# Voting System Project *Using* DJANGO Framework

# Project Title: pollster (Voting System) web application using DJANGO framework.

# Introduction: We will create a pollster (voting system) web application using DJANGO. This application will conduct a series of questions along with many choices. A user will be allowed to give voting for that question by selecting a choice. We will also build the admin part of this project.

# Pre-requisite: Knowledge of python and basics of DJANGO Framework. Python should be installed in the system. Visual studio code or any code editor to work on the application.

# Technologies used in the project: DJANGO framework and SQL database which comes by default with DJANGO.

# Implementation of the Project:

# Creating Project:

# Step-1: Create an empty folder pollster \_project in your directory.

# Step-2: Now switch to your folder and create a virtual environment in this folder using the following commend.

# Pip install pip env

# Pip env shell

# Step-3: A Pip file will be created in your folder from the above step. Now install DJANGO in your folder using the following commend.

# PIPENV install DJANGO

# Step-4: Now we need to establish the DJANGO project. Run the following command in your folder and initiate a DJANGO project.

# DJANGO-admin start project pollster

# A new Folder with name pollster will be created. Switch to the pollster folder using the following command.

# CD pollster

# The folder structure will look something like this

# Check if the application running or not using your http://127.0.0.1:8000/ in your browser.

# Step-5: Create an app ‘polls’ using the following command

# Python manage.py start app polls

# polls-app

# 

# Create Models:

# Step-1: In your models.py file write the code given below to create two tables in your database. One is ‘Question’ and the other one is ‘Choice’’. ‘Question’ will have two fields of ‘question \_text’ and a ‘pub \_date’. Choice has three fields: ‘question’, ‘choice \_text’, and ‘vote from DJANGO. DB import models

# # Create your models here.

# class Question(models .Model):

# question \_text = models .Char Field (max \_length = 200)

# pub \_date = models .Date Time Field('date published')

# def \_\_ str \_\_(self):

# return self .question \_text

# from DJANGO .db import models

# # Create your models here.

# Class Question(models .Model):

# Question \_text = models .Char Field (max \_length = 200)

# Pub \_date = model . Date Time Field ('date published')

# Def \_\_ str \_\_(self):

# Return self .question \_text

# Class Choice (models.Model):

# question = models .Foreign Key (Question, on \_delete = models. CASCADE)

# choice \_text = models .Char Field(max length = 200)

# votes = models. Integer Field (default = 0)

# def \_\_str \_\_(self):

# return self. Choice \_text

# class Choice(models .Model):

# question = models. Foreign Key (Question, on \_delete = models. CASCADE)

# choice \_text = models. Char Field (max \_length = 200)

# votes = models. (default = 0)

# def \_\_str \_\_(self):

# 

# Step-2: Go to the settings.py file and in the list, INSTALLED\_APPS write down the code below to include the app in our project. This will refer to the polls apps.py-Polls class.

# INSTALLED\_APPS = [

# 'polls . Apps .Polls Config',

# 'django . contrib. . admin',

# 'django. Contrib. .auth',

# 'django .contrib .content types',

# 'django. contrib. static files’

# 'django. contrib. messages',

# 'django. Contrib. static files’,

# Create an Admin User:

# Step-1: Run the command given below to create a user who can login to the admin site.

# Python manage.py create super user

# User name: geeks123

# Now it will prompt an email address which again we need to enter here.

# Email Address : vmf@example.com

# Final step is to enter the password

# Password: \*\*\*\*\*\*

# Password (again):\*\*\*\*\*\*

# Super user created successfullyadmin-panel

# Step-2: In the admin.py file we will write the code given below to map each question with choices to select. form Django .contrib .import admin

# # Register your models here.

# from .models import Question, Choice

# # admin. site register(Question)

# # admin.site register(Choice)

# admin.site.site\_header = "Pollster Admin"

# admin.site.site\_title = "Pollster Admin Area"

# admin.site.index\_title = "Welcome to the Pollster Admin Area"

# class Choice In Line (admin. Tabular Inline):

# model = Choice

# extra = 3

# class QuestionAdmin(admin.ModelAdmin):

# fieldsets = [(None, {'fields': ['question\_text']}), ('Date Information', {

# 'fields': ['pub\_date'], 'classes': ['collapse']}), ]

# inlines = [ChoiceInLine]

# admin.site.register(Question,QuestionAdmin)

# Create Views:

# question-choices

# Step-1: Open views.py file and write down the code given below.

# from django.template import loader

# from django.http import HttpResponse, HttpResponseRedirect

# from django.shortcuts import get\_object\_or\_404, render

# from django.urls import reverse

# from .models import Question, Choice

# # Get questions and display them

# def index(request):

# latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

# context = {'latest\_question\_list': latest\_question\_list}

# return render(request, 'polls / index.html', context)

# # Show specific question and choices

# def detail(request, question\_id):

# try:

# question = Question.objects.get(pk = question\_id)

# except Question.DoesNotExist:

# raise Http404("Question does not exist")

# return render(request, 'polls / detail.html', {'question': question})

# # Get question and display results

# def results(request, question\_id):

# question = get\_object\_or\_404(Question, pk = question\_id)

# return render(request, 'polls / results.html', {'question': question})

# # Vote for a question choice

# def vote(request, question\_id):

# # print(request.POST['choice'])

# question = get\_object\_or\_404(Question, pk = question\_id)

# try:

# selected\_choice = question.choice\_set.get(pk = request.POST['choice'])

# except (KeyError, Choice.DoesNotExist):

# # Redisplay the question voting form.

# return render(request, 'polls / detail.html', {

# 'question': question,

# 'error\_message': "You didn't select a choice.",

# })

# else:

# selected\_choice.votes += 1

# selected\_choice.save()

# # Always return an HttpResponseRedirect after successfully dealing

# # with POST data. This prevents data from being posted twice if a

# # user hits the Back button.

# return HttpResponseRedirect(reverse('polls:results', args =(question.id, )))

# Step-2: Create a file urls.py inside the pollster-polls folder to define the routing for all the methods we have implemented in views.py file. from django.urls import path

# from . import views

# app\_name = 'polls'

# urlpatterns = [

# path('', views.index, name ='index'),

# path('<int:question\_id>/', views.detail, name ='detail'),

# path('<int:question\_id>/results/', views.results, name ='results'),

# path('<int:question\_id>/vote/', views.vote, name ='vote'),

# ]

# Create Template:

# Step-1: Follow the steps given below to create the front layout of the page.

# Create a folder ‘templates’ in toplevel pollster folder.

# Create ‘base.html’ file inside the template folder.

# Create ‘base.html’ file inside the template folder.

# Step-2: Open settings.py file and add the code given below in the list ‘TEMPLATES’.

# Lightbox

# TEMPLATES = [

# {

# # make changes in DIRS[].

# 'BACKEND': 'django.template.backends.django.DjangoTemplates',

# 'DIRS': [os.path.join(BASE\_DIR, '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',

# ],

# },

# },

# ]

# Step-3: Open index.html file and write the code given below. This file will display the list of guestions which are stored in our database.

# {% extends 'base.html' %}

# {% block content %}

# <h1 class="text-center mb-3">Poll Questions</h1>

# {% if latest\_question\_list %}

# {% for question in latest\_question\_list %}

# <div class="card-mb-3">

# <div class="card-body">

# <p class="lead">{{ question.question\_text }}</p>

# <a href="{% url 'polls:detail' question.id %}" class="btn btn-primary btn-sm">Vote Now</a>

# <a href="{% url 'polls:results' question.id %}" class="btn btn-secondary btn-sm">Results</a>

# </div>

# </div>

# {% endfor %}

# {% else %}

# <p>No polls available</p>

# {% endif %}

# {% endblock %}

# Step-4: Open detaile.html file and write the code given below.

# {% extends 'base.html' %}

# {% block content %}

# <a class="btn btn-secondary btn-sm mb-3" href="{% url 'polls:index' %}">Back To Polls</a>

# <h1 class="text-center mb-3">{{ question.question\_text }}</h1>

# {% if error\_message %}

# <p class="alert alert-danger">

# <strong>{{ error\_message }}</strong>

# </p>

# {% endif %}

# <form action="{% url 'polls:vote' question.id %}" method="post">

# {% csrf\_token %}

# {% for choice in question.choice\_set.all %}

# <div class="form-check">

# <input type="radio" name="choice" class="form-check-input" id="choice{{ forloop.counter }}"

# value="{{ choice.id }}" />

# <label for="choice{{ forloop.counter }}">{{ choice.choice\_text }}</label>

# </div>

# {% endfor %}

# <input type="submit" value="Vote" class="btn btn-success btn-lg btn-block mt-4" />

# </form>

# {% endblock %}

# Step-5: Open the result. html file and write the code given below. {% extends 'base.html' %}

# {% block content %}

# <h1 class="mb-5 text-center">{{ question.question\_text }}</h1>

# <ul class="list-group mb-5">

# {% for choice in question.choice\_set.all %}

# <li class="list-group-item">

# {{ choice.choice\_text }} <span class="badge badge-success float-right">{{ choice.votes }}

# vote{{ choice.votes | pluralize }}</span>

# </li>

# {% endfor %}

# </ul>

# <a class="btn btn-secondary" href="{% url 'polls:index' %}">Back To Polls</a>

# <a class="btn btn-dark" href="{% url 'polls:detail' question.id %}">Vote again?</a>

# {% endblock %}

# Step-6: Lets create a navigation bar for our application.

# <nav class="navbar navbar-dark bg-primary mb-4">

# <div class="container">

# <a class="navbar-brand" href="/">Pollster</a>

# </div>

# </nav>

# Step-7: We can write this code in just one single file in base.html

# <!DOCTYPE html>

# <html lang="en">

# <head>

# <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

# integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

# <title>Pollster {% block title %}{% endblock %}</title>

# </head>

# <body>

# <!--NavBar-->

# {% include 'partials/\_navbar.html'%}

# <div class="container">

# <div class="row">

# <div class=".col-md-6 m-auto">

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

# </div>

# </div>

# </div>

# </body>

# Create Landing Page:

# The URL <http://127.0.0.1:8000/> should display a landing page for our web application.

# Step-1: Switch to the top-level pollster folder and run the command given below.

# Python manage.py start app pages

# Lightbox

# Step-2: Open ‘views.py’ inside ‘pages’ folder.

# from django.shortcuts import render

# # Create your views here.

# def index(request):

# return render(request, 'pages / index.html')

# Step-3: Create urls.py file inside the ‘pages’ folder.

# from django.urls import path

# from . import views

# urlpatterns = [

# path('', views.index, name ='index'),

# ]

# Step-4: Create a folder ‘pages’ inside the ‘template’ folder.

# {% extends 'base.html' %}

# {% block content %}

# <div class="card text-center">

# <div class="card-body">

# <h1>Welcome To Pollster!</h1>

# <p>This is an Polling Web Application built with Django</p>

# <a class="btn btn-dark" href="{% url 'polls:index' %}">

# View Available Polls</a>

# </div>

# </div>

# {% endblock %}

# Testing of the Application:

# Admin Frontend:

# Step-1: Run the server using the command python manage.py run server .Now the username and password to login into the system.

# admin-login

# Step-2: Click on ‘add’ button next to the ‘Questions’.

# Lightbox

# User Frontend:

# Step-1:This is an example Djando polling app landing-page

# Step-2: You will see list of questions with two options ‘vote now’ and ‘results’.

# poll-questions

# Step-3: Once this is done select any one choice and click on ‘Vote’ button.

# You can also check the total votes for any question using the option ‘Results’ from the ‘poll Questions’ page.

# Lightbox

# Conclusion: A voting application built with the django framework could be dynamic platform for conducting polls, surveys, or elections online. Django built in security measures and admin flexibility make it well suited for ensuring the integrity and reliability of such applications.

# 