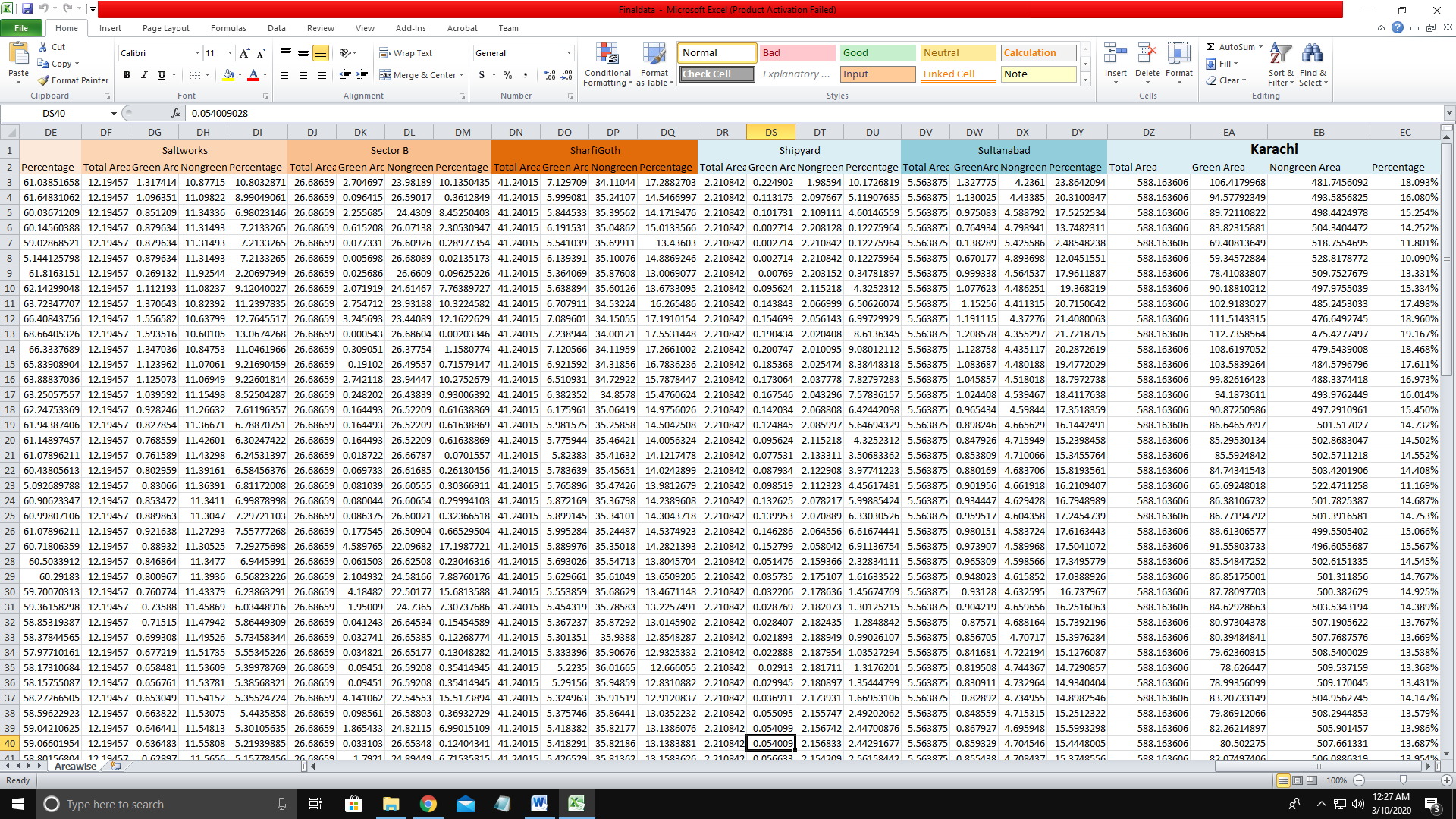
**Screenshots of web app: **

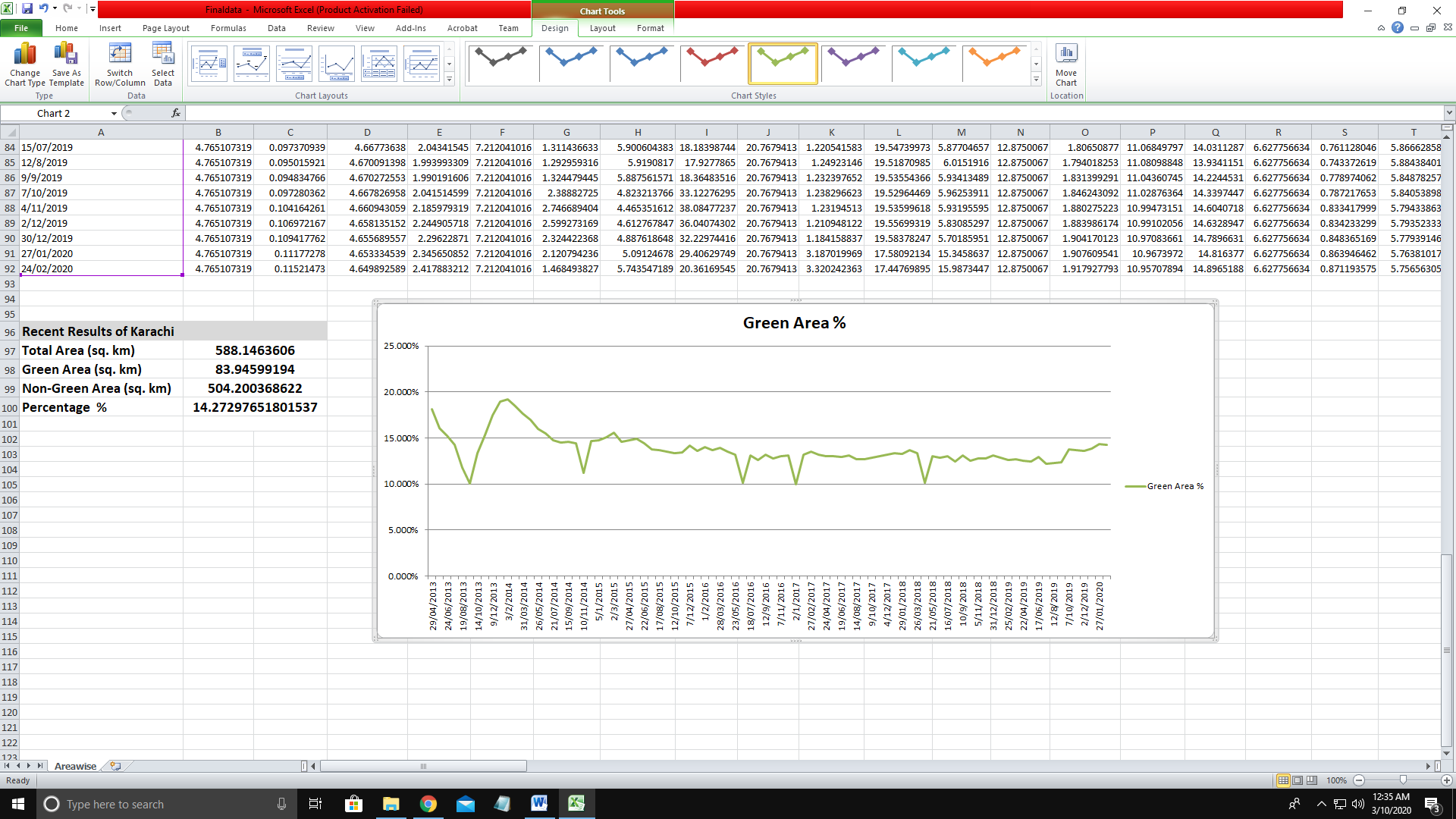
****

****

**Data Collection of Karachi:**

****

****

****

****

**Covered Areas of Karachi**

**Some Other Result Images:**

****

****

**Conclusion:**

It is detecting green area in satellite maps and showing results accordingly, in short it is a great initiative towards awareness for the plantation of trees in city like Karachi where we lack number of trees and it is causing health problems. We can beat air pollution with more plantations.