PROJECT REPORT

Of

A Desktop software for helping Flood affected people to be used at flood relief camp.

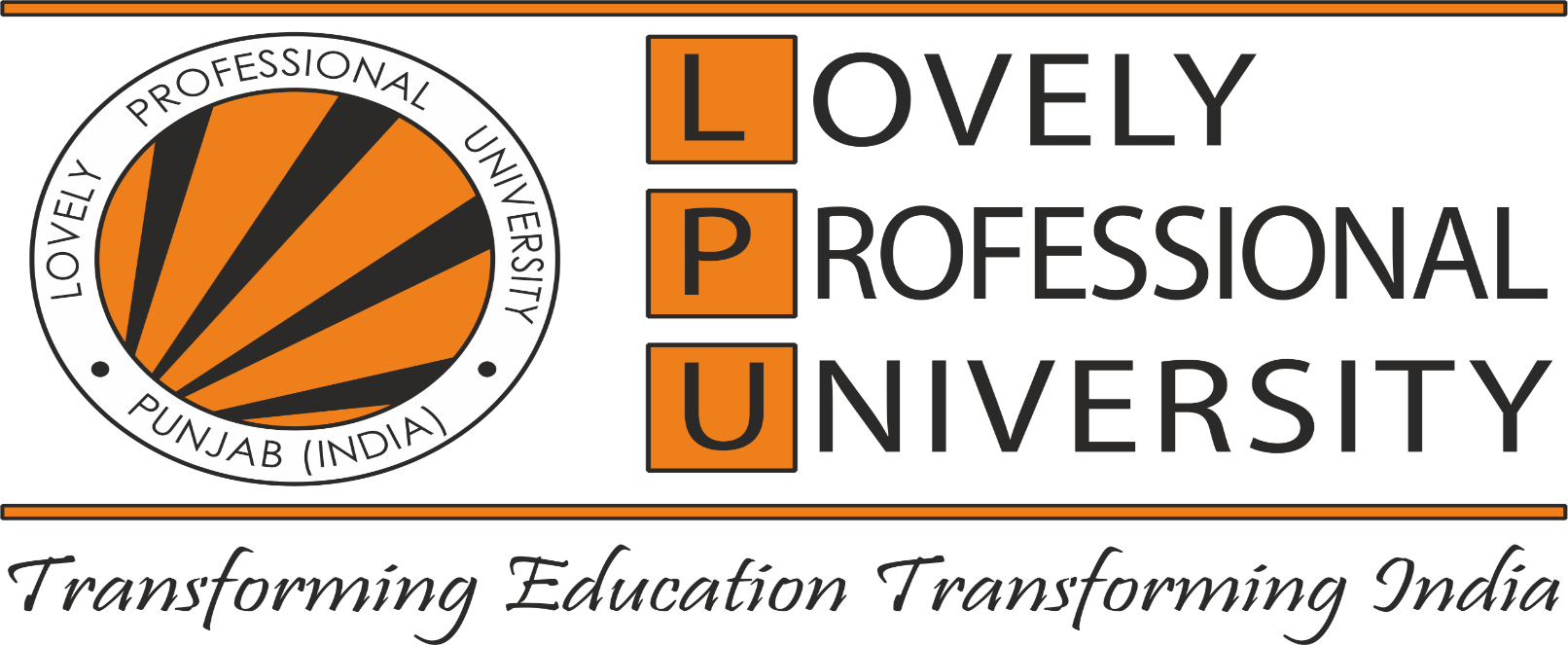
Submitted by:

1. S. Jahnavi Prasad, 11803179, R.no 42, K18RD
2. Shreedhar Niskawade, 11808727, R.no 67, K18RD

Submitted to:

Sagar Pandey Sir, Asst. Professor

**School of Computer Sciences and Engineering**

****

**AIM**

Our software will be used by mainly flood relief shelters during the period of ongoing disaster to monitor data of lost people, to provide shelter to the victims of the flood, to keep check on the supplies in stock for prolonged survival of the refugees.

We have total 4 modules in this software, they are, Details of missing people, Details of the refuges in the shelter, Details of supplies in the shelter and finally Google map on flood affected areas and nearest relief centers with directions.

**OBJECTIVE**

We are working on developing a database frame that can help us create first three modules, we will download real life data sets of Lost people, refugees and types of supplies collected in flood relief shelters and link them to our database system.

We are working on a way to incorporate a google map api in our project. This map will give detailed representation of flood affected regions and nearest flood shelter and directions to that place.

**INTRODUCTION**

**What is flood?**

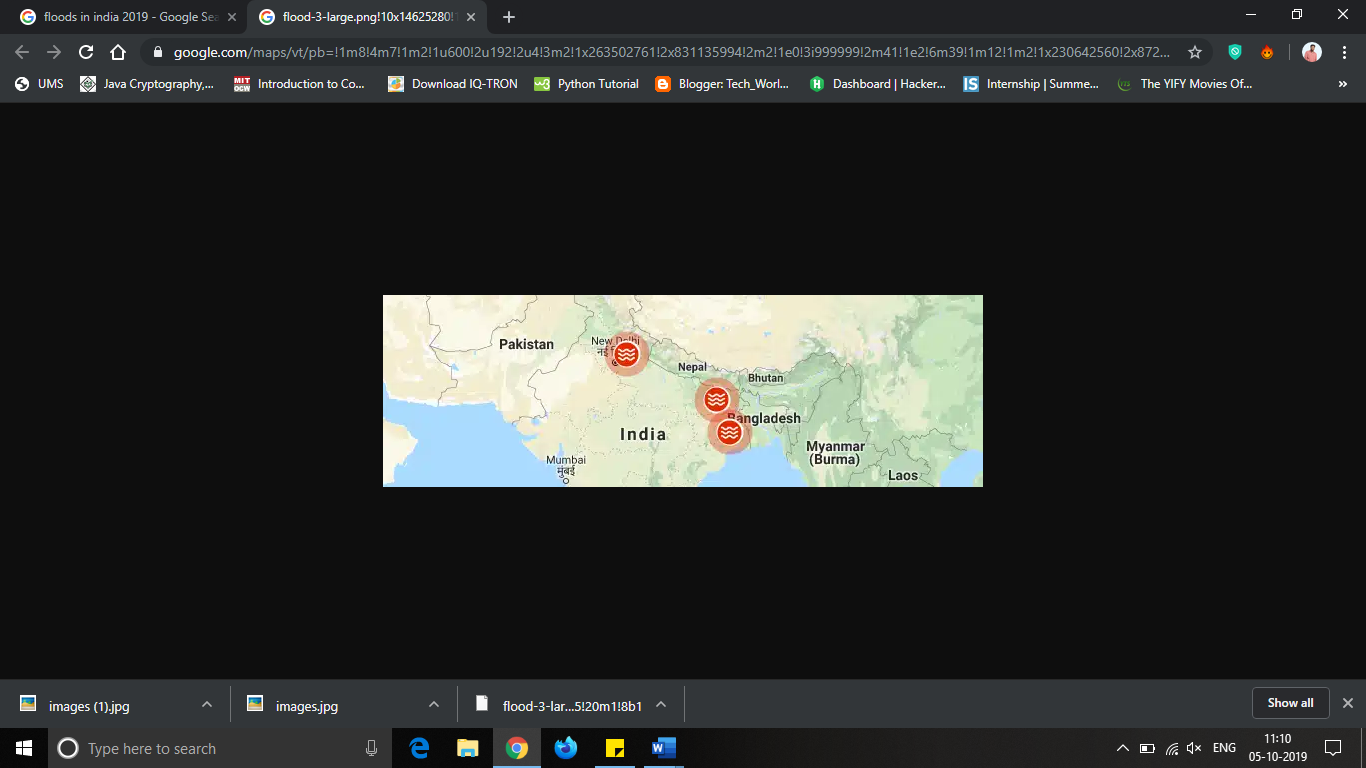
A **flood** is an overflow of water that submerges land. Floods are natural occurrences where an area or land that is normally dry abruptly is submerged in water.

**Effects of floods**

Flood has been considered as one of the most recurring and frequent disaster in the world. India also has continuously suffered by many flood events which claimed huge loss of life and economy.

**Recent floods**

In 2019 a series of floods affected over thirteen states in July and early August 2019, due to incessant rains. More than 1600 people died about a million people were displaced between June and October 2019. Karnataka and Maharashtra were the most severely affected states. It was the heaviest monsoon in last 25 years.

  A group of people in the water

Description automatically generated

Flood affected areas Odisha floods Karnataka floods as of September 2019 (12/07/19) (03/08/19)

**Motivation for this project**

After learning about the floods ravaging India, we decided to do something useful to help the brave souls working on rescue and relocation of victims, thus this project was born.

**ROLE OF TEAM MEMBERS**

1. S. Jahnavi Prasad – Coding (GUI, adding datasets, google maps api, data filtering ).
2. Shreedhar Niskawade – Data collection, analysis, Report preparation, google maps api.

**LIBRARIES USED**

**For Database management**

1. **Pandas-** for importing CSV files and displaying data

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the python programming language.

We will pandas to import our datasets in the program

And display them in the output for our modules.

1. **SYS** – used to import the entire system into the python module

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

**For Maps**

**Google maps 3.1.3**

This library brings the Google maps platform web services to python applications.

The python client for Google maps services is a python client library for following APIs:

* Directions API
* Distance matrix API
* Elevation API
* Geocoding API
* Geolocation API
* Time Zone API
* Roads API
* Places API

**Urllib –** used to fetch the urls from internet

Urllib module is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the *urlopen* function and is able to fetch URLs using a variety of different protocols.

Urllib is a package that collects several modules for working with URLs, such as:

* urllib.request for opening and reading.
* urllib.parse for parsing URLs
* urllib.error for the exceptions raised
* urllib.robotparser for parsing robot.txt files

**Json –** used to send service request to the target url to import co-ordinates of the map area.

**JSON**is a standard format for data exchange, which is inspired by JavaScript. Generally, JSON is in string or text format. **JSON**stands for**J**ava**S**cript **O**bject **N**otation.

JSON Library of Python performs following translation of Python objects into JSON objects by default converting Python data to JSON is called an Encoding operation. Encoding is done with the help of JSON library method – **dumps()**

**dumps()**method converts dictionary object of python into JSON string data format.

**SSL –** used to send service request to the target url to import co-ordinates of the map area.

This module provides access to Transport Layer Security (often known as “Secure Sockets Layer”) encryption and peer authentication facilities for network sockets, both client-side and server-side. This module uses the OpenSSL library.

**For Graphical User Interface (GUI)**

**Tkinter -** to create GUI

Tkinter is the standard GUI library for python when combined with Tkinter provides fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

**Page IDE –** for generating our GUI

PAGE is a cross-platform drag-and-drop GUI generator, bearing a resemblance to Visual Basic. It allows one to easily create GUI windows containing a selection of Tk and ttk widgets. Required are Tcl/Tk 8.6 and Python 2.7+. I am actually using Tcl/Tk 8.6 and Python 3.7. I am no longer responding to problem related to Python 2. PAGE springs from Virtual Tcl, a Tcl/Tk program, forked to generate Python modules that realize the desired GUI. Tcl is required for running PAGE but is not required for executing the generated Python code.

**Modules implemented**

We have total 4 modules in this software, they are:

1. Details Lost or presumed dead people

This module will contain a list of those people who were the victims of the flood and got lost with flow of water or died during the flood so as to keep a check on the total number of survivors in that area. It will also help us to initiate a search and rescue mission for them if they are still alive and holding their ground somewhere.

1. Details of the refugees in the shelter

The module will help us determine the total number of flood victims taking refuge in the shelter so as to maintain a check on supplies whether every person is getting enough food and water and other essentials or not and the most important thing is that we can share this data to government and ask them for helping in an evacuation mission if the shelter is near the flooded area.

1. Details Supplies in the shelter

The module will maintain a list of supplies such as food, water, clothes etc. and will help us to keep the people alive in the shelter for as long as possible until a evacuation mission or until the situation is handled.

1. Google maps for showing flood affected regions

We are basically using Google maps to show the user of this application the data (region) he/she will give as an input to the application will in return give the corresponding output of areas affected by the flood.

We used three major libraries for integrating google maps in our project, they are Json,Ssl,Urlib.

Json and Ssl used to send service request to the target url

to Import co-ordinates of the map.

Urllib used to fetch urls using urlopen and a variety of other

Protocols.

**Code**

#! /usr/bin/env python

# -\*- coding: utf-8 -\*-

#

# GUI module generated by PAGE version 4.26

# in conjunction with Tcl version 8.6

# Nov 11, 2019 04:24:56 PM IST platform: Windows NT

import sys

import pandas as pd

from geo import geomap

try:

import Tkinter as tk

except ImportError:

import tkinter as tk

try:

import ttk

py3 = False

except ImportError:

import tkinter.ttk as ttk

py3 = True

import HOME\_support

def vp\_start\_gui():

'''Starting point when module is the main routine.'''

global val, w, root

root = tk.Tk()

top = Toplevel1 (root)

HOME\_support.init(root, top)

root.mainloop()

w = None

def create\_Toplevel1(root, \*args, \*\*kwargs):

'''Starting point when module is imported by another program.'''

global w, w\_win, rt

rt = root

w = tk.Toplevel (root)

top = Toplevel1 (w)

HOME\_support.init(w, top, \*args, \*\*kwargs)

return (w, top)

def destroy\_Toplevel1():

global w

w.destroy()

w = None

class Toplevel1:

def \_\_init\_\_(self, top=None):

'''This class configures and populates the toplevel window.

top is the toplevel containing window.'''

\_bgcolor = '#d9d9d9' # X11 color: 'gray85'

\_fgcolor = '#32a89d' # X11 color: 'black'

\_compcolor = '#d9d9d9' # X11 color: 'gray85'

\_ana1color = '#d9d9d9' # X11 color: 'gray85'

\_ana2color = '#ececec' # Closest X11 color: 'gray92'

top.geometry("650x457+382+140")

top.minsize(120, 1)

top.maxsize(1370, 749)

top.resizable(1, 0)

top.title("HOME")

top.configure(background="#32a89d")

self.menubar = tk.Menu(top,font="TkMenuFont",bg=\_bgcolor,fg='#ff8040')

top.configure(menu = self.menubar)

self.Frame1 = tk.Frame(top)

self.Frame1.place(relx=0.015, rely=0.044, relheight=0.93, relwidth=0.962)

self.Frame1.configure(relief='groove')

self.Frame1.configure(borderwidth="2")

self.Frame1.configure(relief="groove")

self.Frame1.configure(background="#d9d9d9")

self.Canvas1 = tk.Canvas(self.Frame1)

self.Canvas1.place(relx=0.0, rely=0.0, relheight=1.0, relwidth=1.0)

self.Canvas1.configure(background="#50eb4d")

self.Canvas1.configure(borderwidth="2")

self.Canvas1.configure(insertbackground="black")

self.Canvas1.configure(relief="ridge")

self.Canvas1.configure(selectbackground="#a88732")

self.Canvas1.configure(selectforeground="black")

def open1():

df = pd.read\_excel(io='F:\PROJECT\FRMS\DATASETS\REFUGEES.xlsx')

print(df.head(2000))

def open2():

df = pd.read\_excel(io='F:\PROJECT\FRMS\DATASETS\MISSING\_PEOPLE.xlsx')

print(df.head(2000))

def open3():

df = pd.read\_excel(io='F:\PROJECT\FRMS\DATASETS\SUPPLIES.xlsx')

print(df.head(2000))

self.Button1 = tk.Button(self.Canvas1,command=lambda: open1())

self.Button1.place(relx=0.144, rely=0.306, height=24, width=151)

self.Button1.configure(activebackground="#ececec")

self.Button1.configure(activeforeground="#000000")

self.Button1.configure(background="#d9d9d9")

self.Button1.configure(disabledforeground="#a3a3a3")

self.Button1.configure(foreground="#000000")

self.Button1.configure(highlightbackground="#d9d9d9")

self.Button1.configure(highlightcolor="black")

self.Button1.configure(pady="0")

self.Button1.configure(text='''REFUGEE'S INFORMATION''')

self.Button2 = tk.Button(self.Canvas1,command=lambda: open2())

self.Button2.place(relx=0.592, rely=0.306, height=24, width=157)

self.Button2.configure(activebackground="#ececec")

self.Button2.configure(activeforeground="#000000")

self.Button2.configure(background="#d9d9d9")

self.Button2.configure(disabledforeground="#a3a3a3")

self.Button2.configure(foreground="#000000")

self.Button2.configure(highlightbackground="#d9d9d9")

self.Button2.configure(highlightcolor="black")

self.Button2.configure(pady="0")

self.Button2.configure(text='''MISSING PEOPLE'S DETAILS''')

self.Button3 = tk.Button(self.Canvas1,command=lambda: open3())

self.Button3.place(relx=0.144, rely=0.588, height=24, width=154)

self.Button3.configure(activebackground="#ececec")

self.Button3.configure(activeforeground="#000000")

self.Button3.configure(background="#d9d9d9")

self.Button3.configure(disabledforeground="#a3a3a3")

self.Button3.configure(foreground="#000000")

self.Button3.configure(highlightbackground="#d9d9d9")

self.Button3.configure(highlightcolor="black")

self.Button3.configure(pady="0")

self.Button3.configure(text='''SUPPLIES INFORMATION''')

self.Button4 = tk.Button(self.Canvas1,command=lambda: geomap())

self.Button4.place(relx=0.508, rely=0.588, height=24, width=250)

self.Button4.configure(activebackground="#ececec")

self.Button4.configure(activeforeground="#000000")

self.Button4.configure(background="#d9d9d9")

self.Button4.configure(disabledforeground="#a3a3a3")

self.Button4.configure(foreground="#000000")

self.Button4.configure(highlightbackground="#d9d9d9")

self.Button4.configure(highlightcolor="black")

self.Button4.configure(pady="0")

self.Button4.configure(text='''SHOW LATEST FLOOD AFFECTED AREA''')

self.Label1 = tk.Label(self.Canvas1)

self.Label1.place(relx=0.288, rely=0.094, height=50, width=300)

self.Label1.configure(background="#eb6f4d")

self.Label1.configure(disabledforeground="#a3a3a3")

self.Label1.configure(foreground="#000000")

self.Label1.configure(text='''FLOOD RELIEF SHELTER MANAGEMENT SYSTEM''')

self.Label2 = tk.Label(self.Canvas1)

self.Label2.place(relx=0.100, rely=0.894, height=30, width=500)

self.Label2.configure(background="#58d92e")

self.Label2.configure(disabledforeground="#58d92e")

self.Label2.configure(foreground="#000000")

self.Label2.configure(text='''Created by Jahnavi Prasad and Shreedhar Niskawade''')

def show\_refugee():

print("working")

@staticmethod

def popup1(event, \*args, \*\*kwargs):

Popupmenu1 = tk.Menu(root, tearoff=0)

Popupmenu1.configure(activebackground="#f9f9f9")

Popupmenu1.configure(activeborderwidth="5")

Popupmenu1.configure(activeforeground="black")

Popupmenu1.configure(background="#d9d9d9")

Popupmenu1.configure(borderwidth="5")

Popupmenu1.configure(disabledforeground="#a3a3a3")

Popupmenu1.configure(font="{Symbol} 30")

Popupmenu1.configure(foreground="black")

Popupmenu1.post(event.x\_root, event.y\_root)

if \_\_name\_\_ == '\_\_main\_\_':

vp\_start\_gui()

**Code - geo.py**

def geomap():

import urllib.request, urllib.parse, urllib.error

import json

import ssl

api\_key = False

# If you have a Google Places API key, enter it here

# api\_key = 'AIzaSy\_\_\_IDByT70'

# https://developers.google.com/maps/documentation/geocoding/intro

if api\_key is False:

api\_key = 42

serviceurl = 'http://py4e-data.dr-chuck.net/json?'

else:

serviceurl = 'https://maps.googleapis.com/maps/api/geocode/json?'

# Ignore SSL certificate errors

ctx = ssl.create\_default\_context()

ctx.check\_hostname = False

ctx.verify\_mode = ssl.CERT\_NONE

while True:

address = 'kerala'

if len(address) < 1:

break

parms = dict()

parms['address'] = address

if api\_key is not False: parms['key'] = api\_key

url = serviceurl + urllib.parse.urlencode(parms)

print('Retrieving', url)

uh = urllib.request.urlopen(url, context=ctx)

data = uh.read().decode()

print('Retrieved', len(data), 'characters')

try:

js = json.loads(data)

except:

js = None

print(json.dumps(js, indent=4))

lat = js['results'][0]['geometry']['location']['lat']

lng = js['results'][0]['geometry']['location']['lng']

print('lat', lat, 'lng', lng)

location = js['results'][0]['formatted\_address']

print(location)

**References**

<https://pypi.org/project/Pandas/>

<https://docs.python-guide.org/scenarios/db/>

<https://pypi.org/project/Ssl/>

<https://pypi.org/project/Json/>

<https://pypi.org/project/googlemaps/>

<https://github.com/googlemaps/google-maps-services-python>

<https://www.geeksforgeeks.org/python-gui-tkinter/>

<https://en.wikipedia.org/wiki/2019_Indian_floods>

<https://en.wikipedia.org/wiki/2019_Karnataka_floods>

<https://stackoverflow.com/questions/2505072/what-is-the-simplest-possible-payment-gateway-to-implement-using-django>

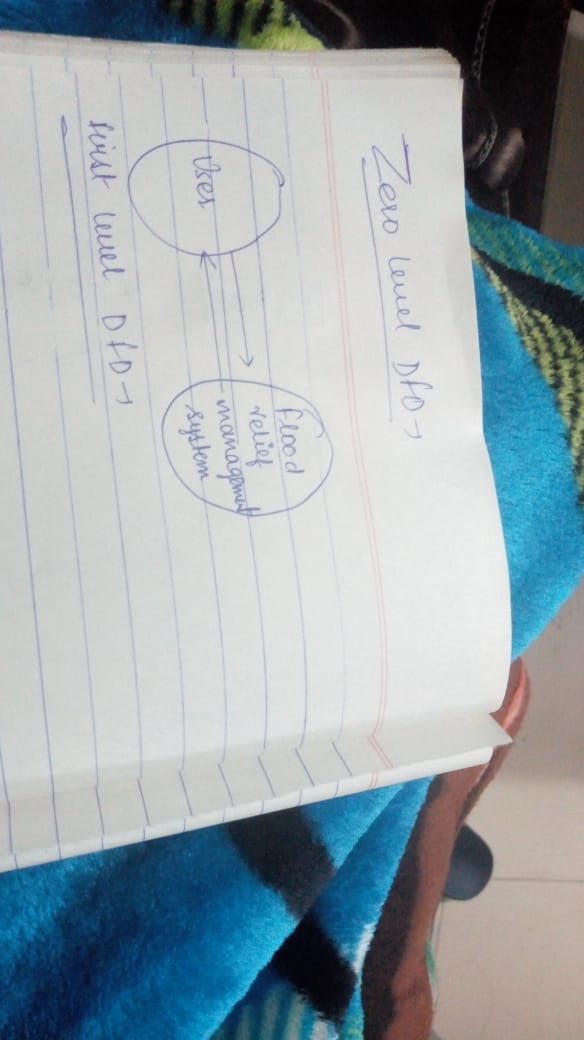
<https://github.com/abinshoby/Flood_Relief_Project>

<https://pypi.org/project/Urllib/>

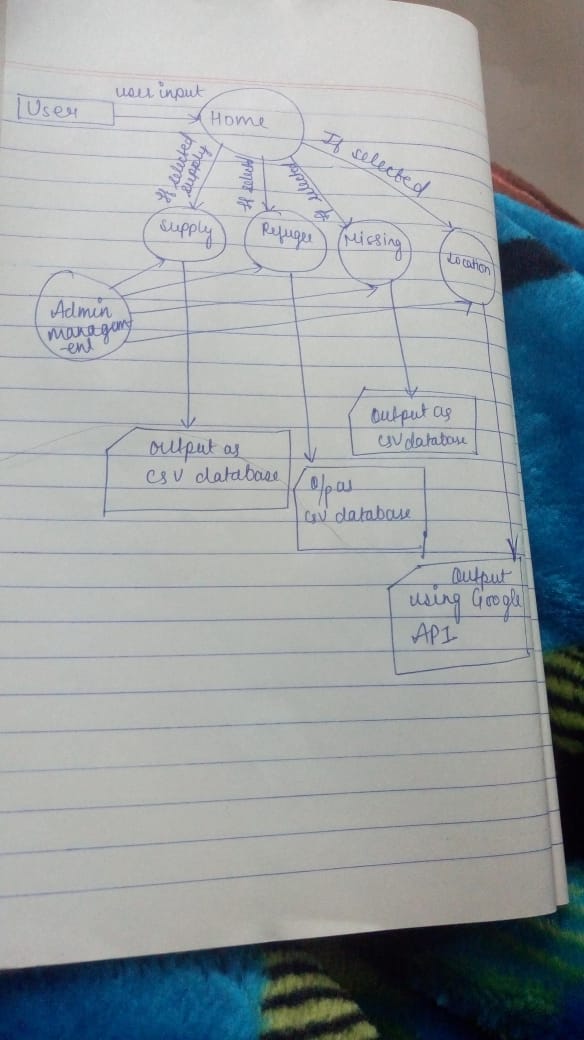
<https://pypi.org/project/Sys/>

**DFD – data flow diagram.**

**Level zero DFD**



**Level one DFD**

****