

Internship Presentation

2

Name: Arbaaz Khan

Roll No.: 17

Intern @ Elite Software

Learnings in the Month of February

► Python:

Learned python language starting with the basic concepts such as Variables, Data types, Operators, Python data structure to more advanced topics such as Classes and Function, GUI Programming.

► Django:

Django by learning the basics of web development, including HTML, CSS, and JavaScript. From there, I gradually worked my way up to Django, starting with the fundamental concepts such as URLs, views, templates, and models. I gained a deeper understanding of the Model-View-Controller (MVC) architecture and how it is implemented in Django.

Some Important Django Commands

- ▶ Creating a new Django project: **django-admin startproject project_name**
- ▶ Creating a new Django application within a project: **python manage.py startapp app_name**
- ▶ Starting the Django development server: **python manage.py runserver**
- ▶ Creating database tables for the application: **python manage.py migrate**
- ▶ Creating a superuser for the Django admin: **python manage.py createsuperuser**
- ▶ Running unit tests for the application: **python manage.py test app_name**
- ▶ Creating a new migration for changes to models: **python manage.py makemigrations**
- ▶ Applying migrations to the database: **python manage.py migrate**
- ▶ Displaying a list of available commands: **python manage.py help**

Files that Django Automatically Generates for Faster development Process:

- ▶ When we Create a Django Project it will create python files for faster development
- ▶ Some Files are-
- ▶ Settings.py : configuration file that stores all the settings and configuration variables for a Django project
- ▶ Models.py: models.py is a Python module that defines the structure of the database tables for a Django application
- ▶ Views.py:It is responsible for processing incoming HTTP requests, interacting with models to retrieve and modify data, and returning HTTP responses.
- ▶ Urls.py:defines the URL patterns or routes for a web application. It maps the requested URL to the appropriate view function, which will generate the HTTP response.
- ▶ Manage.py: This is the command-line utility for interacting with your project. You can use it to run development servers, create database tables, and execute other management tasks.

Some tasks Related to Python and Django done in the month of February

Task: To Generate QR code for any given link in python

Implementation:

The Program Implemented using Python Tkinter library for the user interface and the QR code and other libraries for generating and saving the QR codes.

When the user enters the title and URL and clicks the "Enter" button, the program generates a QR code image and saves it to a file with the given title. The "View QR CODE" button displays the generated QR code in a GUI window.

```
3_QR_Code_Generator.py - E:\ARBAAZ KHAN\Elite\Presentation\Presentation 2\3_QR_Code_Generator.py (3.8.5)
File Edit Format Run Options Window Help
from tkinter import *
from PIL import ImageTk, Image
import qrcode
import os

window= Tk()
window.title('QR Code Generator')

def my_click():
    canvas.delete("all")
    my_text=e2.get()
    img = qrcode.make(my_text)
    img.save(os.getcwd()+ '/Images'+e.get()+'.jpeg')

    my_button2 = Button(UI_frame, text='View QRCODE', command=myqrviewer)
    my_button2.grid(row=4, column=1, padx=30, pady=20)

def myqrviewer():
    global my_img1
    my_img1 = ImageTk.PhotoImage(Image.open(os.getcwd()+ '/Images'+e.get()+'.jpeg'))

    canvas.create_image(100,50, anchor=NW, image=my_img1)
    window.update_idletasks()

UI_frame=Frame(window, width=500, height=500)
UI_frame.grid(row=0, column=0)

l1=Label(UI_frame, text="Enter Title:")
l1.grid(row=0, column=0)

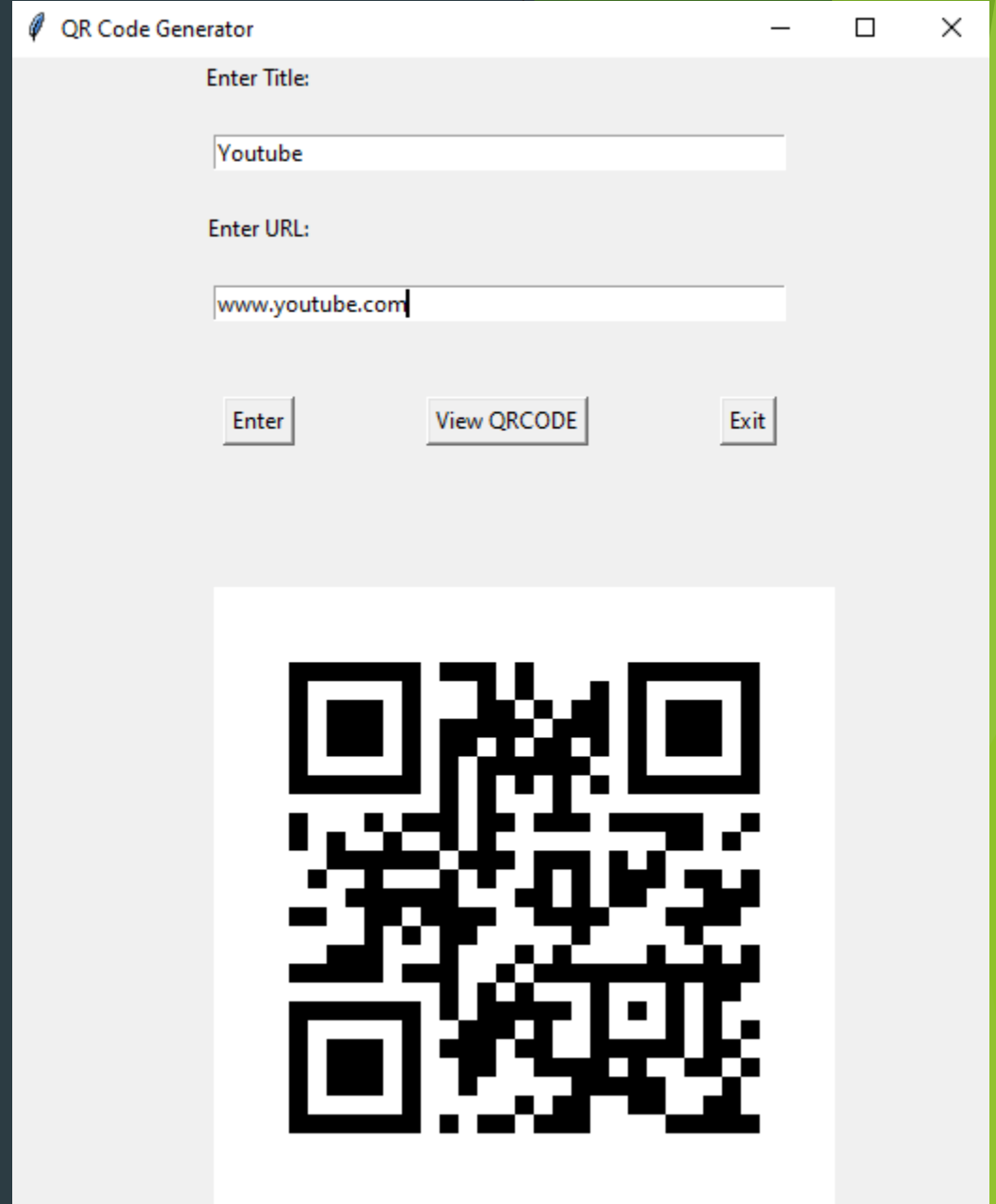
e=Entry(UI_frame, width=50)
e.grid(row=1, column=0, columnspan=3, padx=30, pady=20)

l2=Label(UI_frame, text="Enter URL:")
l2.grid(row=2, column=0)

e2=Entry(UI_frame, width=50)
e2.grid(row=3, column=0, columnspan=3, padx=30, pady=20)

b1= Button(UI_frame, text="Enter", command=my_click)
b1.grid(row=4, column=0, padx=30, pady=20)
```

Ln: 20 Cok: 62



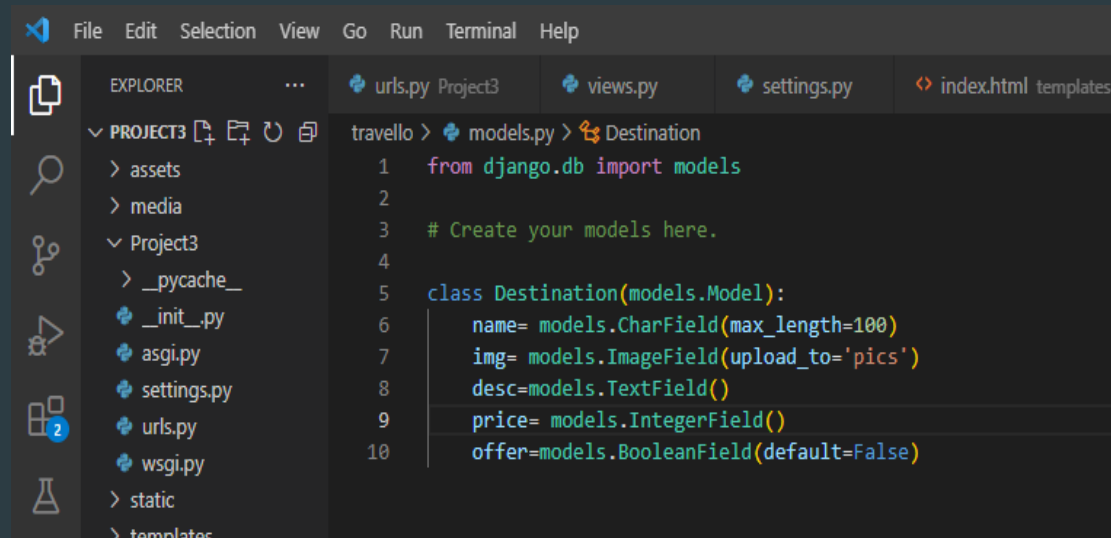
Python-Django Task

- ▶ **Task: To Change the Content of Static webpage (Destination Part) to Dynamic webpage using Django.**
- ▶ **Implementation:**

Steps Followed to accomplish the given task:

1. Creating Database Schema in Models.py
2. Enable Database in Settings.py
3. Mapping Urls to include app urls
4. Creating Views and passing Database object as parameter
5. Making changes in our html file (changing static content to dynamic)
6. Adding Content by using Django Admin Panel
7. Verifying in postgresQL using pgAdmin.

Models.py



```
File Edit Selection View Go Run Terminal Help
EXPLORER
PROJECT3
  assets
  media
  Project3
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  static
  templates

travello > models.py > Destination
1 from django.db import models
2
3 # Create your models here.
4
5 class Destination(models.Model):
6     name= models.CharField(max_length=100)
7     img= models.ImageField(upload_to='pics')
8     desc=models.TextField()
9     price= models.IntegerField()
10    offer=models.BooleanField(default=False)
```

Settings.py

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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'project3',
        'USER': 'postgres',
        'PASSWORD': 'admin1',
        'HOST': 'localhost'
    }
}
```


urls.py

```
15
16 from django.contrib import admin
17 from django.urls import path,include
18 from django.conf import settings
19 from django.conf.urls.static import static
20
21 urlpatterns = [
22     path('',include('travello.urls')),
23     path('admin/', admin.site.urls),
24 ]
25
26 urlpatterns+=static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
27
```

Views.py

```
Go Run Terminal Help views.py -
urls.py Project3 views.py X settings.py index.html templates Release Notes: 1
travello > views.py > index
1 from django.shortcuts import render
2 from .models import Destination
3
4 # Create your views here.
5 def index(request):
6     dests=Destination.objects.all()
7     return render(request,'index.html', {'dests':dests})
```

- Making changes in Index.html file.

```
267 <!-- Destinations -->
268
269 <div class="destinations" id="destinations">
270     <div class="container">
271         <div class="row">
272             <div class="col text-center">
273                 <div class="section_subtitle">simply amazing places</div>
274                 <div class="section_title">
275                     <h2>Popular Destinations</h2>
276                 </div>
277             </div>
278         </div>
279         <div class="row destinations_row">
280             <div class="col">
281                 <div class="destinations_container item_grid">
282
283                     {% for dest in dests %}
284                     <!-- Destination -->
285                     <div class="destination item">
286                         <div class="destination_image">
287                             
288                             <div class="spec_offer text-center"><a href="#">Special Offer</a></div>
289                         </div>
290                         <div class="destination_content">
291                             <div class="destination_title"><a href="destinations.html">{{dest.name}}</a>
292                             </div>
293                             <div class="destination_subtitle">
294                                 <p>{{dest.desc}}</p>
295                             </div>
296                             <div class="destination_price">From ${{dest.price}}</div>
297                         </div>
298                     </div>
299
300                     {% endfor %}
301
302                 </div>
303             </div>
304         </div>
305     </div>
306 </div>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: python

PS E:\ARBAAZ KHAN\Django\Project3> python .\manage.py runserver
Watching for file changes with StatReloader

- ▶ This is Django Adminstration panel from here we can add details for our destination like Name, Image, Description, Price, and whether it is in offer or not

Django administration

WELCOME, **ARBAAZ** [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) › [Travello](#) › [Destinations](#) › Add destination

Add destination

Name:

Img:

Choose File

Pune.jpg

Desc:

Pune: Where heritage meets modernity.

Price:

☒ Offer

SAVE

Save and add another

Save and continue editing

- The data which we will enter in Django Administration panel will be added in postgresSQL

pgAdmin 4

File Object Tools Help

Browser

- > Casts
- > Catalogs
- ✓ Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- ✓ Schemas (1)
 - ✓ public
 - > Aggregates
 - > Collations
 - > Domains
 - > FTS Configurations
 - > FTS Dictionaries
 - > Aa FTS Parsers
 - > FTS Templates
 - > Foreign Tables
 - > Functions
 - > Materialized Views
 - > Operators
 - > Procedures
 - > 1.3 Sequences
 - ✓ Tables (11)
 - > auth_group
 - > auth_group_permission
 - > auth_permission
 - > auth_user
 - > auth_user_groups
 - > auth_user_user_permissions
 - > django_admin_log

Dashboard Properties SQL Statistics Dependencies Dependents Processes public.travello_... public.travello_...

public.travello_destination/project3/postgres@PostgreSQL 15

Query Query History

```
1 SELECT * FROM public.travello_destination
2 ORDER BY id ASC
```

Data Output Messages Notifications

	id [PK] bigint	name character varying (100)	img character varying (100)	desc text	offer boolean	price integer
1	1	Mumbai	pics/Mumbai.jpg	The city that never sleeps: Discov...	false	700
2	2	Pune	pics/Pune.jpg	Pune: Where heritage meets mod...	true	600
3	3	Hyderabad	pics/hyderabad.jpg	Experience the royal charm of Hy...	true	650
4	4	Jammu & Kashmir	pics/jammu.jpg	A Paradise on earth	true	700

Finally when we run this Command
>Python manage.py runserver

Our Project will be running on localhost
and now our content for destination part
is dynamic and responsive.

