Contents

1 views.py

```python
from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.

def index(request):

    # Page from the theme
    return render(request, 'pages/custom-index.html')
```

2 tests.py

```python
from django.test import TestCase

# Create your tests here.
```

3 urls.py

```
from django.urls import path, include

from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('api/', include('api.urls')),
]
```

4 admin.py

```python
from django.contrib import admin

# Register your models here.
```

5 models.py

```python
from django.db import models

# Create your models here.
```

6 apps.py

```
from django.apps import AppConfig


class HomeConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "home"
```

7 templates/pages/forms/indicators.html

```
<div class="container">
  <div class="row">
    <div style="display:flex;flex-direction:row;">
      <div style="width:50%;">
        <table class="table␣table-hover">
          <tbody>
            <tr>
              <td style="text-align:left;vertical-align:middle;"><label for="
                  suspectName">Suspect Name</label></td>
              <td><input type="text" v-model="suspect.name" class="form-control"
                   id="suspectName"></td>
            </tr>
            <tr>
              <td style="text-align:left;vertical-align:middle;"><label for="
                  suspectLinks">Suspect Links</label></td>
              <td><input type="text" v-model="suspect.links" class="form-control
                  " id="suspectLinks"><input type="hidden" v-model="suspect.id"><
                  /td>
            </tr>
        </table>
      </div>
      <div style="display:flex;align-items:center;justify-content:space-evenly;
          flex-direction:column;width:100%;max-width:50%">
        <div>
        <button class="btn␣btn-primary" type="button" @click="apiComputeSuspect"
            >COMPUTE</button>
        </div>
        <div>
        <button class="btn␣btn-danger" type="button" @click="resetIndicators">
            RESET</button>
        </div>
      </div>
    </div>
  </div>
  <div class="row␣mb-4␣mb-lg-5">
    <div class="mb-4␣horizontal" style="display:flex;flex-direction:row;">
      <table class="table␣table-hover">
        <thead>
          <th></th>
          <th>Date</th>
          <th>Location</th>
        </thead>
        <tbody>
          <tr v-for="indicator␣in␣indicators" class="indicator" @click="
              addRemoveIndicator">
            <td style="text-align:␣left;">
              <label>[[ indicator.name ]]</label>
            </td>
            <td>
              <input type="date" class="form-control" id="indicator_date"
                  @change="updateModifiedIndicators">
            </td>
            <td>
              <input type="text" class="form-control" id="indicator_loc" @change
                  ="updateModifiedIndicators">
              <input type="hidden" :value="indicator.id" id="indicator_id">
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

8 templates/pages/layouts/modal.html

```
<div class="modal␣fade" id="modal-default" tabindex="-1" role="dialog" aria-
    labelledby="modal-default" aria-hidden="true" style="background:rgba
    (0,0,0,0.2);" @click="hideModal">
  <div class="modal-dialog␣modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h2 class="h6␣modal-title">[[ modal.title ]]</h2>
        <button type="button" class="btn-close" data-bs-dismiss="modal"
            aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p>[[ modal.body ]]</p>
      </div>
      <div class="modal-footer">
      </div>
    </div>
  </div>
</div>
```

9 templates/pages/layouts/assessment.html

```
<div class="container">
  <div class="row">
    <div style="display:flex;flex-direction:row;">
      <div class="mt-4" style="display:flex;align-items:center;␣flex-direction:
          column;width:50%">
        <div>
        <h3>PROBABILITY</h3>
        </div>
        <div style="border-color:black;border-style:solid;border-radius:50%;
            padding:2rem;">
        <h3>[[ suspect.score ]]%</h3>
        </div>
      </div>
      <div class="mt-4" style="text-align:left">
        <h5>SUSPECT NAME: [[ suspect.name ]]</h5>
        <p>FIRST SURVEILLANCE: [[ suspect.first_surv ]]</p>
        <p>SECOND SURVEILLANCE: [[ suspect.second_surv ]]</p>
      </div>
    </div>
  </div>
  <br>
  <div class="row">
    <div class="mt-4">
      <h3>ESTIMATED LOCATION</h3>
      <div id="suspect-location-map">
      </div>
    </div>
  </div>
</div>
```

10 templates/pages/layouts/base.html

```
<!--

=========================================================
* Pixel Free Bootstrap 5 UI Kit
=========================================================

* Product Page: https://themesberg.com/product/ui-kit/pixel-free-bootstrap-5-ui-
    kit
* Copyright 2021 Themesberg (https://www.themesberg.com)

* Coded by https://themesberg.com

=========================================================

* The above copyright notice and this permission notice shall be included in all
     copies or substantial portions of the Software. Contact us if you want to
    remove it.

-->

{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta http-equiv="Content-Type" content="text/html;␣charset=utf-8" />
    <!-- Primary Meta Tags -->
    <title>App</title>
    <meta name="viewport" content="width=device-width,␣initial-scale=1,␣shrink-
        to-fit=no">
    <meta name="title" content="App">


    <!-- Favicon -->
    <link rel="apple-touch-icon" sizes="120x120" href="{%␣static␣'assets/img/
        favicon/apple-touch-icon.png'␣%}">
    <link rel="icon" type="image/png" sizes="32x32" href="{%␣static␣'assets/img/
        favicon/favicon-32x32.png'␣%}">
    <link rel="icon" type="image/png" sizes="16x16" href="{%␣static␣'assets/img/
        favicon/favicon-16x16.png'␣%}">
    <link rel="manifest" href="{%␣static␣'assets/img/favicon/site.webmanifest'␣
        %}">
    <link rel="mask-icon" href="{%␣static␣'assets/img/favicon/safari-pinned-tab.
        svg'␣%}" color="#ffffff">
    <meta name="msapplication-TileColor" content="#ffffff">
    <meta name="theme-color" content="#ffffff">

    <!-- Fontawesome -->
    <link type="text/css" href="{%␣static␣'vendor/@fortawesome/fontawesome-free/
        css/all.min.css'␣%}" rel="stylesheet">

    <!-- Pixel CSS -->
    <link type="text/css" href="{%␣static␣'css/pixel.css'␣%}" rel="stylesheet">
    <link type="text/css" href="{%␣static␣'css/datatables.min.css'␣%}" rel="
        stylesheet">
    <script src="{%␣static␣'js/vue.js'␣%}"></script>
    <script src="{%␣static␣'js/datatables.min.js'␣%}"></script>
    <script src="{%␣static␣'js/echarts.min.js'␣%}"></script>

</head>

<body style="background-color:#1c2540;">

  {% block header %}


  {% endblock header %}

  {% block content %}{% endblock content %}

    <script src="{%␣static␣'js/app.js'␣%}"></script>

  {% block footer %}


  {% endblock footer %}

  {% include 'includes/scripts.html' %}
  {% block javascripts %}
<script>
  app.apiListIndicators();
</script>
  {% endblock javascripts %}


</body>
    <script src="{%␣static␣'js/bokeh-3.4.1.min.js'␣%}"></script>
    <script src="{%␣static␣'js/bokeh-widgets-3.4.1.min.js'␣%}"></script>
    <script src="{%␣static␣'js/bokeh-tables-3.4.1.min.js'␣%}"></script>
    <script src="{%␣static␣'js/bokeh-api-3.4.1.min.js'␣%}"></script>
    <script src="{%␣static␣'js/bokeh-gl-3.4.1.min.js'␣%}"></script>
    <script src="{%␣static␣'js/bokeh-mathjax-3.4.1.min.js'␣%}"></script>

</html>
```

11 templates/pages/layouts/database.html

```
<div class="container">
  <div class="row">
    <table id="suspect-table" class="hover">
    </table>
  </div>
</div>
```

12 templates/pages/layouts/network.html

```
<div class="container" style="width:␣900px;height:600px;" id="network-chart">
</div>
```

13 templates/pages/custom-index.html

```
{% extends 'pages/layouts/base.html' %}
{% load static %}

{% block content %}

<main>
  <section class="section␣section-lg␣bg-secondary">
    <div class="container">
      <div class="card␣shadow-soft␣border-soft␣bg-gray-200␣p-5␣text-center␣mb-4"
          >
        <div class="row␣justify-content-center">
          <div class="col-12␣col-md-12␣col-lg-12">
            <!-- Tab Nav -->
            <div class="nav-wrapper␣position-relative␣mb-2">
              <ul class="nav␣nav-pills␣nav-fill␣flex-column␣flex-md-row" id="
                  tabs-text" role="tablist">
                <li class="nav-item">
                  <a class="nav-link␣mb-sm-3␣mb-md-0␣active" id="tabs-text-1-tab
                      " data-bs-toggle="tab"
                  href="#tabs-text-1" role="tab" aria-controls="tabs-text-1"
                      aria-selected="true" @click="apiListIndicators">Indicators<
                      /a>
                </li>
                <li class="nav-item">
                  <a class="nav-link␣mb-sm-3␣mb-md-0" id="tabs-text-2-tab" data-
                      bs-toggle="tab" href="#tabs-text-2" role="tab" aria-
                      controls="tabs-text-2" aria-selected="false" @click="
                      apiGetLocationMap">Assessment</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link␣mb-sm-3␣mb-md-0" id="tabs-text-3-tab" data-
                      bs-toggle="tab"
                    href="#tabs-text-3" role="tab" aria-controls="tabs-text-3"
                    aria-selected="false" @click="apiGetNetwork">Networks</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link␣mb-sm-3␣mb-md-0" id="tabs-text-4-tab" data-
                      bs-toggle="tab"
                    href="#tabs-text-4" role="tab" aria-controls="tabs-text-4"
                    aria-selected="false" @click="apiListSuspects">Database</a>
                </li>
                <li class="nav-item">
                  <a class="nav-link␣mb-sm-3␣mb-md-0" id="tabs-text-5-tab" data-
                      bs-toggle="tab"
                    href="#tabs-text-5" role="tab" aria-controls="tabs-text-5"
                    aria-selected="false">About</a>
                </li>
              </ul>
            </div>
            <!-- End of Tab Nav -->
            <!-- Tab Content -->
            <div class="card␣border-0">
              <div class="card-body␣p-0">
                <div class="tab-content" id="tabcontent1">
                  <div class="tab-pane␣fade␣show␣active" id="tabs-text-1" role="
```

```
                      tabpanel"
                    aria-labelledby="tabs-text-1-tab">
                    {% include 'pages/forms/indicators.html' %}
                  </div>
                  <div class="tab-pane␣fade" id="tabs-text-2" role="tabpanel"
                    aria-labelledby="tabs-text-2-tab">
                    {% include 'pages/layouts/assessment.html' %}
                  </div>
                  <div class="tab-pane␣fade" id="tabs-text-3" role="tabpanel"
                    aria-labelledby="tabs-text-3-tab">
                    {% include 'pages/layouts/network.html' %}
                  </div>
                  <div class="tab-pane␣fade" id="tabs-text-4" role="tabpanel"
                    aria-labelledby="tabs-text-4-tab">
                    {% include 'pages/layouts/database.html' %}
                  </div>
                  <div class="tab-pane␣fade" id="tabs-text-5" role="tabpanel"
                    aria-labelledby="tabs-text-5-tab">
                      <p>This software is the result of research conducted as
                          part of a PhD program in Conflict Resolution at the
                          University of Vigo.</p>
                      <p>Calculations are based on all terrorist attacks occured
                           in the West between 2010 and 2020.</p>
                      <p>The parameters used by the sofware have a tested
                          effectiveness of ~70% in reducing fatality rates.</p>
                      <p>The recommended surveillance periods are 4 months for
                          the first surveillance, and 8 months for the second one
                          .</p>
                  </div>
                </div>
              </div>
            </div>
          <!-- End of Tab Content -->
          </div>
        </div>
      </div>
    </div>
    {% include 'pages/layouts/modal.html' %}
  </section>
  </main>

{% endblock content %}
{% block javascripts %}
<script>
  app.apiListIndicators();
  app.csrf = "{{␣csrf_token␣}}";
</script>
{% endblock javascripts %}
```

14 static/js/app.js

```
var app = new Vue({
  el:'main',
  delimiters:['[[', ']]'],
  data: {
    error : {
      title: "ERROR",
      body: "Unknown␣Error",
    },
    indicators: [],
    modifiedIndicators: {},
    suspect: {
      id: -1,
      name: "",
      links: "",
      score: 0,
      first_surv: "",
      second_surv: "",
    },
    suspectList : [],
    modal : {
      title : "",
      body : "",
    },
    csrf: "",
    suspectTable: null,
  },
  methods: {
    /* Helpers */
    updateSuspectTable: function() {
      if (this.suspectTable === null) {
        this.suspectTable = new DataTable("#suspect-table", {
          columns: [
            { title: 'Name' },
            { title: 'Probability' },
            { title: 'First Surv.' },
            { title: 'Second Surv.' },
            { title: 'Location' },
            { title: 'Delete' },
          ],
          data: this.suspectList
        });
      } else {
        this.suspectTable.clear();
        this.suspectTable.rows.add(this.suspectList);
        this.suspectTable.draw();
      }

    },
    sendGetRequest: async function(url, params) {
      let g_url = url;
      if (Object.keys(params).length > 0) {
        g_url += '?';
        for (let key in params) {
```

```
      g_url += key + '=' + params[key] + '&';
    }
    g_url = g_url.slice(0, g_url.length - 1);
  }
  let response = await fetch(g_url);
  let jsonResp = await response.json();

  return jsonResp;
},
sendPostRequest: async function(url, data) {
  const response = await fetch(url, {
    method: 'POST',
    mode: 'same-origin',
    cache: 'no-cache',
    credentials: 'same-origin',
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'X-CSRFTOKEN': this.csrf,
    },
    origin: 'http://127.0.0.1',
    redirect: 'follow',
    body: JSON.stringify(data),
  });
  try {
    let jsonResp = await response.json();
    if (jsonResp["error"]) {
      return false;
    } else {
      return true;
    }
  } catch (err) {
    return false;
  }
},
resetIndicators: function() {
  this.suspect = {
    id: -1,
    name: "",
    links: "",
    score: 0,
    first_surv: "",
    second_surv: "",
  };
  this.modifiedIndicators = {};
  let tags = document.getElementsByTagName("input");
  for (var i = 0; i < tags.length; ++i) {
    if (tags[i].getAttribute("type") === "hidden") {
      continue;
    }
    tags[i].value = "";
  }
  let rows = document.getElementsByTagName("tr");
  for (var i = 0; i < rows.length; ++i) {
    rows[i].style.background = '';
  }
  let plot = document.getElementById("suspect-location-map");
  plot.innerHTML = "";
```

```
},
updateModifiedIndicators: function() {
  let indicators = document.getElementsByClassName("indicator");
  for (var i = 0; i < indicators.length; ++i) {
    let indicatorId = indicators[i].children[2].children[1].value;
    let indicatorDate = indicators[i].children[1].children[0].value;
    let indicatorLocation = indicators[i].children[2].children[0].value;
    if (indicatorDate != "" || indicatorLocation != "") {
      this.modifiedIndicators[indicatorId] = {
        date: indicatorDate,
        loc: indicatorLocation
      };
      indicators[i].style.background = 'gray';
    }
  }
},
addRemoveIndicator: function(element) {
  let target = element.target;
  while (target.tagName != "TR") {
    target = target.parentElement;
    if (target.tagName == "BODY") {
      return;
    }
  }
  let indicatorId = target.children[2].children[1].value;
  if (target.style.background == 'gray') {
    target.style.background = '';
    delete this.modifiedIndicators[indicatorId];
  } else {
    target.style.background = 'gray';
    let indicatorDate     = target.children[1].children[0].value;
    let indicatorLocation = target.children[2].children[0].value;
    this.modifiedIndicators[indicatorId] = {
      date: indicatorDate,
      loc: indicatorLocation
    };
  }
},
reloadSuspect: function(el) {
  this.resetIndicators();
  this.suspect.name = el.text;
  this.suspect.links = "";
  this.apiGetSuspect();
},
showModal : function(title, body) {
  let m = document.getElementById("modal-default");
  m.classList.add("show");
  m.style.display = "block";
  this.modal.title = title;
  this.modal.body = body;
},
hideModal : function() {
  let m = document.getElementById("modal-default");
  m.classList.remove("show");
  m.style.display = "none";
},
/* API Functions */
```

```
apiDeleteSuspect: async function(suspectName) {
  let url = "/api/suspect/delete";
  let params = {name: suspectName};
  let reqSuccess = await this.sendPostRequest(url, params);
  if (reqSuccess) {
    this.showModal("Success", "Successfully␣deleted␣suspect");
    this.apiListSuspects();
  }
},
apiListIndicators : async function() {
  let url = "/api/indicators/list";
  let params = {};

  let resp = await this.sendGetRequest(url, params);
  if (resp.error == false) {
    this.indicators = resp.indicators;
  }
},
apiComputeSuspect : async function() {
  let url = "/api/suspect/update";
  let params = {
    id : this.suspect.id,
    name : this.suspect.name,
    links : this.suspect.links,
    indicators : this.modifiedIndicators,
  };
  this.showModal("Loading...", "Please␣wait␣while␣the␣score␣is␣being␣
      computed");
  let requestSuccess = await this.sendPostRequest(url, params);
  this.hideModal();
  if (!requestSuccess) {
    this.showModal("Error","Failed␣to␣add␣suspect␣location␣to␣database.␣
        Please␣try␣again␣later");
  }
  this.apiGetSuspect();
},
apiGetSuspect : async function() {
  let url = "/api/suspect/info";
  let params = {
    name : this.suspect.name,
  };
  let resp = await this.sendGetRequest(url, params);
  if (resp.error == false) {
    this.suspect = resp.suspect;
    document.getElementById("tabs-text-2-tab").click();
  }
},
apiListSuspects : async function() {
  let url = "/api/suspect/list";
  let params = {};
  let resp = await this.sendGetRequest(url, params);
  if (resp.error == false) {
    this.suspectList = resp.suspectList;
    this.updateSuspectTable();
  }
},
apiGetNetwork : async function() {
  let url = "/api/suspect/links"
  let params = {};
  let resp = await this.sendGetRequest(url, params);

  if (resp.error == false) {
    var chartDom = document.getElementById('network-chart');
    var myChart = echarts.init(chartDom);
    var option;

    option = {
      title: {
        text: 'Suspect Network',
        top: 'bottom',
        left: 'right',
      },
      tooltip: {},
      legend: [{data: resp.nodes.map(function(a) {return a.name;})}],
      animationDurationUpdate: 1500,
      animationEasingUpdate: 'quinticInOut',
      series:[
        {
          name: 'Suspect Network',
          type: 'graph',
          layout: 'circular',
          circular: {rotateLabel: true},
          data: resp.nodes,
          links: resp.edges,
          categories: resp.nodes.map(function(a) {return {name: a.name};}),
          roam: true,
          label: {
            position: 'right',
            formatter: '{b}'
          },
          lineStyle: {
            color: 'source',
            curveness: 0.3
          },
          emphasis: {
            focus: 'adjacency',
            lineStyle: {
              width: 10
            }
          }
        }
      ],
    };
    myChart.setOption(option);
  }
},
apiGetLocationMap: async function() {
  let url = '/api/suspect/location';
  let params = {'name':this.suspect.name};
  let resp = await this.sendGetRequest(url, params);
  if (resp.error == false) {
    let plot = document.getElementById("suspect-location-map");
    plot.innerHTML = "";
    var scr = document.createElement("script");
```

```
        scr.text = resp.script;
        //eval(scr.text);
        var div = document.createElement("div");
        div.innerHTML = resp.div;
        plot.appendChild(div);
        plot.appendChild(scr);
      } else {
        this.showModal("ERROR", resp.reason);
      }
    },

  },
});
```

Contents

1 views.py

```
import json
import datetime
import numpy as np
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut

import xyzservices.providers as xyz
from bokeh.plotting import figure, show
from bokeh.tile_providers import get_provider
from bokeh.embed import components
from bokeh.models import TapTool, WheelZoomTool, HoverTool
from bokeh.models import ColumnDataSource

from dateutil.relativedelta import relativedelta

from django.http import JsonResponse
from django.db.models import Q
from django.shortcuts import render
from django.utils import timezone

from .models import (
    Indicator,
    Suspect,
    SuspectIndicator,
    SuspectLink,
)

from .error import (
    api_error,
    api_success,
    INVALID_METHOD,
    INVALID_SUSPECT,
    SUSPECT_UPDATE_OK,
)

FIRST_SURVEILLANCE_INDICATORS = set([
    'Criminal␣suspect',
    'Contacting␣authorities␣about␣their␣intentions',
    'Engaging␣in␣conspiracy',
    'Contact␣with␣homeland␣terrorist',
    'Direction␣of␣a␣cell',
    'Confessing␣cellmates␣about␣intentions',
    'Confessing␣terror␣charges␣from␣another␣state',
    'Travelling␣to␣war␣zone␣or␣region␣with␣insurrectional␣activity',
    'Radical␣statements␣made␣public',
    'Considered␣as␣a␣threat',
    'Specific␣foreign␣intelligence␣warnings',
    'Membership␣of␣radical␣group',
    'Sentences␣for␣terrorism',
    'Terrorism␣recruitment␣or␣training␣for␣foreign␣conflict',
    'Adopting␣salafist␣behaviours',
    'Contact␣with␣foreign␣terrorist',
    'Participation␣in␣jihadist␣insurgency␣abroad',
    'Relative␣contacting␣authorities␣about␣their␣intentions',
    'Threatening␣institutions',
    'Dissemination␣of␣radical␣propaganda',
    'Search␣of␣radical␣websites',
    'Criminal␣links'
])

# Create your views here.
def add_suspect_link(suspect1, suspect2, is_direct):
    if suspect1 == suspect2:
        return
    if suspect1.name < suspect2.name:
        SuspectLink.objects.create(
            origin=suspect1,
            destin=suspect2,
            is_direct=is_direct
        )
    else:
        SuspectLink.objects.create(
            origin=suspect2,
            destin=suspect1,
            is_direct=is_direct
        )

def get_coordinates(location):
    geolocator = Nominatim(user_agent="LocationAppv1.0")
    geolocation = geolocator.geocode(location, timeout=10)
    longitude, latitude = wgs84_to_web_mercator(
        geolocation.longitude,
        geolocation.latitude)
    return longitude, latitude

def update_suspect_location(suspect, current_indicators):
    """
    Updates the suspect location.
    Suspect location only shows if there is date or:
      - If there is only one indicator sent and no other indicator has location
          and date
    """
    db_indicators = SuspectIndicator.objects.filter(Q(suspect=suspect) & ~Q(
        location=None)).order_by('-date').all()
    latest_indicator = None
    for indicator in db_indicators:
        try:
            if indicator.date is not None and latest_indicator.date < indicator.
                date:
                latest_indicator = indicator
        except AttributeError:
            if indicator.date is not None:
                latest_indicator = indicator
    if latest_indicator is None:
        if len(current_indicators) == 1:
            for _ind_id, ind in current_indicators.items():
                if ind['loc']:
                    suspect.longitude, suspect.latitude = get_coordinates(ind['
                        loc'])
                    suspect.location = ind['loc']
                    suspect.save()
    else:
        suspect.longitude, suspect.latitude = get_coordinates(latest_indicator.
            location)
        suspect.location = latest_indicator.location
        suspect.save()



def wgs84_to_web_mercator(lon, lat):
    coords = [lon, lat]
    k = 6378137
    coords[0] = coords[0] * (k * np.pi/180.0)
    coords[1] = np.log(np.tan((90 + coords[1]) * np.pi/360.0)) * k
    return coords

def is_indicator_in_DB(indicator, db_indicators):
    for dbi in db_indicators:
        if indicator['name'] == dbi.name:
            return True
    return False

def IndicatorList(request):
    if request.method != 'GET':
        return api_error(INVALID_METHOD)
    indicators = Indicator.objects.all()
    json_indicators = map(lambda x: {
        "id" : x.id,
        "name" : x.name,
    }, indicators)

    json_resp = {
        "error": False,
        "indicators" : list(json_indicators)
    }
    return JsonResponse(json_resp)

def ComputeSuspectProbability(request):
    if request.method != 'POST':
        return api_error(INVALID_METHOD)

    suspect_info = json.loads(request.body)


    # Try to retrieve by name
    suspect = Suspect.objects.filter(name=suspect_info['name'].strip()).first()
    # Add suspect to the database
    if suspect is None:
        suspect = Suspect.objects.create(
            name = suspect_info['name'].strip(),
        )
    # Add links to the database
    try:
        links = [x.strip() for x in suspect_info['links'].split(',')]
    except KeyError:
        links = []

    for link in links:
```

```
    if link == "":
        continue
    to_suspect = Suspect.objects.filter(name=link).first()
    if to_suspect is None:
        to_suspect = Suspect.objects.create(name=link)
    # Suspect links are always alphabetical, origin -> destin
    suspect_link = SuspectLink.objects.filter((Q(origin=suspect) & Q(destin=
        to_suspect)) | (Q(origin=to_suspect) & Q(destin=suspect))).first()
    if suspect_link is None:
        add_suspect_link(suspect, to_suspect, True)
    elif suspect_link.is_direct == False:
        add_suspect_link(suspect, to_suspect, True)

# Build indirect links
for j in range(len(links)-1):
    if links[j] == "":
        continue
    suspect1 = Suspect.objects.filter(name=links[j]).first()
    suspect2 = Suspect.objects.filter(name=links[j+1]).first()
    if suspect2 is None:
        continue
    if suspect2.name == suspect1.name:
        continue

    indirect_link = SuspectLink.objects.filter(
        (Q(origin=suspect1) & Q(destin=suspect2)) | (Q(origin=suspect2) & Q(
            destin=suspect1))).first()
    if indirect_link is None:
        add_suspect_link(suspect1, suspect2, False)
# Add indirect links for suspects already in db
for link in links:
    suspect2 = Suspect.objects.filter(name=link).first()
    suspect_links = list(SuspectLink.objects\
            .filter((Q(origin=suspect2) | Q(destin=suspect2)) & Q(is_direct=
                True)).all())
    suspect_links.extend(list(SuspectLink.objects\
            .filter((Q(origin=suspect) | Q(destin=suspect)) & Q(is_direct=
                True)).all()))
    # Check all direct links for the suspect and attempt to add indirect
        links
    for sl in suspect_links:
        if sl.origin == suspect and sl.destin == suspect2:
            continue
        if sl.origin == suspect2 and sl.destin == suspect:
            continue
        # Now that we know that this link is not between suspect and
            suspect2
        # check if there is already a link between the current link suspect
        # and the other suspect of the link (the one that is not "suspect")
        if sl.origin == suspect:
            linko = SuspectLink.objects.filter((Q(origin=suspect2) & Q(
                destin=sl.destin)) | (Q(origin=sl.destin) & Q(destin=suspect2
                ))).first()
            if linko is None:
                add_suspect_link(sl.destin, suspect2, False)
        else:
            linko = SuspectLink.objects.filter((Q(origin=suspect2) & Q(
                destin=sl.origin)) | (Q(origin=sl.origin) & Q(destin=suspect2
                ))).first()
            if linko is None:
                add_suspect_link(sl.origin, suspect2, False)
        if sl.origin == suspect2:
            linko = SuspectLink.objects.filter((Q(origin=suspect) & Q(destin
                =sl.destin)) | (Q(origin=sl.destin) & Q(destin=suspect))).
                first()
            if linko is None:
                add_suspect_link(sl.destin, suspect, False)
        else:
            linko = SuspectLink.objects.filter((Q(origin=suspect) & Q(destin
                =sl.origin)) | (Q(origin=sl.origin) & Q(destin=suspect))).
                first()
            if linko is None:
                add_suspect_link(sl.origin, suspect, False)


# Add indicators to the database
for ind_id, ind in suspect_info['indicators'].items():
    indicator = Indicator.objects.filter(id=ind_id).first()
    db_ind = SuspectIndicator.objects.create(
        suspect = suspect,
        indicator = indicator,
    )
    if ind['date']:
        db_ind.date = ind['date']
        db_ind.save()
    if ind['loc']:
        db_ind.location = ind['loc']
        db_ind.save()
score = 0
total_frequency = 0

indicators = SuspectIndicator.objects.filter(suspect=suspect).all()
first_surveillance = timezone.now().date()
most_recent = datetime.datetime(year=1900, month=1, day=1).date()
add_surveillance = False
for sus_indicator in indicators:
    try:
        if sus_indicator.date > most_recent:
            if sus_indicator.location and sus_indicator.date:
                location = sus_indicator.location
                most_recent = sus_indicator.date
        if sus_indicator.indicator.name in FIRST_SURVEILLANCE_INDICATORS:
            first_surveillance = min(sus_indicator.date, first_surveillance)
            add_surveillance = True
    except TypeError:
        pass
    score += sus_indicator.indicator.weight / 100 * sus_indicator.indicator.
        frequency
    total_frequency += sus_indicator.indicator.frequency
if total_frequency > 0:
    score /= total_frequency
else:
    score = 0
suspect.score = round(score * 100, 2)
```

```
    update_suspect_location(suspect, suspect_info['indicators'])

    if add_surveillance:
        suspect.first_surv = first_surveillance
        suspect.second_surv = first_surveillance + relativedelta(months=15)
    suspect.save()

    return api_success(SUSPECT_UPDATE_OK)

def GetSuspectByName(request):
    if request.method != "GET":
        return api_error(INVALID_METHOD)
    name = request.GET.get("name")
    if name is None:
        return api_error("Missing␣suspect␣name")
    suspect = Suspect.objects.filter(name=name).first()
    if suspect is None:
        return api_error("Suspect␣is␣not␣in␣the␣database")

    if suspect.first_surv is not None:
        json_suspect = {
            "id": suspect.id,
            "name": suspect.name,
            "score": suspect.score,
            "location": suspect.location,
            "first_surv": suspect.first_surv.year,
            "second_surv": suspect.second_surv.year,
        }
    else:
        json_suspect = {
            "id": suspect.id,
            "name": suspect.name,
            "score": suspect.score,
            "location": suspect.location,
            "first_surv": "-",
            "second_surv": "-",
        }
    json_resp = {
        "error": False,
        "suspect": json_suspect,
    }
    return JsonResponse(json_resp)

def SuspectList(request):
    if request.method != "GET":
        return api_error(INVALID_METHOD)

    suspects = Suspect.objects.all()

    json_suspects = []
    for x in suspects:
        if x.first_surv is None:
            json_suspects.append([
                f'<a␣href="#"␣onclick="app.reloadSuspect(this)">{x.name}</a>',
                x.score,
                '-',
                '-',
                x.location,
                f'<button␣type="button"␣class="btn␣btn-icon-only␣btn-danger"␣
                    onclick="app.apiDeleteSuspect(\'{x.name}\')"><span␣class="fas
                    ␣fa-times"></span></button>'])
        else:
            json_suspects.append([
                f'<a␣href="#"␣onclick="app.reloadSuspect(this)">{x.name}</a>',
                x.score,
                f'{x.first_surv.year}-{str(x.first_surv.month).zfill(2)}',
                f'{x.second_surv.year}-{str(x.second_surv.month).zfill(2)}',
                x.location,
                f'<button␣type="button"␣class="btn␣btn-icon-only␣btn-danger"␣
                    onclick="app.apiDeleteSuspect(\'{x.name}\')"><span␣class="fas
                    ␣fa-times"></span></button>'])

    json_resp = {
        "error": False,
        "suspectList": json_suspects,
    }
    return JsonResponse(json_resp)

def DeleteSuspect(request):
    if request.method != "POST":
        return api_error(INVALID_METHOD)
    suspect = json.loads(request.body)
    db_suspect = Suspect.objects.filter(name= suspect['name']).first()

    db_suspect.delete()

    return api_success("Successfully␣deleted␣suspect")


def GetNetwork(request):
    if request.method != "GET":
        return api_error(INVALID_METHOD)
    links = SuspectLink.objects.all()
    suspects = Suspect.objects.all()
    nodes = []
    node_idx = {}
    for i, s in enumerate(suspects):
        node = {
            "id": i,
            "name": s.name,
            "symbolSize": 10,
            "value": s.score,
            "category": s.name,
            "label":{"show": True}
        }
        node_idx[s.name] = i
        nodes.append(node)

    edges = []
    for lnk in links:
        source = node_idx[lnk.origin.name]
        target = node_idx[lnk.destin.name]
        if lnk.is_direct:
            nodes[source]["symbolSize"] += 10
            nodes[target]["symbolSize"] += 10
            edges.append({
                "target": target,
                "source": source,
                "lineStyle":{
                    "type":"solid",
                    "color": "rgb(0,␣0,␣0)",
                }
            })
        else:
            edges.append({
                "target": target,
                "source": source,
                "lineStyle":{
                    "type":"dashed",
                    "color": "rgb(0,␣0,␣0)",
                }
            })

    json_resp = {
        "error": False,
        "nodes": nodes,
        "edges": edges,
    }
    return JsonResponse(json_resp)

def clean_script(scr):
    return scr.replace('<script␣type="text/javascript">','').replace('</script>'
        ,'')

def GetSuspectLocation(request):
    if request.method != "GET":
        return api_error(INVALID_METHOD)
    suspect_name = request.GET.get('name')
    if suspect_name is None:
        return api_error(INVALID_SUSPECT)

    suspect = Suspect.objects.filter(name=suspect_name).first()

    if suspect is None:
        return api_error(INVALID_SUSPECT)

    HEIGHT = 600
    WIDTH  = 900

    tile_provider = get_provider(xyz.OpenStreetMap.Mapnik)

    TOOLTIPS = [
        ("ID", "@name"),
        ("LOCATION", "@location"),
        ("COORDS", "(@x,␣@y)"),
    ]
    #p = figure(x_range=(-6.560e5,-6.460e5), y_range=(5.362e6, 5.374e6),
    if suspect.longitude == 0 and suspect.latitude == 0:
        return JsonResponse({
            'error': False,
            'script': "",
            'div': "",
        })
    p = figure(x_range=(suspect.longitude - 0.1e5,suspect.longitude + 0.1e5),
               y_range=(suspect.latitude - 0.05e6, suspect.latitude + 0.05e6),
               height=HEIGHT, width=WIDTH, tooltips=TOOLTIPS, active_scroll="
                   wheel_zoom",
               x_axis_type="mercator", y_axis_type="mercator")
    p.grid.visible = False
    p.add_tile(tile_provider)
    p.circle(
        x='x',
        y='y',
        radius=15000,
        alpha=0.5,
        color="red",
        source=ColumnDataSource(data=dict(
            x=[int(suspect.longitude)],
            y=[int(suspect.latitude)],
            name=[suspect.name],
            location=[suspect.location]
        )),
    )
    script, div = components(p)
    json_resp = {
        'error': False,
        'script': clean_script(script),
        'div':div
    }
    return JsonResponse(json_resp)
```

2 tests.py

```
from django.test import TestCase

# Create your tests here.
```

3 urls.py

```python
from django.urls import path

from . import (
    views,
)

app_name = 'api'

urlpatterns = [
    path('indicators/list', views.IndicatorList),
    path('suspect/update', views.ComputeSuspectProbability),
    path('suspect/info', views.GetSuspectByName),
    path('suspect/list', views.SuspectList),
    path('suspect/delete', views.DeleteSuspect),
    path('suspect/links', views.GetNetwork),
    path('suspect/location', views.GetSuspectLocation),
]
```

4 admin.py

```python
from django.contrib import admin

from .models import (
    Indicator,
    Suspect,
    SuspectIndicator,
    SuspectLink,
)

# Register your models here.
admin.site.register(Indicator)
admin.site.register(Suspect)
admin.site.register(SuspectIndicator)
admin.site.register(SuspectLink)
```

5 error.py

```
from django.http import JsonResponse

def api_error(reason):
    """
    Returns the error response
    """
    resp = {
        'error': True,
        'reason': reason
    }
    return JsonResponse(resp)

def api_success(reason):
    """
    Returns the success response
    """
    resp = {
        'error': False,
        'reason': reason
    }
    return JsonResponse(resp)

# Error messages
INVALID_METHOD = 'Invalid␣method'
INVALID_SUSPECT = 'Invalid␣suspect'
SUSPECT_UPDATE_OK = 'Suspect␣update␣done'
```

6 models.py

```
from django.db import models

class Suspect(models.Model):
    """
    Class that describes a suspect
    """
    name        = models.TextField()
    first_surv  = models.DateField(null=True)
    second_surv = models.DateField(null=True)
    location    = models.TextField(null=True)
    latitude    = models.FloatField(default=0)
    longitude   = models.FloatField(default=0)
    score       = models.FloatField(null=True)
    def __str__(self):
        return self.name

class SuspectLink(models.Model):
    """
    Stores links betwees suspects
    """
    origin = models.ForeignKey(Suspect, related_name="origin", on_delete=models.
        CASCADE)
    destin = models.ForeignKey(Suspect, related_name="destin", on_delete=models.
        CASCADE)
    is_direct = models.BooleanField(default=True)

class Indicator(models.Model):
    """
    The indicators, as defined in the document
    """
    name      = models.TextField()
    weight    = models.IntegerField()
    frequency = models.IntegerField()
    def __str__(self):
        return self.name

class SuspectIndicator(models.Model):
    """
    Indicators per suspect
    """
    indicator = models.ForeignKey(Indicator, on_delete=models.CASCADE)
    suspect = models.ForeignKey(Suspect, on_delete=models.CASCADE)
    date = models.DateField(null=True)
    location = models.TextField(null=True)
```

7 apps.py

```python
from django.apps import AppConfig


class ApiConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'api'
```

Contents

1 urls.py

```
"""core URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('home.urls')),
    path("admin/", admin.site.urls),
    path("", include('theme_pixel.urls'))
]
```

2 settings.py

```
"""
Django settings for core project.

Generated by 'django-admin startproject' using Django 4.1.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/
"""

import os, random, string
from pathlib import Path
from dotenv import load_dotenv
from str2bool import str2bool

load_dotenv()  # take environment variables from .env.

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ.get('SECRET_KEY')
if not SECRET_KEY:
    SECRET_KEY = ''.join(random.choice( string.ascii_lowercase  ) for i in range
        ( 32 ))

# Enable/Disable DEBUG Mode
DEBUG = str2bool(os.environ.get('DEBUG'))
#print(' DEBUG -> ' + str(DEBUG) )

# Docker HOST
ALLOWED_HOSTS = ['*']

# Add here your deployment HOSTS
CSRF_TRUSTED_ORIGINS = ['http://localhost:8000', 'http://localhost:5085', 'http
    ://127.0.0.1:8000', 'http://127.0.0.1:5085']

RENDER_EXTERNAL_HOSTNAME = os.environ.get('RENDER_EXTERNAL_HOSTNAME')
if RENDER_EXTERNAL_HOSTNAME:
    ALLOWED_HOSTS.append(RENDER_EXTERNAL_HOSTNAME)

# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    'theme_pixel',
    "home",
    "api",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "whitenoise.middleware.WhiteNoiseMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "core.urls"

HOME_TEMPLATES = os.path.join(BASE_DIR, 'home', 'templates')

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [HOME_TEMPLATES],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "core.wsgi.application"


# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DB_ENGINE   = os.getenv('DB_ENGINE'   , None)
DB_USERNAME = os.getenv('DB_USERNAME' , None)
DB_PASS     = os.getenv('DB_PASS'     , None)
DB_HOST     = os.getenv('DB_HOST'     , None)
DB_PORT     = os.getenv('DB_PORT'     , None)
DB_NAME     = os.getenv('DB_NAME'     , None)

if DB_ENGINE and DB_NAME and DB_USERNAME:
    DATABASES = {
      'default': {
        'ENGINE'  : 'django.db.backends.' + DB_ENGINE,
        'NAME'    : DB_NAME,
        'USER'    : DB_USERNAME,
        'PASSWORD': DB_PASS,
        'HOST'    : DB_HOST,
        'PORT'    : DB_PORT,
        },
    }
else:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': 'db.sqlite3',
        }
    }

# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME": "django.contrib.auth.password_validation.
            UserAttributeSimilarityValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator"
            ,
    },
    {
        "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator
            ",
    },
    {
        "NAME": "django.contrib.auth.password_validation.
            NumericPasswordValidator",
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = "en-us"

TIME_ZONE = "UTC"

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = '/static/'
#STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

#if not DEBUG:
#     STATICFILES_STORAGE = 'whitenoise.storage.
    CompressedManifestStaticFilesStorage'
```

```
# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

LOGIN_REDIRECT_URL = '/'
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

3 wsgi.py

```
"""
WSGI config for core project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "core.settings")

application = get_wsgi_application()
```

4 asgi.py

```
"""
ASGI config for core project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "core.settings")

application = get_asgi_application()
```