**Company\_website with Django**

Create the root directory.

Create the project root directory as a folder on the desktop, name is django\_practice2, right click and open with vscode or open the root directory with the following command inside terminal: mkdir django\_practice2

Create the virtual environment.

Cd into django\_practice2 directory and create the virtual environment.

Create a virtual environment using the venv module with the command python -m venv .venv or pip install virtualenv

The name of the virtual environment directory is .venv and the period that precedes the virtual environment directory indicates that the directory will be hidden.

Activate the virtual environment

Inside the django\_practice2 directory use the following command to activate the virtual environment: .venv/scripts/activate

Or using the gitbash terminal use the following command: source .venv/scripts/activate

Install Django

With the virtual environment activated and at the root of django\_practice2 directory, use the following command to install Django package: pip install django

Start a new project

With the virtual environment activated and at the root of django\_practice2 directory, use the following command to start a new project and name the new project django\_project: django-admin startproject company\_website\_project .

The period after the company\_website\_project name is intentional; this will make Django to place all the files inside one single directory.

Apply migrations

When Django created the company\_website\_project directory, several built-in apps were created along .

We need to apply migrations so that these changes can be applied to the database.

With the virtual environment activated, and at the root of django\_practice2 directory, use the following command to apply migration: python manage.py migrate

Start the development server

With the virtual environment activated and at the root of django\_practice2 directory, use the following command to start the development server: python manage.py runserver

Stop the development server with the following command: control + c

Starting an app.

With the virtual environment activated and at the root of django\_practice2, use the following command to start an app named pages: python manage.py startapp pages2.

Add the pages2 app to the list of installed apps inside the company\_website\_project directory settings

Create a function-based view + class based view

Open the view.py file of pages2 app and write the following codes

from django.http import HttpResponse

from django.shortcuts import render

from django.views.generic import TemplateView

def home\_page\_view(request):

context = {

"inventory\_list": ["Widget 1", "Widget 2", "Widget 3"],

"greeting": "THAnk you FOR visitING.",

}

return render(request, "home.html", context)

**class AboutPageView**(TemplateView):

template\_name = "about.html"

**def** get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context["contact\_address"] = "123 Main Street"

context["phone\_number"] = "555-555-5555"

**return** context

Project-level templates

Create a single templates directory inside the company\_website\_project directory

Open the setting.py file of company\_website\_project directory and add the following codes to the TEMPLATES

TEMPLATES = [

{

...

"DIRS": [BASE\_DIR / "templates"], # new

...

},

]

Create & Update the home.html file

Create a new file called “home.html” within the templates directory.

Add the following code to the home.html file:

{% extends "base.html" %}

{% block content %}

<**h1**>Company Homepage</**h1**>

<**p**>The current date and time is: {% now "DATETIME\_FORMAT" %}</**p**>

<**p**>There are {{ inventory\_list|length }} items of inventory.

<**ul**>

{% for item in inventory\_list %}

<**li**>{{ item }}</**li**>

{% endfor %}

</**ul**>

<**p**>{{ greeting| title }}</**p**>

{% comment %}Add more content here!{% endcomment %}

{% endblock %}

Create & Update the about.html file

To render context variables in our template, we use double brackets, {{ }}.

Create a new file called “about.html” within the templates directory.

Add the following code to the home.html file:

{% extends "base.html" %}

{% block content %}

<**h1**>Company About Page</**h1**>

<**p**>The company address is {{ contact\_address }} and the phone number is

{{ phone\_number }}.</**p**>

{% endblock %}

Create and update the base.html fie

Within the templates directory, create a base.html file containing a header with links to our home page and about pages. This is our parent template, which all other child templates will inherit from

<**header**>

<**a** href="{% url 'home' %}">Home</**a**>

<**a** href="{% url 'about' %}">About</**a**>

</**header**>

{% block content %}{% endblock %}

Create a urls.py file inside the pages2 app

we need to match a url path to this view function.

create a urls.py file inside the pages2 app and write the following code

from django.urls import path

from .views import home\_page\_view, AboutPageView

urlpatterns = [

path( “”, home\_page\_view, name="home" ),

path(“about/”, AboutPageView.as\_view(), name="about")

]

update the urls.py (gateway to other urls) file inside the root directory of company\_website\_project

There is an built-in urls.py file inside the root directory company\_website\_project

Open the urls.py file and import the following code:

From django.urls import path, include

Add the pages2 app url path to the list of the url inside the urls.py file of company\_website\_project directory

The urls.py file has an inbuilt admin url path

urlpatterns = [

path(“admin/”, admin.site.urls),

path(“”, include(“pages2.urls”),

path(“about/”, include(“pages2.urls”),

]

Start the development server with python manage.py runserver.

This will display hello world

.GITIGNORE FILE

At the root of django\_practice2, create a file and name it .gitignore.

This will be used to store the virtual environment, to prevent it from been pushed to github.

Open the .gitignore file and add .venv

\_\_pycache\_\_/

db.sqlite3

testing

Since no database is involved in our project, we will import SimpleTestCase at the top of the file. For our first tests, we’ll check that the two URLs for our website, the Homepage and About page, return HTTP 200 status codes, the standard response for a successful HTTP request.

With virtual environment activated and at the root of django\_practice2

Open the test.py file of the pages2 app and add the following codes.

from django.test import SimpleTestCase

**from django.urls import** reverse

class HomepageTests(SimpleTestCase):

def test\_url\_exists\_at\_correct\_location(self):

response = self.client.get("/")

self.assertEqual(response.status\_code, 200)

**def** test\_url\_available\_by\_name(self):

response = self.client.get(reverse("home"))

self.assertEqual(response.status\_code, 200)

**def** test\_template\_name\_correct(self):

response = self.client.get(reverse("home"))

self.assertTemplateUsed(response, "home.html")

**def** test\_template\_content(self):

response = self.client.get(reverse("home"))

self.assertContains(response, "<h1>Company Homepage</h1>")

class AboutpageTests(SimpleTestCase):

def test\_url\_exists\_at\_correct\_location(self):

response = self.client.get("/about/")

self.assertEqual(response.status\_code, 200)

**def** test\_url\_available\_by\_name(self):

response = self.client.get(reverse("about"))

self.assertEqual(response.status\_code, 200)

def test\_template\_name\_correct(self):

response = self.client.get(reverse("about"))

self.assertTemplateUsed(response, "about.html")

**def** test\_template\_content(self):

response = self.client.get(reverse("about"))

self.assertContains(response, "<h1>Company About Page</h1>")

Stop the development server with ctrl + c

Use the following command to test python manage.py test

Storing the installed packages of the virtual environment

Record of installed packages inside virtual environment

With virtual environment activated at the root of django\_practice2, use the following command to store the packages inside a requiremens.txt file.

pip freeze> requirement.txt

pushing to github account

stop the development server with ctrl + c and at the root of django\_practice2 directory, deactivate the virtual environment with the following command: deactivate

go to github account and create a repository named django\_pracitce2

At the root of django\_practice2 directory, use the following command to initialize git and push the code to github

git init

git add -A

git commit -m “first commit”

git branch -M main

git remote add origin https://github.com/trevor4ivan/django\_practice2.git

git push -u origin main