OSTL - MINI PROJECT REPORT

1.Title – URL SHORTENER

Roll no. – 02-Jatin Advani 15-Deepanshu Bakhru

2.Program statement – A program to shorten the given url.

3.Explanation-

- **JINJA** –It is a template engine for Python. It is similar to the Django template engine.
- A template engine or template processor is a library designed to combine templates with a data model to produce documents**.**
- Python template engine used to create HTML or other markup formats that are returned to the user via an HTTP response.
- The Jinja template engine allows customization of tags, filters, tests, and globals. Also, unlike the Django template engine, Jinja allows the template designer to call functions with arguments on objects.
- **Some of the features of Jinja are:**
- template inheritance
- compiles down to the optimal Python code just-in-time
- optional ahead-of-time template compilation
- easy to debug
- configurable syntax

- **HTML** -HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language.

- HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages.

- HTML is a markup language which is used by the browser to manipulate text, images and other content to display it in required format.

- HTML uses predefined tags and elements which tells the browser about content display property.

- HTML page can be created using any text editor (notepad). Then save that file using .html extension and open that file in browser. It will get the HTML page response.

- **CSS** - CSS stands for Cascading Style Sheet. Cascading Style Sheet is used to set the style in web pages which contain HTML elements. It sets the background color, font-size, font-family, color,etc property of elements in a web pages.
- There are three types of CSS which are given below:
- Inline CSS
- Internal or Embedded CSS
- External CSS
- Inline CSS: Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using style attribute.
- Internal or Embedded CSS: This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.
- External CSS: External CSS contains separate CSS file which contains only style property with the help of tag attributes. CSS property written in a separate file with .css extension and should be linked to the HTML document using link tag. This means that for each element, style can be set only once and that will be applied across web pages

- **DJANGO** - Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks.
- It's very easy to switch database in Django framework.
- It has built-in admin interface which makes easy to work with it.
- Django is fully functional framework that requires nothing else.
- It has thousands of additional packages available.
- It is very scalable.
- Features of Django-
- **Versatility of Django**
  Django can build almost any type of website. It can also work with any client-side framework and can deliver content in any format such as HTML, JSON, XML etc.
- **Security**
  Since Django framework is made for making web development easy, it has been engineered in such a way that it automatically do the right things to protect the website.
- **Portability**
  All the codes of the Django framework are written in Python, which runs on many platforms. Which leads to run Django too in many platforms such as Linux, Windows and Mac OS.

- **REQUESTS** -Requests library is one of the integral part of Python for making HTTP requests to a specified URL.
- When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response.

File - D:\miniproject\manage.py

```
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks
   ."""
3  import os
4  import sys
5
6
7  def main():
8      os.environ.setdefault('DJANGO_SETTINGS_MODULE', '
   miniproject.settings')
9      try:
10         from django.core.management import
   execute_from_command_line
11     except ImportError as exc:
12         raise ImportError(
13             "Couldn't import Django. Are you sure it's
   installed and "
14             "available on your PYTHONPATH environment
   variable? Did you "
15             "forget to activate a virtual environment?"
16         ) from exc
17     execute_from_command_line(sys.argv)
18
19
20 if __name__ == '__main__':
21     main()
22
```

File - D:\miniproject\Run Miniproject.bat

```
1 @echo off
2 python manage.py runserver
3 start localhost:8000
```

File - D:\miniproject\scripts\gen.py

```python
from scripts.main import ParentValues
import random as rand
import requests
import pickle


class Generator(ParentValues):
    def generateKey(self):
        positions = {}

        if self.total_count < self.maximum_entries:

            num_pos = rand.randint(0, 5)
            if not num_pos == 0:
                positions[num_pos] = rand.randint(0, 9)

            for i in range(1, 6):
                caps = False
                if i == num_pos:
                    continue
                if rand.randint(0, 1) == 1:
                    caps = True
                if caps:
                    positions[i] = chr(rand.randint(65, 90
))
                else:
                    positions[i] = chr(rand.randint(97, 122
))

            tlink = ''
            # print(f'{tlink} and {positions}')
            for i in range(1, 6):
                tlink = tlink + str(positions[i])
            if not self.keyPresent(tlink):
                fLink = self.host_name + tlink
            else:
                self.generateKey()

            self.total_count += 1
            self.ageMapper[self.total_count] = fLink

        else:

            fLink = self.ageMapper[self.oldest]
            # self.oldest+=1
            if self.oldest == self.maximum_entries:
```

```
45                  self.oldest = 1
46              else:
47                  self.oldest += 1
48
49          return fLink
50
51      def keyPresent(self, key):
52          f = self.linkMapper.get(key)
53          if f == None:
54              return False
55          else:
56              return True
57
58      def validateURL(self, user_link):
59
60          # Part where https:// is appended to the start of
    the URL string
61          if user_link[:4] == 'http':
62              if user_link[4:7] == '://':
63                  user_link = 'https://' + user_link[7:]
64
65              elif user_link[4:8] == 's://':
66                  pass
67              else:
68                  return('Invalid URL Syntax')
69
70          else:
71              user_link = 'https://' + user_link
72
73          #     try:
74          #         if user_link == 'https://':
75          #             raise InvalidURLError
76          #     except:
77          #         print('No URL entered')
78
79          emptyLink = False
80          # print(user_link)
81          try:
82
83              if user_link == 'https://':
84                  emptyLink = True
85              try:
86                  if emptyLink:
87                      raise Exception('No URL Entered')
88              except:
89                  return('No URL Entered')
```

```
 90
 91
 92             else:
 93                 validator = requests.get(user_link)
 94
 95                 # print(f'{validator}')
 96                 # validator=requests.get(user_link)
 97                 f = True
 98                 for i in range(400, 452):
 99                     if i == validator.status_code:
100                         return(
101                             f"Server Response Code: {
    validator.status_code}\nThis link can't be parsed due to a
     client error")
102                         f = False
103                         break
104                 for i in range(500, 512):
105                     if i == validator.status_code:
106                         return(
107                             f"Server Response Code: {
    validator.status_code}\nThis link can't be parsed due to a
     server error")
108                         f = False
109                         break
110                 if f:
111                     return self.addKey(user_link)
112         except requests.ConnectionError:
113             return('The site provided does not exist')
114         # return 'lmao'
115
116     def addKey(self, user_link):
117         file = self.generateKey()
118         dat = file.split('/a?i=')
119         dat = str(dat[1])
120         self.linkMapper[dat] = user_link
121         self.writeValues()
122         print(self.linkMapper)
123         print(self.ageMapper)
124         return file
125
126
127     def writeValues(self):
128         a = {'total_count': self.total_count}
129         b = {'oldest': self.oldest}
130         with open('D:\\oldest.pickle', 'wb') as fp:
131             pickle.dump(b, fp, protocol=pickle.
```

```
131 HIGHEST_PROTOCOL)
132         with open('D:\\total.pickle', 'wb') as fp:
133             pickle.dump(a, fp, protocol=pickle.
    HIGHEST_PROTOCOL)
134         with open('D:\\linkMapper.pickle', 'wb') as fp:
135             pickle.dump(self.linkMapper, fp, protocol=
    pickle.HIGHEST_PROTOCOL)
136         with open('D:\\ageMapper.pickle', 'wb') as fp:
137             pickle.dump(self.ageMapper, fp, protocol=
    pickle.HIGHEST_PROTOCOL)
138
```

File - D:\miniproject\scripts\code.py

```python
from scripts.gen import Generator


g=Generator()
g.validateURL('google.com')
print(g.linkMapper)
print(g.ageMapper)

#NOT USED
```

File - D:\miniproject\scripts\main.py

```python
import random as rand
import requests
import csv
import pickle


class ParentValues:
    host_name = 'localhost:8000/a?i='
    # test=10
    try:
        with open('D:\\linkMapper.pickle', 'rb') as fp:
            linkMapper = pickle.load(fp)
    except FileNotFoundError:
        linkMapper = {}

    try:
        with open('D:\\ageMapper.pickle', 'rb') as fp:
            ageMapper = pickle.load(fp)
    except FileNotFoundError:
        ageMapper = {}

    try:
        with open('D:\\total.pickle', 'rb') as fp:
            total_count = pickle.load(fp)
            total_count = int(total_count['total_count'])
    except FileNotFoundError:
        total_count = 0

    try:
        with open('D:\\oldest.pickle', 'rb') as fp:
            oldest = pickle.load(fp)
            oldest = int(oldest['oldest'])
    except FileNotFoundError:
        oldest = 1

    maximum_entries = 52 * 52 * 52 * 52 * 10

```

```python
import pickle

from scripts.main import ParentValues
from scripts.gen import Generator


class Retrieve(ParentValues):
    def __init__(self, user_link):
        self.user_link = user_link

    def searchLink(self):
        # print(ParentValues.test)
        # with open('D:\linkMapper.pickle', 'rb') as fp:
        #     print(pickle.load(fp))
        try:
            m=self.linkMapper[self.user_link]
            return m
        except Exception as efgr:
            print(efgr)
            exit(1)
            # return 'bhai ye kya dale ho tum'

    def lol(self):
        return self.linkMapper

```

```css
.centrewrapper{
    text-align: center;
}

.ralign{
    text-align: right;
    font: normal 80%/normal Roboto, Helvetica, sans-serif;
}

.heading{
    font: normal 500%/normal "American Purpose Casual 01", Helvetica, sans-serif;
}

input[type=text]{
    margin-bottom: 2vw;
    font: normal 120%/normal Calibri, Helvetica, sans-serif;
    border: 2px solid red;
    text-align: center;
    border-radius: 3vw;
    padding: 10px 10px;
    width: 20%;
    transition: width 0.5s ease-in-out;
}

input[type=text]:focus{
    outline: none;
    width: 60%;
    background-color: ;
}

.short{
    background-color: white;
    border: 2px solid black;
    -webkit-border-radius: 3.125em;
    border-radius: 3.125em;
    font: normal 150%/normal Calibri, Helvetica, sans-serif;

    text-align: center;
    -o-text-overflow: clip;
    text-overflow: clip;
    padding: 0.25em 1px;
    width: 25%;
    text-align: center;
```

```
44 }
45
46 .short:hover{
47     background-color: lightpink;
48     box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px
    0 rgba(0,0,0,0.19);
49 }
50
51 .short:active{
52     background-color: lightcoral;
53   transform: translateY(4px);
54 }
55
```

File - D:\miniproject\website\html\index.html

```
<!--{% load static %}-->
<html>
    <head>
        <meta name="viewport" content="initial-scale=1, width=device-width, height=device-height viewport-fit=cover ">
        <meta name="author" content="Deepanshu Bakhru, Jatin Advani">
        <meta name="application-name" content="URL Shortener">
        <meta name="description" content="Shortens long URLs for easy and convenient sharing">
        <link rel="stylesheet" type="text/css" href="../website/css/styling.css">
        <title>URL Shortener</title>
    </head>
    <body style="background-image: url(../website/graphics/69.png)">
        <div class=centrewrapper>
            <p class=ralign>
                Python Project by<br>
                Deepanshu Bakhru<br>
                Jatin Advani<br>
            </p>
            <p class="heading">
                URL SHORTENER
            </p>
            <form name="app" method="POST" action="upload">
                {% csrf_token %}
                <input type="text" name="entry" placeholder="Enter link to shorten" value="{{input}}">
                <br>
                <button class="short" type="submit" name="submit" >SHORTEN</button>
                <p name="output" >
                    <a href="{{output}}" target="_blank" rel="noopener noreferrer">Your link has been shortened</a>
<!--                    anchor this to the whatever-->
                </p>
            </form>
        </div>
    </body>
</html>
```

File - D:\miniproject\shortener\apps.py

```python
from django.apps import AppConfig


class ShortenerConfig(AppConfig):
    name = 'shortener'

```

File - D:\miniproject\shortener\urls.py

```python
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home),
    path('upload', views.upload),
    path('a', views.a) #this calls views.py me def a(). this is of the form localhost:8000/a?id=xyz
]

```

File - D:\miniproject\shortener\admin.py

```python
from django.contrib import admin

# Register your models here.

```

File - D:\miniproject\shortener\tests.py

```python
from django.test import TestCase

# Create your tests here.

```

File - D:\miniproject\shortener\views.py

```python
from django.http import HttpResponse
from django.shortcuts import render, redirect
from scripts.main import ParentValues
from scripts.gen import Generator
from scripts.retrieval import Retrieve

# Create your views here.
output = ''


def home(request):
    return render(request, 'html/index.html', {'input': ''
, 'output': output})


def upload(request):
    urlobject = Generator()
    output = urlobject.validateURL(request.POST['entry'])
    # print(output + 'lol')
    return render(request, 'html/index.html', {'input': ''
, 'output': output})


def a(request):
    part = request.GET['i']  # i is key associated to value
. e.g. localhost:8000/a?i=ooo
    print(part)
    # gotta append https:// to link
    r = Retrieve(part)
    return redirect(r.searchLink())
    # call retrieval for 'part' variable

```

File - D:\miniproject\shortener\models.py

```python
from django.db import models

# Create your models here.

```

```python
"""
ASGI config for miniproject 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/3.0/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'miniproject.settings')

application = get_asgi_application()

```

```python
"""miniproject URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.0/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 path, include


urlpatterns = [
    path('', include('shortener.urls')),
    path('admin/', admin.site.urls),
]

```

File - D:\miniproject\miniproject\wsgi.py

```python
"""
WSGI config for miniproject 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/3.0/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'miniproject.settings')

application = get_wsgi_application()

```

File - D:\miniproject\miniproject\settings.py

```
"""
Django settings for miniproject project.

Generated by 'django-admin startproject' using Django 3.0.4.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'hj!wsc8d988m#)i9$6@5=v^sm*m(0$!&mw2oz=#s^tnz&83a39'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'shortener',
```

```
39 ]
40
41 MIDDLEWARE = [
42     'django.middleware.security.SecurityMiddleware',
43     'django.contrib.sessions.middleware.SessionMiddleware',
44     'django.middleware.common.CommonMiddleware',
45     'django.middleware.csrf.CsrfViewMiddleware',
46     'django.contrib.auth.middleware.
   AuthenticationMiddleware',
47     'django.contrib.messages.middleware.MessageMiddleware',
48     'django.middleware.clickjacking.XFrameOptionsMiddleware
   ',
49 ]
50
51 ROOT_URLCONF = 'miniproject.urls'
52
53 TEMPLATES = [
54     {
55         'BACKEND': 'django.template.backends.django.
   DjangoTemplates',
56         'DIRS': [os.path.join(BASE_DIR, 'website')],
57         'APP_DIRS': True,
58         'OPTIONS': {
59             'context_processors': [
60                 'django.template.context_processors.debug',
61                 'django.template.context_processors.request
   ',
62                 'django.contrib.auth.context_processors.
   auth',
63                 'django.contrib.messages.context_processors
   .messages',
64             ],
65         },
66     },
67 ]
68
69 WSGI_APPLICATION = 'miniproject.wsgi.application'
70
71 # Database
72 # https://docs.djangoproject.com/en/3.0/ref/settings/#
   databases
73
74 DATABASES = {
75     'default': {
76         'ENGINE': 'django.db.backends.sqlite3',
77         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
```

```
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.0/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/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/website/'

STATICFILES_DIRS = [
```

```
118     os.path.join(BASE_DIR, "website"),
119 ]
```