APP.PY:

```python
import flask
from flask import Flask, render_template, request, redirect, jsonify
import joblib
import regex
import sys
import requests
import json
import inputScript
import logging
from forms import ContactForm
from flask_mail import Message, Mail

API_KEY = "9fwquUvd1daYqNf6N0f-0viRwLOjDb-__xY73VKnoT2Q"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey":API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-
type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}


mail = Mail()

app = Flask(__name__)
app.secret_key = '670a9a54ac0304f8ad16324a'

app.config['MAIL_SERVER']='smtp.mailtrap.io'
app.config['MAIL_PORT'] = 2525
app.config['MAIL_USERNAME'] = '5cd9be5c01dfef'
app.config['MAIL_PASSWORD'] = '52d662bc320489'
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USE_SSL'] = False

mail.init_app(app)

app.logger.addHandler(logging.StreamHandler(sys.stdout))
app.logger.setLevel(logging.ERROR)


@app.route('/')
@app.route('/index')
def index():
    return flask.render_template('home.html')

@app.route('/about')
```

```
def about():
    return flask.render_template('about.html')

@app.route('/predict', methods = ['POST'])
def make_prediction():
    url = request.form['url']
    checkprediction = inputScript.main(url)
    payload_scoring = {"input_data": [{"field":
[["having_IPhaving_IP_Address","URLURL_Length","Shortining_Service","having_At
_Symbol","double_slash_redirecting",

"Prefix_Suffix","having_Sub_Domain","SSLfinal_State","Domain_registeration_len
gth","Favicon","port",

"HTTPS_token","Request_URL","URL_of_Anchor","Links_in_tags","SFH","Submitting_
to_email",
        "Abnormal_URL","Redirect","on_mouseover","RightClick",
        "popUpWidnow","Iframe","age_of_domain","DNSRecord","web_traffic
Page_Rank","Google_Index","Links_pointing_to_page","Statistical_report"
    ]], "values": checkprediction }]}
    response_scoring = requests.post('https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/01674dac-1f17-4b13-bb68-
3bf84840f4d0/predictions?version=2022-11-09', json=payload_scoring,
    headers={'Authorization': 'Bearer ' + mltoken})
    print(response_scoring)
    pred = response_scoring.json()
    print(pred)
    prediction = pred['predictions'][0]['values'][0][0]
    print(prediction)
    if prediction==1 :
            label = 'website is not legitimate'
    elif prediction==-1:
            label ='website is legitimate'
    return render_template('home.html', label=label)

@app.route('/contact', methods=['GET', 'POST'])
def contact():
  form = ContactForm()

  if request.method == 'POST':
    if form.validate() == False:
      flash('All fields are required.')
      return render_template('contact.html', form=form)
    else:
      msg = Message(form.subject.data, sender='PHIS_TRAP@example.com',
recipients=['your_email@example.com'])
      msg.body = """
      From: %s &lt;%s&gt;
      %s
      """ % (form.name.data, form.email.data, form.message.data)
      mail.send(msg)

      return redirect("/index",)

  elif request.method == 'GET':
    return render_template('contact.html', form=form)


if __name__ == '__main__':
      app.run(host='0.0.0.0',debug=False)
```

FORMS.PY:

```
from flask_wtf import FlaskForm
from wtforms import TextField,TextAreaField,SubmitField
from wtforms import validators,ValidationError

class ContactForm(FlaskForm):
  name = TextField("Name",  [validators.Required("Please enter your name.")])
  email = TextField("Email",  [validators.Required("Please enter your email
address."), validators.Email("Please enter your email address.")])
  subject = TextField("Subject",  [validators.Required("Please enter a
subject.")])
  message = TextAreaField("Message",  [validators.Required("Please enter a
message.")])
  submit = SubmitField("Send")
```

INPUTSCRIPT.PY:

```
# -*- coding: utf-8 -*-

import regex
from tldextract import extract
import ssl
import socket
from bs4 import BeautifulSoup
import urllib.request
import whois
import datetime


def having_IPhaving_IP_Address(url):
    symbol = 
regex.findall(r'(http((s)?)://)((((\d)+).)*)((\w)+)(/((\w)+))?',url)
    if(len(symbol)!=0):
        having_ip = 1 #phishing
    else:
        having_ip = -1 #legitimate
    return(having_ip)
    return 0


def URLURL_Length(url):
    length=len(url)
    if(length<54):
        return -1
    elif(54<=length<=75):
        return 0
    else:
        return 1


def Shortining_Service(url):
    #ongoing
    return 0

def having_At_Symbol(url):
    symbol=regex.findall(r'@',url)
    if(len(symbol)==0):
        return -1
    else:
        return 1

def double_slash_redirecting(url):
    #ongoing
    return 0

def Prefix_Suffix(url):
    subDomain, domain, suffix = extract(url)
    if(domain.count('-')):
        return 1
    else:
        return -1

def having_Sub_Domain(url):
    subDomain, domain, suffix = extract(url)
    if(subDomain.count('.')==0):
        return -1
    elif(subDomain.count('.')==1):
        return 0
    else:
        return 1

def SSLfinal_State(url):
    try:
#check wheather contains https
        if(regex.search('^https',url)):
            usehttps = 1
        else:
            usehttps = 0
#getting the certificate issuer to later compare with trusted issuer
        #getting host name
        subDomain, domain, suffix = extract(url)
        host_name = domain + "." + suffix
        context = ssl.create_default_context()
        sct = context.wrap_socket(socket.socket(), server_hostname =
host_name)
        sct.connect((host_name, 443))
        certificate = sct.getpeercert()
        issuer = dict(x[0] for x in certificate['issuer'])
        certificate_Auth = str(issuer['commonName'])
        certificate_Auth = certificate_Auth.split()
        if(certificate_Auth[0] == "Network" or certificate_Auth ==
"Deutsche"):
            certificate_Auth = certificate_Auth[0] + " " + certificate_Auth[1]
        else:
            certificate_Auth = certificate_Auth[0]
        trusted_Auth =
['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','V
erizon','Trustwave','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network
Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster'
,'VeriSign']
#getting age of certificate
        startingDate = str(certificate['notBefore'])
        endingDate = str(certificate['notAfter'])
        startingYear = int(startingDate.split()[3])
        endingYear = int(endingDate.split()[3])
        Age_of_certificate = endingYear-startingYear

#checking final conditions
        if((usehttps==1) and (certificate_Auth in trusted_Auth) and
(Age_of_certificate>=1) ):
            return -1 #legitimate
        elif((usehttps==1) and (certificate_Auth not in trusted_Auth)):
            return 0 #suspicious
        else:
            return 1 #phishing

    except Exception as e:
```

```python
        return 1

def Domain_registeration_length(url):
    try:
        w = whois.whois(url)
        updated = w.updated_date
        exp = w.expiration_date
        length = (exp[0]-updated[0]).days
        if(length<=365):
            return 1
        else:
            return -1
    except:
        return 0

def Favicon(url):
    #ongoing
    return 0

def port(url):
    #ongoing
    return 0

def HTTPS_token(url):
    subDomain, domain, suffix = extract(url)
    host =subDomain +'.' + domain + '.' + suffix
    if(host.count('https')): #attacker can trick by putting https in domain
part
        return 1
    else:
        return -1

def Request_URL(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        imgs = soup.findAll('img', src=True)
        total = len(imgs)

        linked_to_same = 0
        avg =0
        for image in imgs:
            subDomain, domain, suffix = extract(image['src'])
            imageDomain = domain
            if(websiteDomain==imageDomain or imageDomain==''):
                linked_to_same = linked_to_same + 1
        vids = soup.findAll('video', src=True)
        total = total + len(vids)

        for video in vids:
            subDomain, domain, suffix = extract(video['src'])
            vidDomain = domain
            if(websiteDomain==vidDomain or vidDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.22):
            return -1
        elif(0.22<=avg<=0.61):
            return 0
        else:
            return 1
    except:
        return 0


def URL_of_Anchor(url):
    try:
        subDomain, domain, suffix = extract(url)
        websiteDomain = domain

        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        anchors = soup.findAll('a', href=True)
        total = len(anchors)
        linked_to_same = 0
        avg = 0
        for anchor in anchors:
            subDomain, domain, suffix = extract(anchor['href'])
            anchorDomain = domain
            if(websiteDomain==anchorDomain or anchorDomain==''):
                linked_to_same = linked_to_same + 1
        linked_outside = total-linked_to_same
        if(total!=0):
            avg = linked_outside/total

        if(avg<0.31):
            return -1
        elif(0.31<=avg<=0.67):
            return 0
        else:
            return 1
    except:
        return 0

def Links_in_tags(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')

        no_of_meta =0
        no_of_link =0
        no_of_script =0
        anchors=0
        avg =0
        for meta in soup.find_all('meta'):
            no_of_meta = no_of_meta+1
        for link in soup.find_all('link'):
            no_of_link = no_of_link +1
        for script in soup.find_all('script'):
            no_of_script = no_of_script+1
        for anchor in soup.find_all('a'):
            anchors = anchors+1
        total = no_of_meta + no_of_link + no_of_script+anchors
        tags = no_of_meta + no_of_link + no_of_script
        if(total!=0):
            avg = tags/total

        if(avg<0.25):
            return -1
        elif(0.25<=avg<=0.81):
            return 0
        else:
            return 1
    except:
        return 0

def SFH(url):
    #ongoing
    return 0

def Submitting_to_email(url):
    try:
        opener = urllib.request.urlopen(url).read()
        soup = BeautifulSoup(opener, 'lxml')
        if(soup.find('mailto:')):
            return 1
        else:
            return -1
```

```python
    except:
        return 0

def Abnormal_URL(url):
    #ongoing
    return 0

def Redirect(url):
    #ongoing
    return 0

def on_mouseover(url):
    #ongoing
    return 0

def RightClick(url):
    #ongoing
    return 0

def popUpWidnow(url):
    #ongoing
    return 0

def Iframe(url):
    #ongoing
    return 0

def age_of_domain(url):
    try:
        w = whois.whois(url)
        start_date = w.creation_date
        current_date = datetime.datetime.now()
        age =(current_date-start_date[0]).days
        if(age>=180):
            return -1
        else:
            return 1
    except Exception as e:
        print(e)
        return 0

def DNSRecord(url):
    #ongoing
    return 0

def web_traffic(url):
    #ongoing
    return 0
```

```
def Page_Rank(url):
    #ongoing
    return 0

def Google_Index(url):
    #ongoing
    return 0


def Links_pointing_to_page(url):
    #ongoing
    return 0

def Statistical_report(url):
    #ongoing
    return 0

def main(url):



    check =
[[having_IPhaving_IP_Address(url),URLURL_Length(url),Shortining_Service(url),h
aving_At_Symbol(url),

 double_slash_redirecting(url),Prefix_Suffix(url),having_Sub_Domain(url),SSLfi
nal_State(url),

Domain_registeration_length(url),Favicon(url),port(url),HTTPS_token(url),Reque
st_URL(url),

URL_of_Anchor(url),Links_in_tags(url),SFH(url),Submitting_to_email(url),Abnorm
al_URL(url),

Redirect(url),on_mouseover(url),RightClick(url),popUpWidnow(url),Iframe(url),

age_of_domain(url),DNSRecord(url),web_traffic(url),Page_Rank(url),Google_Index
(url),
              Links_pointing_to_page(url),Statistical_report(url)]]



    return check
```

LAYOUT.HTML:

```
<!DOCTYPE html>
```

```html
<html>
  <head>

    <title>PHIS TRAP</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="This website is develop for identify the
safety of url.">
    <meta name="keywords" content="phishing url,phishing,cyber
security,machine learning,classifier,python">
    <!-- BootStrap -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css
"
        integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
    <link rel="stylesheet" href="{{ url_for('static',filename='cover.css') }}"
/>

  </head>
  <body>
    <div class="site-wrapper">

      <div class="site-wrapper-inner">

        <div class="cover-container">

          <div class="masthead clearfix">
            <div class="inner">
              <h3 class="masthead-brand">PHIS TRAP</h3>
              <nav>
                <ul class="nav masthead-nav">
                  <li ><a href="/">Home</a></li>
                  <li><a href="/about">About</a></li>
                  <li><a href="/contact">Contact</a></li>
                </ul>
              </nav>
            </div>
          </div>

          {% block body %} {% endblock %}
            <br>
          {% if label %} <p>{{label}}</p> {% endif %}
          <br>
          <br>
          <br>
          <br>
          <br>
          <br>
          </div>

        </div>

      </div>

    </div>










    </script>

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></scri
pt>
    <script>window.jQuery || document.write('<script
src="https://code.jquery.com/jquery-1.12.4.min.js"><\/script>')</script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></sc
ript>

    <script language="javascript" type="text/javascript">
     $(window).load(function() {
     $('#loading').hide();
     });

     $(".btn").click(function () {
     $("#loading").show();
     });
    </script>




  </body>
</html>
```

HOME.HTML:

```
{% extends 'layout.html' %}

{% block body %}




    <div class="inner cover">

        <h1 class="cover-heading" >Predict legitimacy of websites</h1>
        <div id="loading" style="height:25px">
        <img   src="/static/200w (1).webp" class="center" height="50"
width="50"  />
        <br>
        <h4>Please wait this can take a while</h4>
        </div>
        <br>
        <br>
        <form action ='/predict' method='post'>
           <input class="form-control" type = 'text' name = 'url'>
            <br>
           <input class="btn btn-lg btn-default slide_left" type="submit"
value="Predict">
        </form>
    </div>
{% endblock %}
```

CONTACT.HTML:

```
{% extends "layout.html" %}

{% block body %}
  <h2>Contact us</h2>

  {% for message in form.name.errors %}
    <div class="flash">{{ message }}</div>
  {% endfor %}

  {% for message in form.email.errors %}
    <div class="flash">{{ message }}</div>
  {% endfor %}

  {% for message in form.subject.errors %}
    <div class="flash">{{ message }}</div>
  {% endfor %}

  {% for message in form.message.errors %}
```

```
    <div class="flash">{{ message }}</div>
  {% endfor %}

  <form action="{{ url_for('contact') }}" method=post>
    {{ form.hidden_tag() }}

    {{ form.name.label }}
    {{ form.name }}

    {{ form.email.label }}
    {{ form.email }}

    {{ form.subject.label }}
    {{ form.subject }}

    {{ form.message.label }}
    {{ form.message }}

    {{ form.submit }}
  </form>
  <br>
  {% if label %} <p>{{label}}</p> {% endif %}
{% endblock %}
```

ABOUT.HTML:

```
{% extends 'layout.html' %}

{% block body %}
    <div class="inner cover">
        <p class="lead">
            Detection of phishing websites is a really important safety
measure for most of the online platforms. So, as to save a platform with
malicious requests from such websites, it is important to have a robust
phishing detection system in place.
        </p>
        <p class="lead">
            PHIS TRAP is an intelligent, flexible and effective system that
uses UCI datasets and supervised machine learning algorithms to classify
legitimacy of websites. The main objective of this system is to distinguish
the phishing websites from the legitimate websites and ensure secure
transactions to users.
        </p>
    </div>
{% endblock %}
```

COVER.CSS:

```css
/*
 * Globals
 */

/* Links */
a,
a:focus,
a:hover {
  color: #fff;
}

/* Custom default button */

.btn-default,
.btn-default:focus {
  color: rgb(0, 0, 0);
  position:center;
  border: 2px solid rgb(30, 216, 2);
  border-radius: 0px;
  padding: 18px 36px;
  display: inline-block;
  font-family: "Lucida Console", Monaco, monospace;
  font-size: 14px;
  letter-spacing: 1px;
  cursor: pointer;
  box-shadow: inset 0 0 0 0 #02d826;
  -webkit-transition: ease-out 0.4s;
  -moz-transition: ease-out 0.4s;
  transition: ease-out 0.4s;
}
.slide_left:hover {
  box-shadow: inset 0 0 0 50px #0fff02;
}



/*
 * Base structure
 */

html,
body {
height: 100%;
background: rgb(10,0,36);
background: linear-gradient(198deg, rgba(10,0,36,1) 0%, rgba(9,58,121,1) 48%,
rgba(0,254,255,1) 100%);
}
```

```css
/* Extra markup and styles for table-esque vertical and horizontal centering
*/
.site-wrapper {
  display: table;
  width: 100%;
  height: 100%; /* For at least Firefox */
  min-height: 100%;
  -webkit-box-shadow: inset 0 0 100px rgba(0,0,0,.5);
          box-shadow: inset 0 0 100px rgba(0,0,0,.5);
}
.site-wrapper-inner {
  display: table-cell;
  vertical-align: top;
}
.cover-container {
  margin-right: auto;
  margin-left: auto;
}

/* Padding for spacing */
.inner {
  padding: 30px;
}


/*
 * Header
 */
.masthead-brand {
  margin-top: 10px;
  margin-bottom: 10px;
}

.masthead-nav > li {
  display: inline-block;
}
.masthead-nav > li + li {
  margin-left: 20px;
}
.masthead-nav > li > a {
  padding-right: 0;
  padding-left: 0;
  font-size: 16px;
  font-weight: bold;
  color: #fff; /* IE8 proofing */
  color: rgba(255,255,255,.75);
  border-bottom: 2px solid transparent;
}
.masthead-nav > li > a:hover,
.masthead-nav > li > a:focus {
  background-color: transparent;
  border-bottom-color: #a9a9a9;
  border-bottom-color: rgba(255,255,255,.25);
}
.masthead-nav > .active > a,
.masthead-nav > .active > a:hover,
.masthead-nav > .active > a:focus {
  color: #fff;
  border-bottom-color: #fff;
}

@media (min-width: 768px) {
  .masthead-brand {
    float: left;
  }
  .masthead-nav {
    float: right;
  }
}


/*
 * Cover
 */

.cover {
  padding: 0 20px;
}
.cover .btn-lg {
  padding: 10px 20px;
  font-weight: bold;
}


/*
 * Footer
 */

.mastfoot {
  color: #999; /* IE8 proofing */
  color: rgba(255,255,255,.5);
}


/*
 * Affix and center
 */

@media (min-width: 768px) {
  /* Pull out the header and footer */
  .masthead {
    position: fixed;
    top: 0;
  }
  .mastfoot {
    position: fixed;
    bottom: 0;
  }
  /* Start the vertical centering */
  .site-wrapper-inner {
    vertical-align: middle;
  }
  /* Handle the widths */
  .masthead,
  .mastfoot,
  .cover-container {
    width: 100%; /* Must be percentage or pixels for horizontal alignment */
  }
}

@media (min-width: 992px) {
  .masthead,
  .mastfoot,
  .cover-container {
    width: 700px;
  }
}
h1 { color: #ffffff;
  font-family: 'Raleway',sans-serif;
  font-size: 22px; font-weight:
  800; line-height: 32px;
  margin: 0 0 14px;
  text-align: center;
  text-transform: uppercase;
}
h3{
  text-align: center;
  font-family:'Shojumaru','display';
  font-size: 38px;
  font-weight: 100;
  color: rgba(245, 8, 8, 1);
  text-transform: none;
  font-style: italic;
  text-decoration: underline overline;
  line-height: 2em;
  letter-spacing: 0px;
  text-shadow: 0px 00px 10px rgba(0, 0, 0, 1);
}
h4{ color: #0b0b0b;
  font-family: 'Raleway',sans-serif;
  font-size: 12px; font-weight:
  800; line-height: 62px;
  margin: 0 0 24px;
  text-align: center;
  text-transform: uppercase;
}
p{ color: #f5f5f5;
  font-family: 'Raleway',sans-serif;
  font-size: 22px;
  font-weight: 800;
  line-height: 12px;
  margin: 0 0 4px;
  text-align: center;
  text-transform: uppercase;
}
.center {
  display: block;
  margin-left: auto;
  margin-right: auto;
  width: 10%;
}
/* Contact form */
form label {
  font-size: 1.2em;
  font-weight: bold;
  display: block;
  padding: 10px 0;
}

form input#name,
form input#email,
form input#subject {
  width: 400px;
  background-color: #fafafa;
  -webkit-border-radius: 3px;
     -moz-border-radius: 3px;
          border-radius: 3px;
  border: 1px solid #cccccc;
  padding: 5px;
  font-size: 1.1em;
}
```

```
form textarea#message {
  width: 500px;
  height: 100px;
  background-color: #fafafa;
  -webkit-border-radius: 3px;
     -moz-border-radius: 3px;
          border-radius: 3px;
  border: 1px solid #cccccc;
  margin-bottom: 10px;
  padding: 5px;
  font-size: 1.1em;
}

form input#submit {
  display: block;
  -webkit-border-radius: 3px;
     -moz-border-radius: 3px;
          border-radius: 3px;
  border:1px solid #d8d8d8;
  padding: 10px;
  font-weight:bold;
  text-align: center;
  color: #000000;
  background-color: #f4f4f4;
  background-image: -webkit-gradient(linear, left top, left bottom, color-
stop(0%, #f4f4f4), color-stop(100%, #e5e5e5));
  background-image: -webkit-linear-gradient(top, #f4f4f4, #e5e5e5);
  background-image: -moz-linear-gradient(top, #f4f4f4, #e5e5e5);
  background-image: -ms-linear-gradient(top, #f4f4f4, #e5e5e5);
  background-image: -o-linear-gradient(top, #f4f4f4, #e5e5e5);
  background-image: linear-gradient(top, #f4f4f4,
#e5e5e5);filter:progid:DXImageTransform.Microsoft.gradient(GradientType=0,star
tColorstr=#f4f4f4, endColorstr=#e5e5e5);
}

form input#submit:hover{
  cursor: pointer;
  border:1px solid #c1c1c1;
  background-color: #dbdbdb;
  background-image: -webkit-gradient(linear, left top, left bottom, color-
stop(0%,#dbdbdb), color-stop(100%, #cccccc));
  background-image: -webkit-linear-gradient(top, #dbdbdb, #cccccc);
  background-image: -moz-linear-gradient(top, #dbdbdb, #cccccc);
  background-image: -ms-linear-gradient(top, #dbdbdb, #cccccc);
  background-image: -o-linear-gradient(top, #dbdbdb, #cccccc);
  background-image: linear-gradient(top, #dbdbdb,
#cccccc);filter:progid:DXImageTransform.Microsoft.gradient(GradientType=0,star
tColorstr=#dbdbdb, endColorstr=#cccccc);
```

```css
}
/* Message flashing */
.flash {
  background-color: #FBB0B0;
  padding: 10px;
  width: 400px;
}
h2{
  font-family: 'Zen Dots', 'display';
  text-align: center;
  font-size: 36px;
  font-weight: 100;
  color: rgba(0, 0, 0, 1);
  text-transform: none;
  font-style: normal;
  text-decoration: none;
  line-height: 2em;
  letter-spacing: 0px;
}
```