**Project Title**

ANALYSIS OF BRIDGE CRACKS AND REPORT

Introduction:

* The aim of the project is to check the regular maintenance of the roads and bridges and providing an interface which enhances the experience , so more précised focus can be given .
* The project focuses on the idea of keeping the past data ,checking the roads for cracks

.

* The project is mainly classified into 3 parts .Data cleaning,image processing and sending alert message to users.
* Within this project it might be easier to look after the roads since automation process tends to make it easier
* This will also help to alert people who commute through a route for a particular mishap on the road such as cracks ,heavy rain etc.

**Requirement Specification:**

**Software Requirements:**

* Windows 7 or Higher.
* Python.
* Data wrapper
* Twilio

**Hardware Requirements:**

* Processor: i5.
* Hard Disk: 50Gb.
* RAM: 8Gb.
* Camera(with a better vision)

System Design Details:

Methodology:

* The first objective of the project is to clean data making sure that no data is missing .
* Data collection:- For a new road all the data collected will be true (assuming data is collected from valid sources)information such as season and density of vehicles will be done with automation.it is important to check that whatever data is collected is from valid sources
* Cheching the images for cracks which contains the image processing can be divided in 3 parts

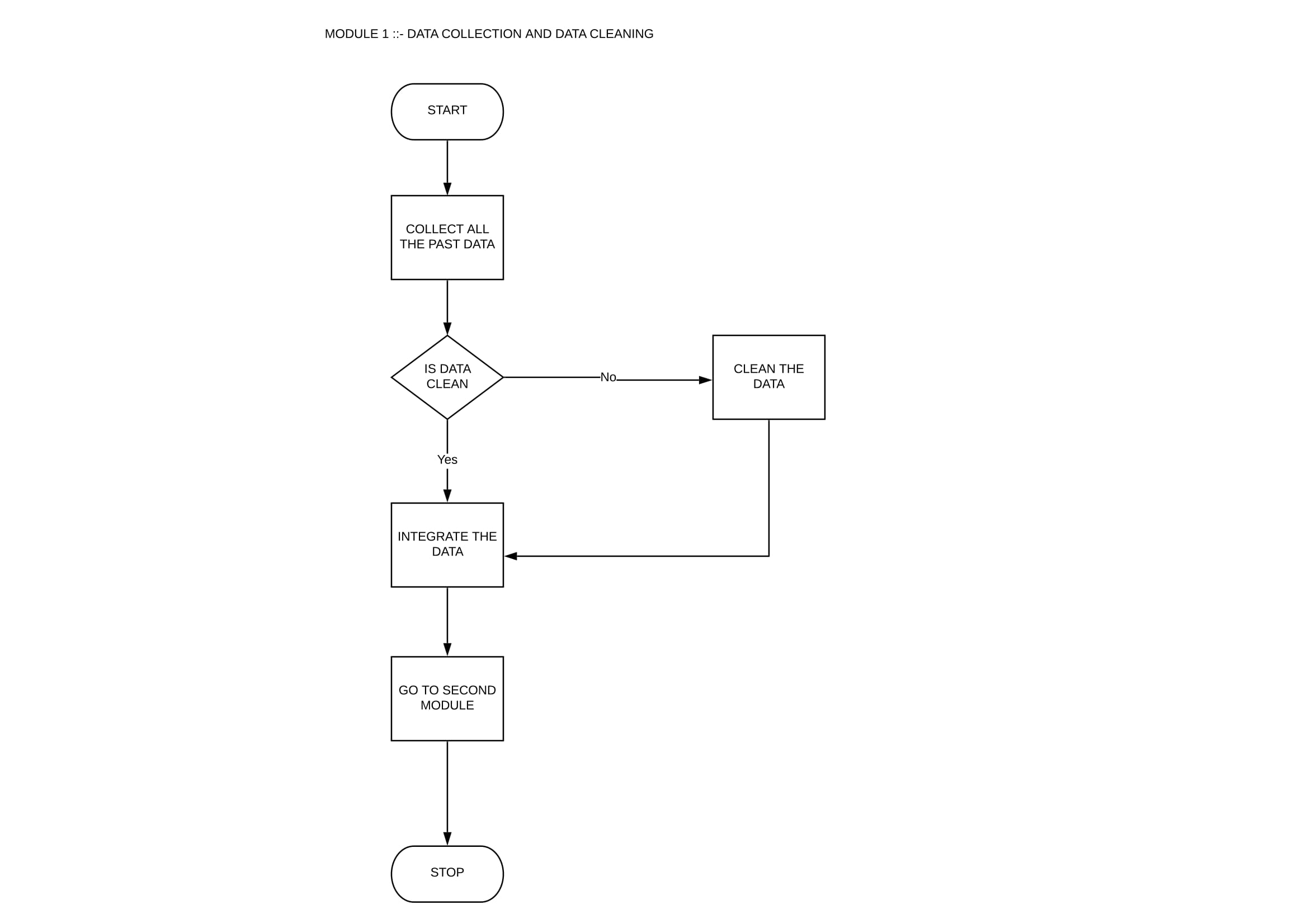
1. Capturing the image
2. Removing noise,blur etc
3. Converting it to grey scale
4. Applying Canny edge algorithm

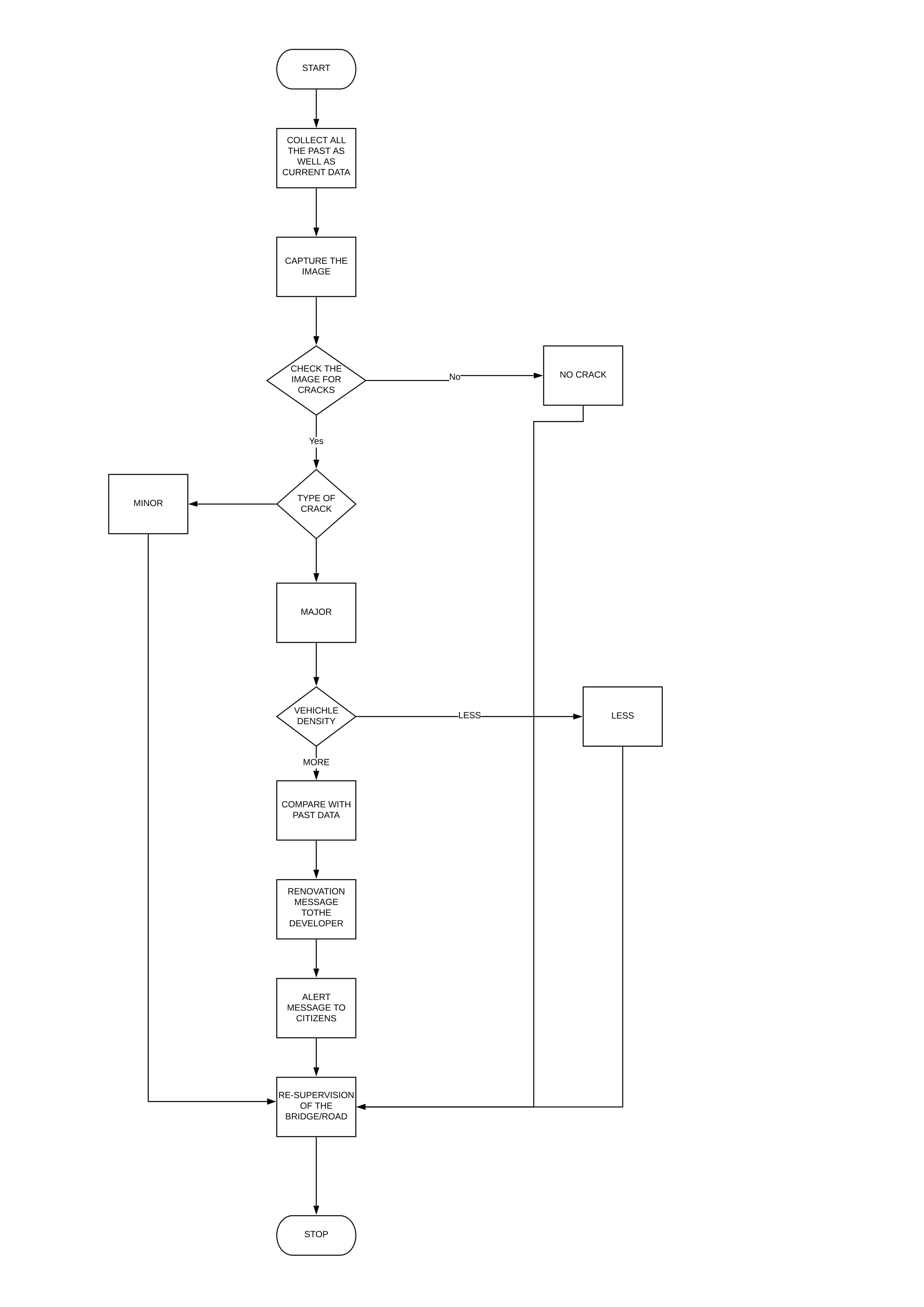
* If there exist a crack which may lead to a particular mishap it is important to alert the commuters of that route so sending text message to the users
* The overview of the road is done with the data visualization tool which will give information about the road such as

1. Traffic signals
2. Season overview
3. Traffic points
4. Cracks
5. Any hospitals or school nearby

Data Flow Diagram:

* A Data Flow Diagram is a graphical representation of the “flow” of data there an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system. Which can later elaborate? DFD‟s can also be used for visualization of data processing (Structured design)



****

**USE CASE DIAGRAM**

* In the software and the system engineering, a use case is a list of steps, typically defining interaction between a role (known in unified Modeling Language (UML) as an “actor‟)and a system to achieve a goal. The actor can be a human an external system as time. In system engineering, use cases are used at a higher level than within software engineering often representing mission or stakeholder goals.
* The detailed requirements may than be captured in system modeling language (sys UML) or as contractual statements. As an important requirement use cases have been widely used in modern software engineering over the last two decades. Use case driven development is a key characteristics of process models and frameworks like Unified Process (UP). Rational Unified Process (RUP). Oracle Unified Method (OUM) etc. With its iterative and evolutionary nature. Use case is also a good fit for agile development.

DATA COLLECTION AMA

Data cleaning

ALERTING USERS

System Implementation:

* Connecting remote camera

import urllib

import urllib.request

import cv2

import numpy as np

import time

# Replace the URL with your own IPwebcam shot.jpg IP:port

url='https://192.168.0.104:8888'

while True:

# Use urllib to get the image and convert into a cv2 usable format

imgResp=urllib.request.urlopen(url)

imgNp=np.array(bytearray(imgResp.read()),dtype=np.uint8)

img=cv2.imdecode(imgNp,-1)

# put the image on screen

cv2.imshow('IPWebcam',img)

#To give the processor some less stress

#time.sleep(0.1)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

* **DATA VIEWING**

import tkinter

from tkinter import \*

from tkinter import messagebox

top = tkinter.Tk()

import pandas as pd

df = pd.read\_excel (r'C:\Users\admin1\Desktop\final\_project\DATA\_COLLECTION(PAST).xlsx')

data = pd.DataFrame(df, columns= ['NAME OF THE BRIDGE'])#printing the only requirement

place=pd.DataFrame(df, columns= ['PLACE'])

CON=pd.DataFrame(df, columns= ['CONTRACTOR'])

DAT=pd.DataFrame(df, columns= ['CONSTRUCTION DATE'])

DENSITY=pd.DataFrame(df, columns= ['VEHICHLE DENSITY(PR/HR)'])

LAST=pd.DataFrame(df, columns= ['LAST RENOVATED'])

COMM=pd.DataFrame(df, columns= ['COMMENTS ON LAST RENOVATION'])

BUD=pd.DataFrame(df, columns= ['BUDGET OF LAST RENOVATION'])

def NAMES():

print(data)

def PLACE():

print(place)

def CONTRACTOR():

print(CON)

def DATE():

print(DAT)

def DENSITY():

print(DENSITY)

def LAST\_RENOVATED():

print(LAST)

def COMMENTS():

print(COMM)

def BUDGET():

print(BUD)

B = tkinter.Button(top, text ="NAME OF THE BRIDGE", command = NAMES,cursor="circle")

B1 = tkinter.Button(top, text ="PLACE", command = PLACE,cursor="plus")

B2 = tkinter.Button(top, text ="CONTRACTOR", command = CONTRACTOR,cursor="circle")

B3 = tkinter.Button(top, text ="DATE", command =DATE,cursor="plus")

B4= tkinter.Button(top, text ="DENSITY", command = DENSITY,cursor="circle")

B5 = tkinter.Button(top, text ="LAST\_RENOVATED", command = LAST\_RENOVATED,cursor="plus")

B6 = tkinter.Button(top, text ="COMMENTS", command = COMMENTS,cursor="circle")

B7 = tkinter.Button(top, text ="BUDGET", command = BUDGET,cursor="plus")

B.grid(row=1, column=0)

B1.grid(row=3, column=0)

B2.grid(row=5, column=0)

B3.grid(row=7, column=0)

B4.grid(row=9, column=0)

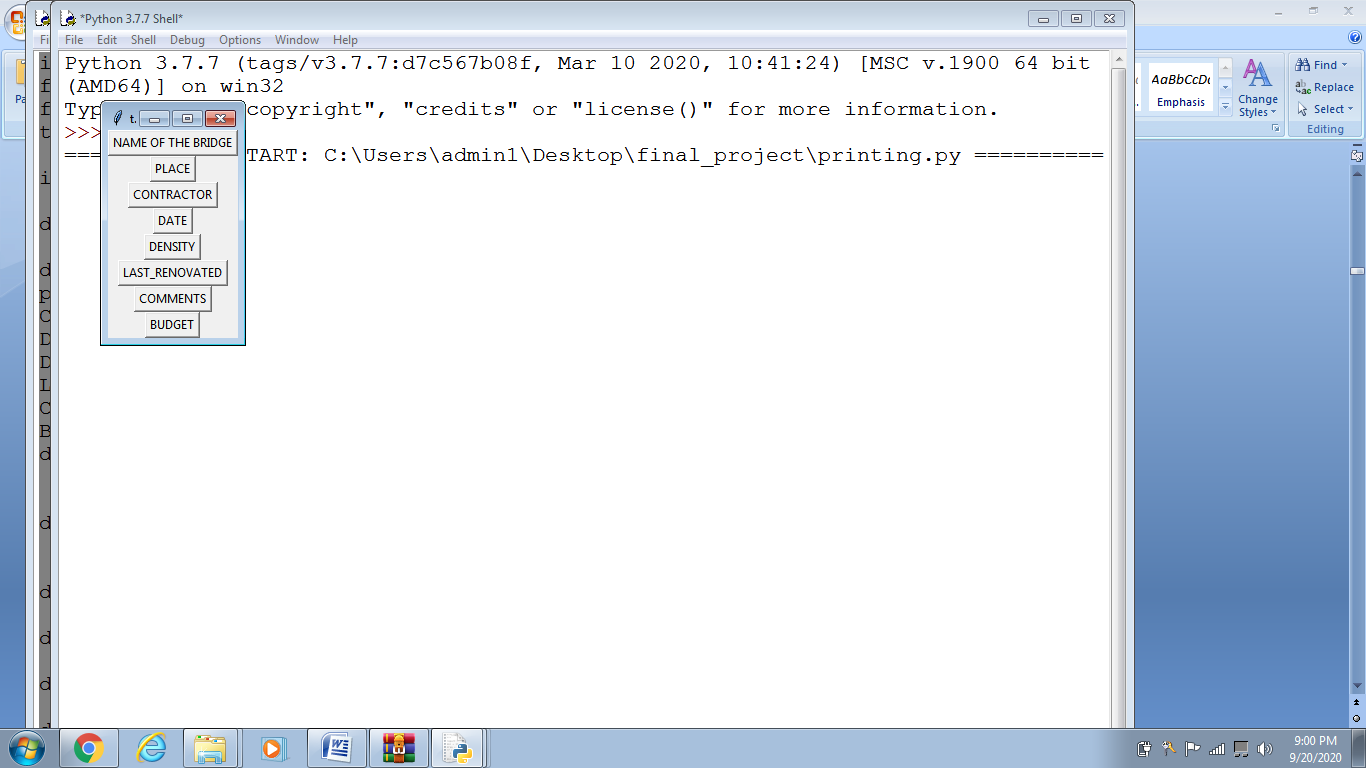
B5.grid(row=11, column=0)

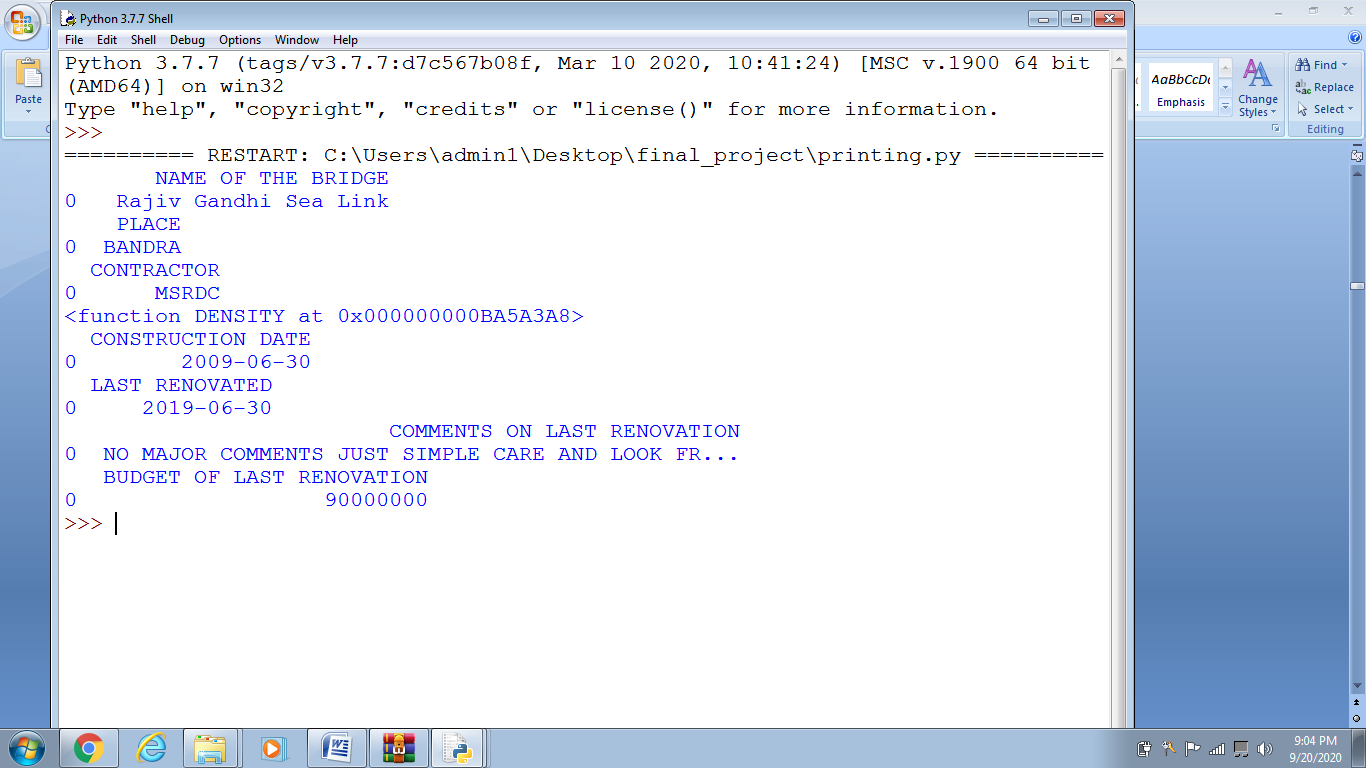
B6.grid(row=13, column=0)

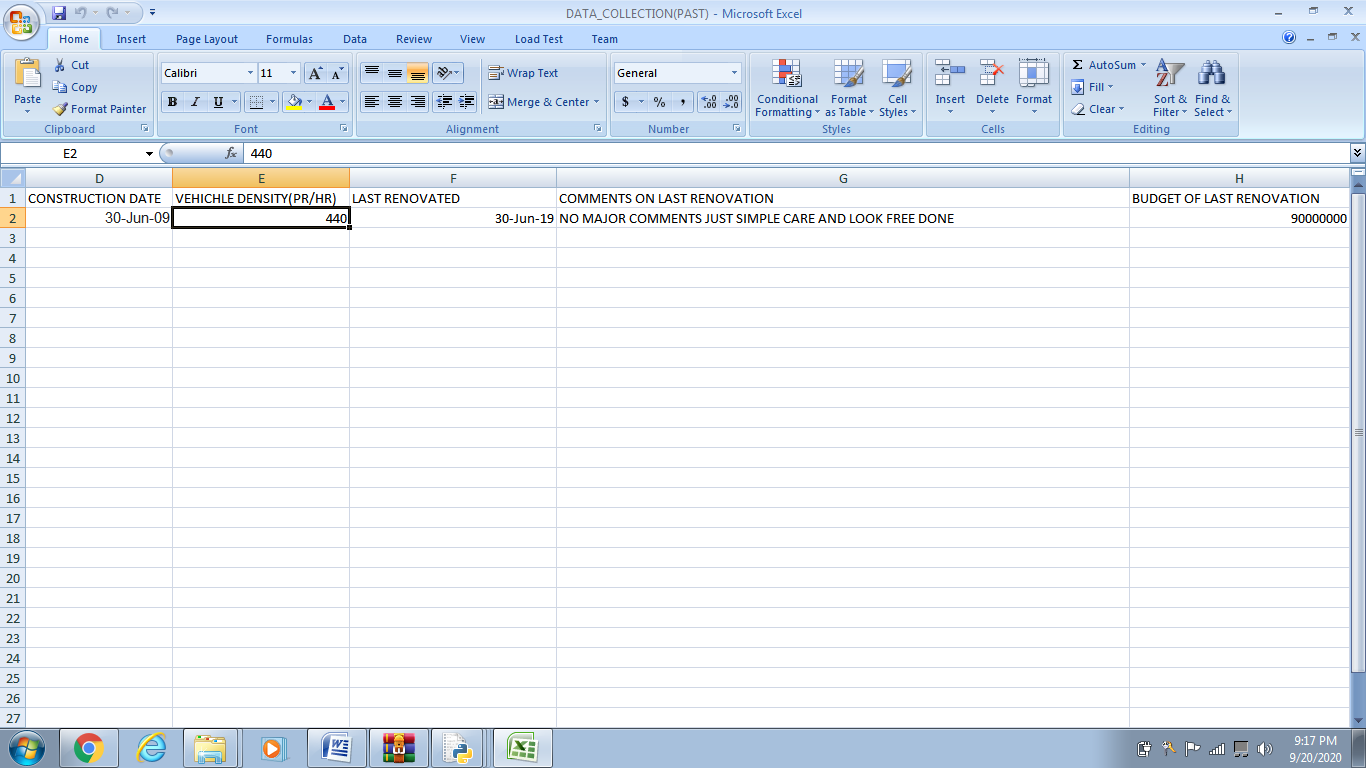
B7.grid(row=15, column=0)

# Code to add widgets will go here...

top.mainloop()





****

* **Data collection:**

# import openpyxl and tkinter modules

from openpyxl import \*

from tkinter import \*

import datetime

import cv2

import matplotlib.pyplot as plt

import cvlib as cv

from cvlib.object\_detection import draw\_bbox

# globally declare wb and sheet variable

# opening the existing excel file

wb = load\_workbook('BRIDGE\_INFO.xlsx')

# create the sheet object

sheet = wb.active

def excel():

# resize the width of columns in

# excel spreadsheet

sheet.column\_dimensions['A'].width = 30

sheet.column\_dimensions['B'].width = 10

sheet.column\_dimensions['C'].width = 10

sheet.column\_dimensions['D'].width = 20

sheet.column\_dimensions['E'].width = 20

sheet.column\_dimensions['F'].width = 40

sheet.column\_dimensions['G'].width = 50

# write given data to an excel spreadsheet

# at particular location

sheet.cell(row=1, column=1).value = "Name of BRIDGE"

sheet.cell(row=1, column=2).value = "PLACE"

sheet.cell(row=1, column=3).value = "CONTRACTOR"

sheet.cell(row=1, column=4).value = "BUILT ON "

sheet.cell(row=1, column=5).value = "LAST RENOVATED"

sheet.cell(row=1, column=6).value = "COMMENTS ON LAST RENOVATION"

sheet.cell(row=1, column=7).value = "BUDGET OF LAST RENOVATION"

sheet.cell(row=1, column=8).value = "CURRENT SEASON"

sheet.cell(row=1, column=9).value = "DENSITY OF VEHICHLES"

def season():

SEASON = datetime.datetime.now()

s=int(SEASON.strftime("%m"))

current\_row = sheet.max\_row

current\_column = sheet.max\_column

if s > 5 and s < 11:

sheet.cell(row=current\_row + 1, column=8).value = "RAINY"

elif s > 10 and s < 2:

sheet.cell(row=current\_row + 1, column=8).value = "WINTER"

else:

sheet.cell(row=current\_row + 1, column=8).value = "SUMMER"

def density():

current\_row = sheet.max\_row

current\_column = sheet.max\_column

im = cv2.imread('car.png')

bbox, label, conf = cv.detect\_common\_objects(im)

#output\_image = draw\_bbox(im, bbox, label, conf)

#plt.imshow(output\_image)

#plt.show()

v=str(label.count('car'))

#print(v)

print('Number of cars in the image is '+ str(label.count('car')))

a=v

sheet.cell(row=current\_row + 1, column=9).value = a

# Function to set focus (cursor)

def focus1(event):

# set focus on the course\_field box

course\_field.focus\_set()

# Function to set focus

def focus2(event):

# set focus on the sem\_field box

sem\_field.focus\_set()

# Function to set focus

def focus3(event):

# set focus on the form\_no\_field box

form\_no\_field.focus\_set()

# Function to set focus

def focus4(event):

# set focus on the contact\_no\_field box

contact\_no\_field.focus\_set()

# Function to set focus

def focus5(event):

# set focus on the email\_id\_field box

email\_id\_field.focus\_set()

# Function to set focus

def focus6(event):

# set focus on the address\_field box

address\_field.focus\_set()

# Function for clearing the

# contents of text entry boxes

def clear():

# clear the content of text entry box

name\_field.delete(0, END)

course\_field.delete(0, END)

sem\_field.delete(0, END)

form\_no\_field.delete(0, END)

contact\_no\_field.delete(0, END)

email\_id\_field.delete(0, END)

address\_field.delete(0, END)

# Function to take data from GUI

# window and write to an excel file

def insert():

# if user not fill any entry

# then print "empty input"

if (name\_field.get() == "" and

course\_field.get() == "" and

sem\_field.get() == "" and

form\_no\_field.get() == "" and

contact\_no\_field.get() == "" and

email\_id\_field.get() == "" and

address\_field.get() == ""):

print("empty input")

else:

# assigning the max row and max column

# value upto which data is written

# in an excel sheet to the variable

current\_row = sheet.max\_row

current\_column = sheet.max\_column

# get method returns current text

# as string which we write into

# excel spreadsheet at particular location

sheet.cell(row=current\_row + 1, column=1).value = name\_field.get()

sheet.cell(row=current\_row + 1, column=2).value = course\_field.get()

sheet.cell(row=current\_row + 1, column=3).value = sem\_field.get()

sheet.cell(row=current\_row + 1, column=4).value = form\_no\_field.get()

sheet.cell(row=current\_row + 1, column=5).value = contact\_no\_field.get()

sheet.cell(row=current\_row + 1, column=6).value = email\_id\_field.get()

sheet.cell(row=current\_row + 1, column=7).value = address\_field.get()

# save the file

wb.save('BRIDGE\_INFO.xlsx')

# set focus on the name\_field box

name\_field.focus\_set()

# call the clear() function

clear()

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

# create a GUI window

root = Tk()

# set the background colour of GUI window

root.configure(background='light green')

# set the title of GUI window

root.title("registration form")

# set the configuration of GUI window

root.geometry("500x300")

excel()

# create a Form label

heading = Label(root, text="Form", bg="light green")

# create a Name label

name = Label(root, text="NAME OF BRIDGE", bg="light green")

# create a Course label

course = Label(root, text="PLACE", bg="light green")

# create a Semester label

sem = Label(root, text="CONTRACTOR", bg="light green")

# create a Form No. lable

form\_no = Label(root, text="BUILT ON.", bg="light green")

# create a Contact No. label

contact\_no = Label(root, text="LAST RENOVATED.", bg="light green")

# create a Email id label

email\_id = Label(root, text="COMMENTS ON LAST RENOVATED", bg="light green")

# create a address label

address = Label(root, text="BUDGET OF LAST RENOVATION", bg="light green")

season = Button(root, text="Season", fg="Black",

bg="Red", command=season)

density = Button(root, text="density", fg="Black",

bg="Red", command=density)

# grid method is used for placing

# the widgets at respective positions

# in table like structure .

heading.grid(row=0, column=1)

name.grid(row=1, column=0)

course.grid(row=2, column=0)

sem.grid(row=3, column=0)

form\_no.grid(row=4, column=0)

contact\_no.grid(row=5, column=0)

email\_id.grid(row=6, column=0)

address.grid(row=7, column=0)

season.grid(row=8,column=0)

density.grid(row=9,column=0)

# create a text entry box

# for typing the information

name\_field = Entry(root)

course\_field = Entry(root)

sem\_field = Entry(root)

form\_no\_field = Entry(root)

contact\_no\_field = Entry(root)

email\_id\_field = Entry(root)

address\_field = Entry(root)

# bind method of widget is used for

# the binding the function with the events

# whenever the enter key is pressed

# then call the focus1 function

name\_field.bind("<Return>", focus1)

# whenever the enter key is pressed

# then call the focus2 function

course\_field.bind("<Return>", focus2)

# whenever the enter key is pressed

# then call the focus3 function

sem\_field.bind("<Return>", focus3)

# whenever the enter key is pressed

# then call the focus4 function

form\_no\_field.bind("<Return>", focus4)

# whenever the enter key is pressed

# then call the focus5 function

contact\_no\_field.bind("<Return>", focus5)

# whenever the enter key is pressed

# then call the focus6 function

email\_id\_field.bind("<Return>", focus6)

# grid method is used for placing

# the widgets at respective positions

# in table like structure .

name\_field.grid(row=1, column=1, ipadx="100")

course\_field.grid(row=2, column=1, ipadx="100")

sem\_field.grid(row=3, column=1, ipadx="100")

form\_no\_field.grid(row=4, column=1, ipadx="100")

contact\_no\_field.grid(row=5, column=1, ipadx="100")

email\_id\_field.grid(row=6, column=1, ipadx="100")

address\_field.grid(row=7, column=1, ipadx="100")

# call excel function

excel()

# create a Submit Button and place into the root window

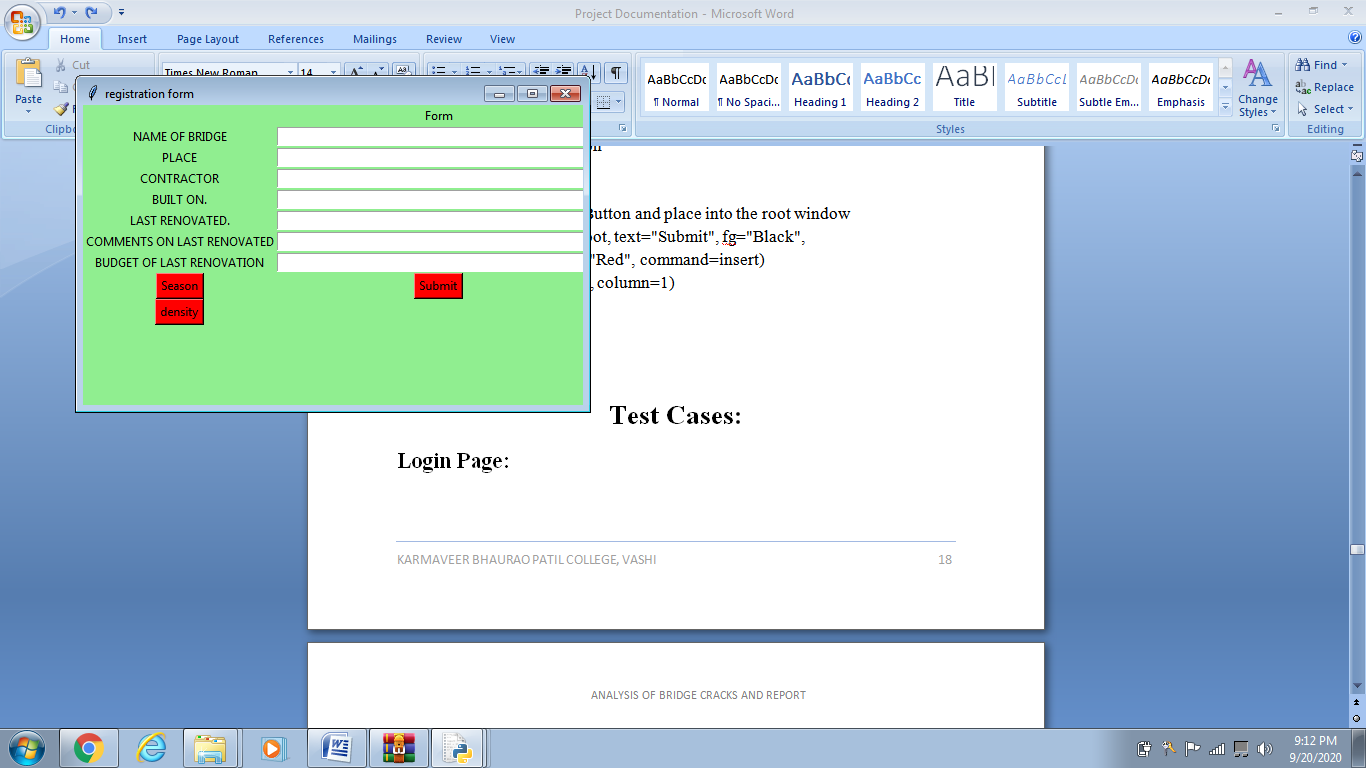
submit = Button(root, text="Submit", fg="Black",

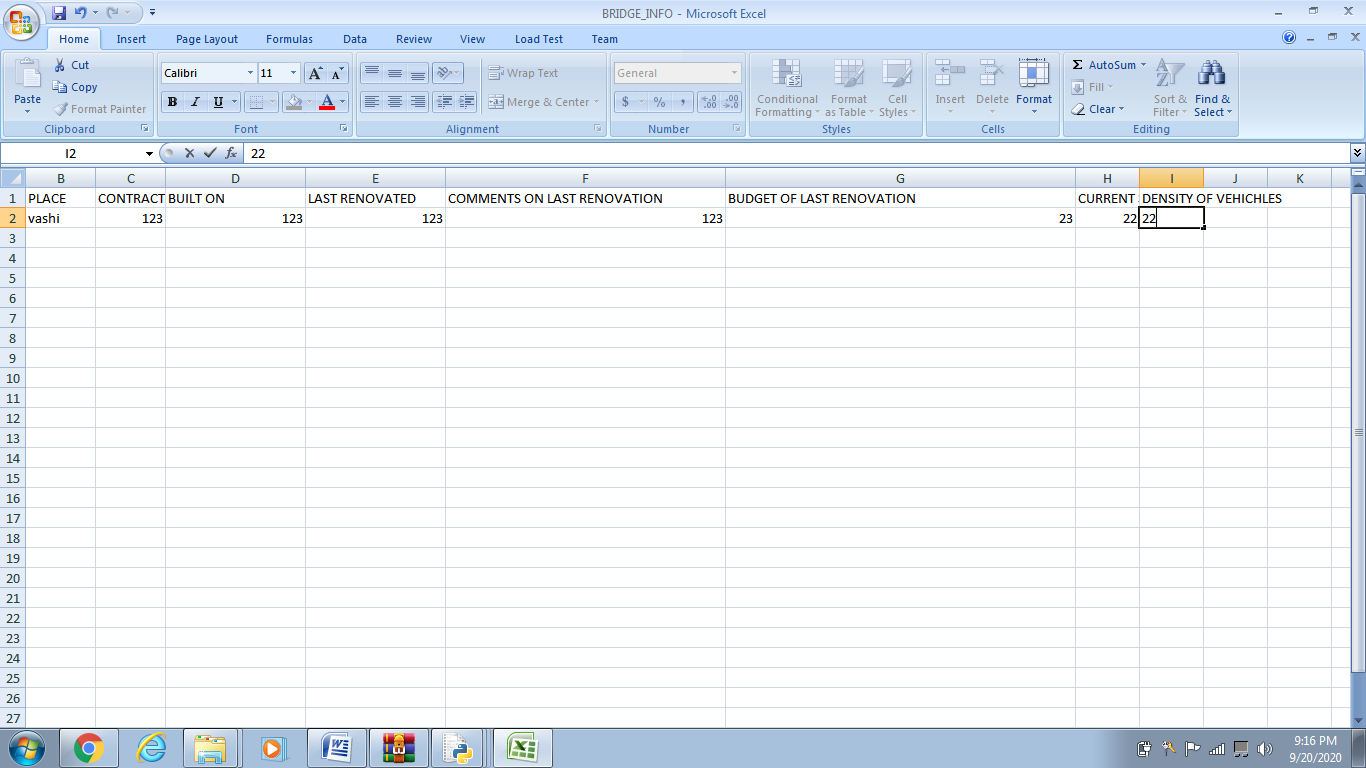
bg="Red", command=insert)

submit.grid(row=8, column=1)

# start the GUI

root.mainloop()



****

Sending Message and Viewing maps;-

from twilio.rest import Client

import tkinter

from tkinter import \*

from tkinter import messagebox

top = tkinter.Tk()

from PIL import Image

# creating a object

im = Image.open(r"road.png")

# Your Account Sid and Auth Token from twilio.com / console

account\_sid = 'ACe535f30a7351931d0c427514620b9e39'

auth\_token = '6895ae3ca95e054ecde5ce6253e884b0'

client = Client(account\_sid, auth\_token)

''' Change the value of 'from' with the number

received from Twilio and the value of 'to'

with the number in which you want to send message.'''

def send():

message = client.messages.create(

from\_='+12184232391',

body ='Heavy Rain Alert ',

to ='+919987611851'

)

def view():

im.show()

B = tkinter.Button(top, text ="SEND MESSAGE", command = send,cursor="circle")

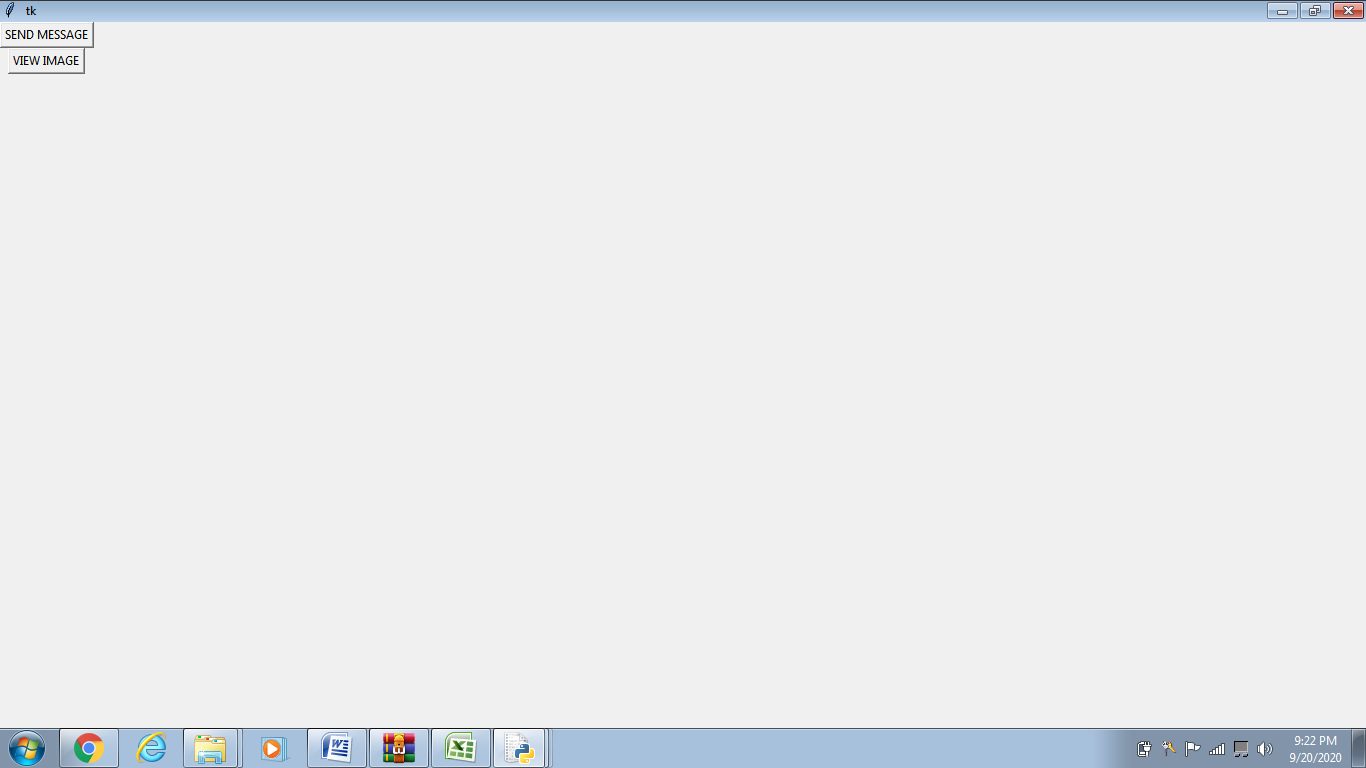
B.grid(row=1, column=0)

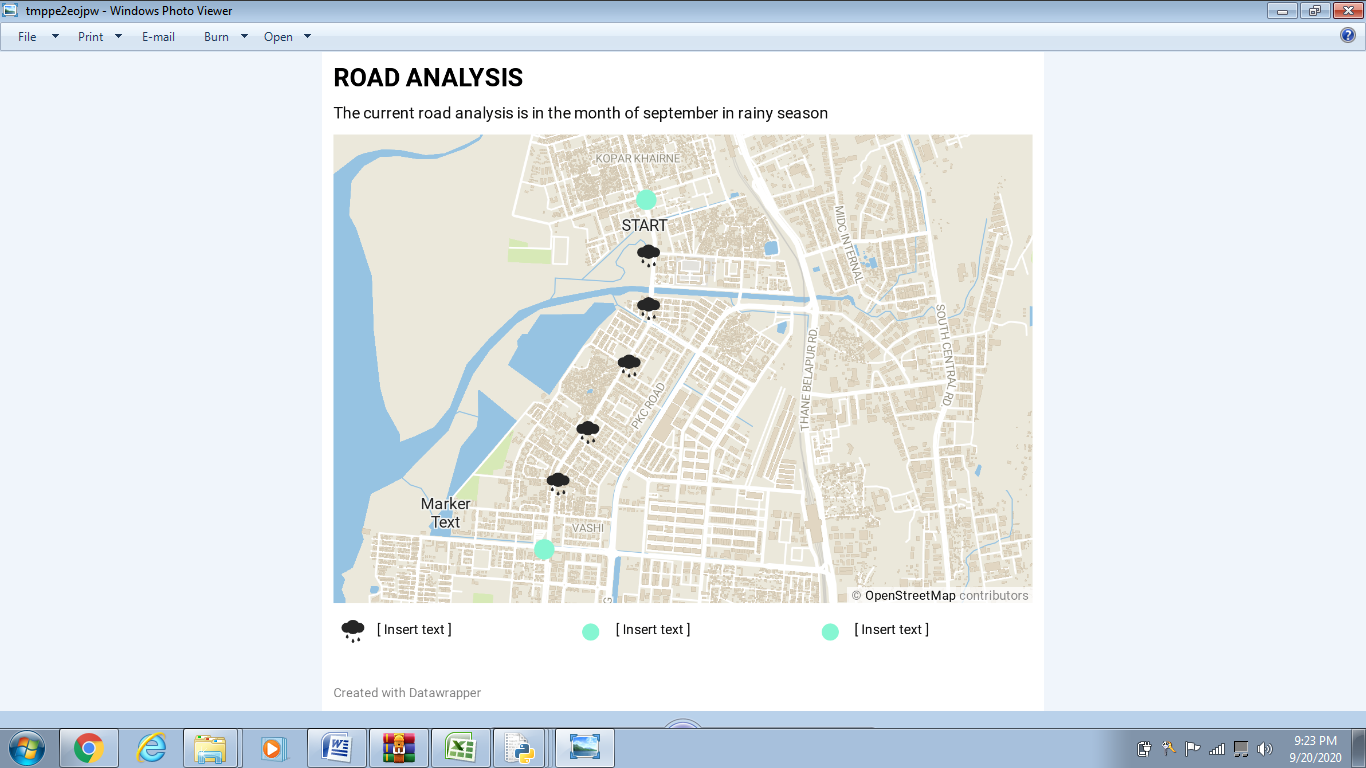
B1= tkinter.Button(top, text ="VIEW IMAGE", command = view,cursor="circle")

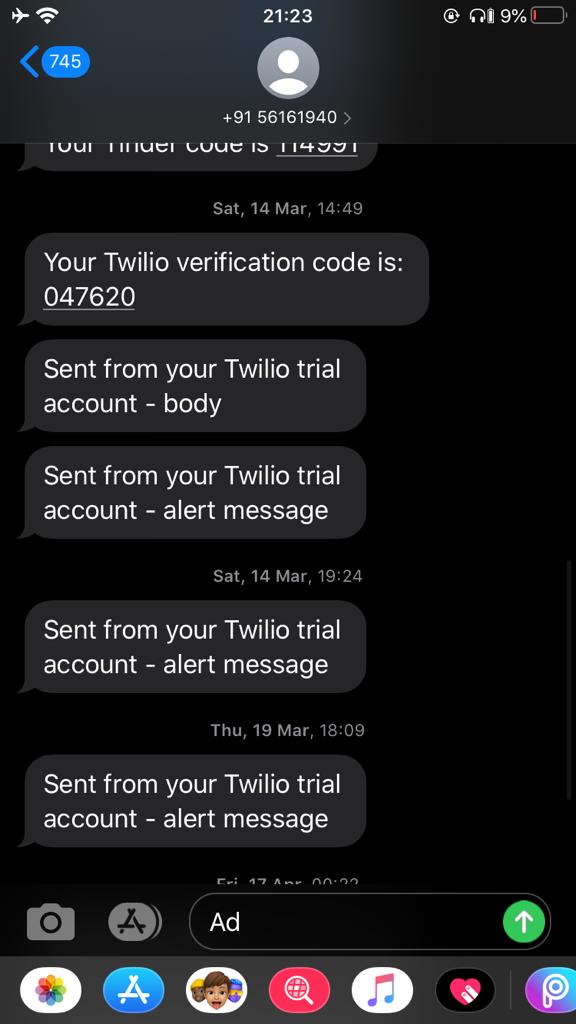
B1.grid(row=2, column=0)

#B = tkinter.Button(top, text ="SEND MESSAGE", command = send,cursor="circle")

#B.grid(row=1, column=0)





D

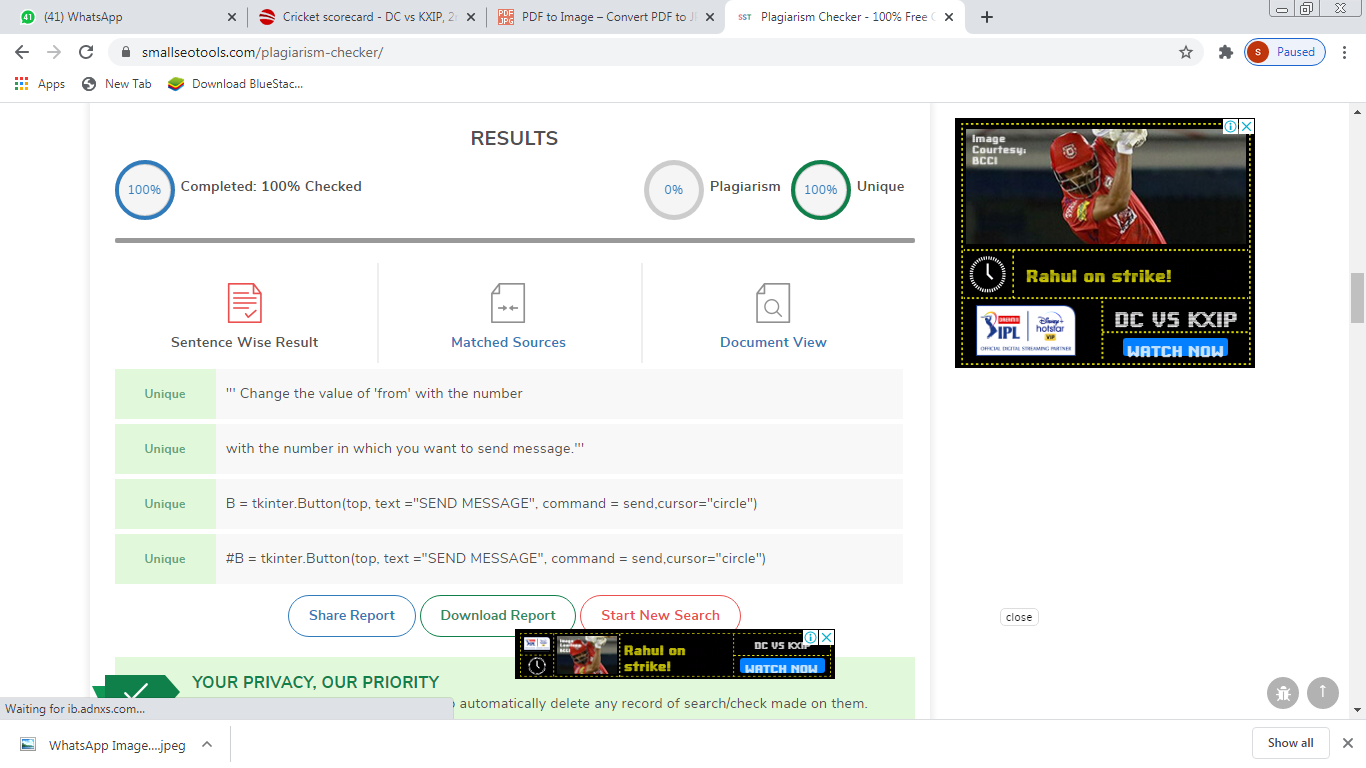
DATA VIEWING:-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Test Scenario | Test Steps | Test Data | Expected Results | Actual Results | Pass/Fail |
| T01 | When Click on Name . | 1. Go to Application 2. Click on button name | None | The user should see the name | As Expected. | Pass |
| T02 | When click on place | 1. Go to Application 2. Click on button place | None | The user should see place | as expected | Pass |
| T03 | When click on contractor | 1. Go to Application 2. Click on button | None | The user should see contractor | As expected | Pass |
| T04 | When click on date | 1. Go to Application 2. Click ondate | None | The user should see date | As Expected | Pass |

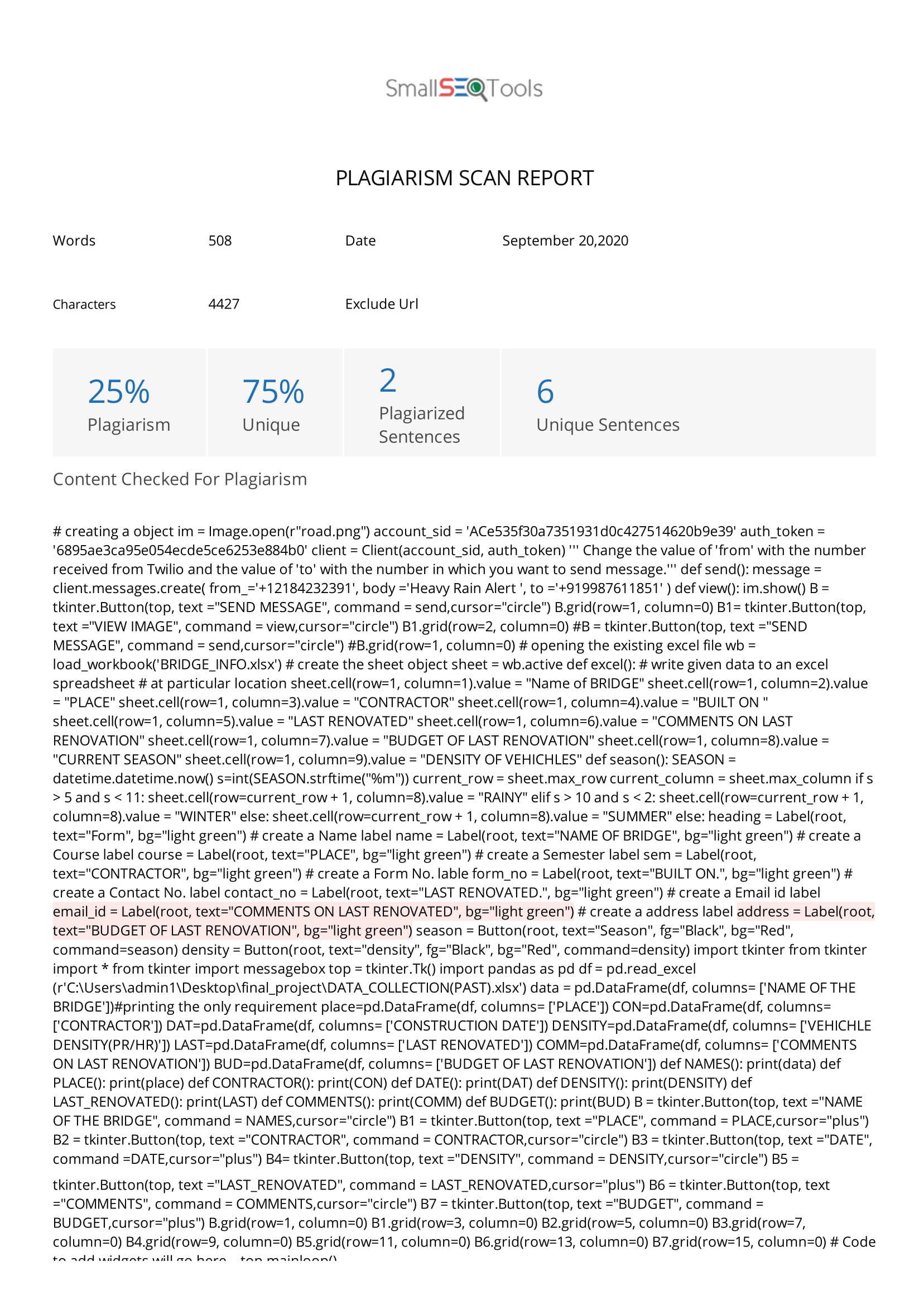
**Finalize Page:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Test Scenario | Test Steps | Test Data | Expected Results | Actual Results | Pass/Fail |
| T01 | When Click on send message | 1. Go to Application 2. Click on send message | None | The user should get Message | As Expected. | Pass |
| T02 | When Click on view map | 1. Go to Application 2. Click on view map | None | The user should get to view map | As Expected. | Pass |

REPORT:



.



**FUTURE ENHANCEMENT:**

Being a desktop application ,we thrive to enhance it in the following direction

1. Using drones to capture
2. Using machine learning so no human intervention will be needed
3. Giving live time updates
4. Keeping transparent data with people

**CONCLUSION:**

1. It will provide a smooth interface for the users
2. Data keeping will be easy and therefore will increase the work intrest
3. It will also help to alert users thus will help in reducing accidents

**References and Bibliography:**

|  |  |  |
| --- | --- | --- |
| Sr. No | Type | Description |
| 1. | Book Name: | * Python |
| 2. | Internet | * **Python/opencv** |