Voting System Project Using DJANGO Framework

# Implementation of the project:

## Step 1: Create a New Project

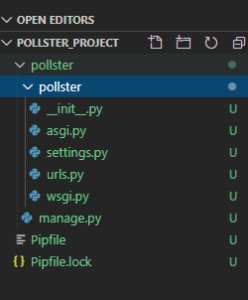
To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language. Before moving further let’s add some color attributes in order to enhance the app bar. Go to app > res > values > colors.xml and add the following color attributes.

## Step 2: Adding Resources

In this step, we will add image resources to our application. These images are placed in the drawable folder(app > res > drawable). You can use any images in place of these. Refer to this article: How to Add Image to Drawable Folder in Android Studio

## step 3:Working with the activity

In this step, we will create the layout for our voting application to compaobjects. For this, go to app > res > layout > activity\_main.xml and add the following code snippet



# Create Models:

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

private TextView txtFirst;

private TextView txtSecond;

private Button btnFirst;

private Button btnSecond;

private int scoreFirst = 0;

private int scoreSecond = 0

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// initializing textview

txtFirst = (TextView) findViewById(R.id.txtFirst);

txtSecond = (TextView) findViewById(R.id.txtSecond);

// initializing button view

btnFirst = (Button) findViewById(R.id.btnFirst);

btnSecond = (Button) findViewById(R.id.btnSecond);

// setting initial value to text view

txtFirst.setText(String.valueOf(0));

txtSecond.setText(String.valueOf(0));

// updating textview on button click

btnFirst.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

scoreFirst++;

txtFirst.setText(String.valueOf(scoreFirst));

}

});

// updating textview on button click

btnSecond.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

scoreSecond++;

txtSecond.setText(String.valueOf(scoreSecond));

}

});

}

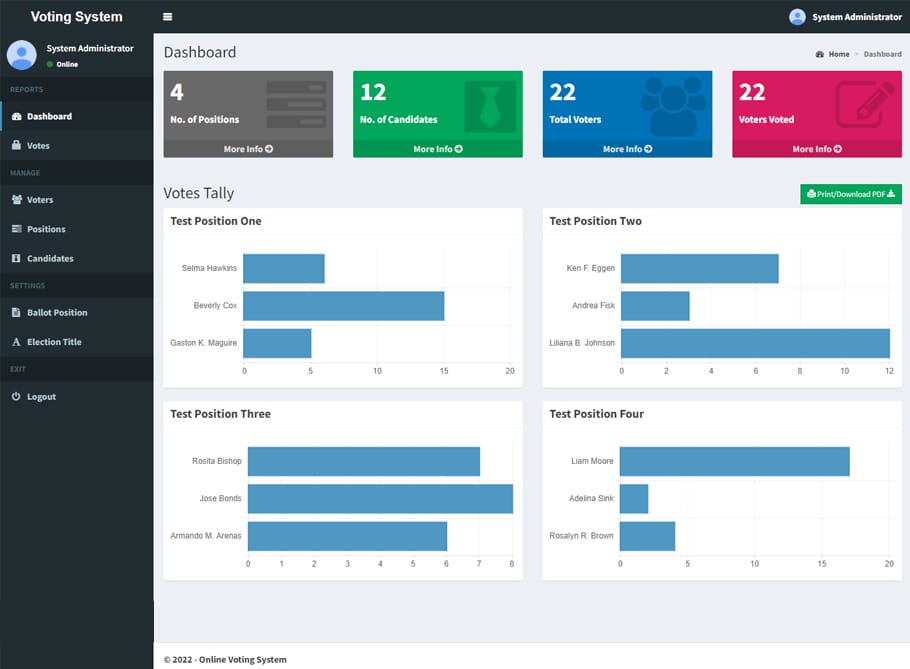
}

## Step 4: Working with the MainActivity.java file

In MainActivity.java, we will initialize our Text and Button views and add onClickListener() to the button which is used to update the values in the TextView. Comments are added inside the code to understand the code in more detail.

# About Online Voting System Django Project:

In particular, this e-voting system project in Python Django focuses mainly on dealing with online voting, and voter-candidate information. Also, the system displays selective overall data using graphical representations. In addition, the system allows managing voters’ records. Evidently, this project is divided into two categories: Voter, and Admin Panel. In an overview of this web application, a voter can simply register into the system. Initially, the system only allows a voter to cast vote and view his/her ballot. In fact, the system restricts users after casting a vote. This means, that one user can only cast vote once, but he/she can select up to 20 candidates. Besides, a voter can only list out his/her ballot which displays the name of voted candidates.



# Available Features:

1 .Voter’s Panel

2. Admin Panel

3. Cast Votes

4. Select Multiple Candidates

5. View Personal Ballot

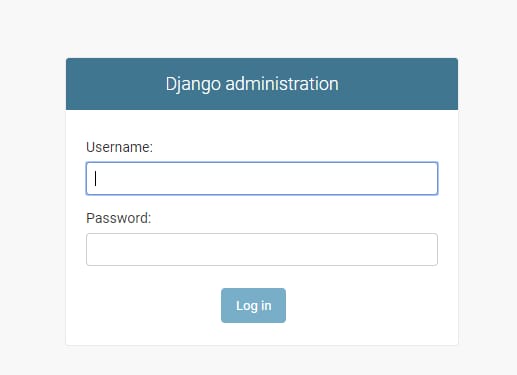
6. Voter Management

7. Candidate Management

8. Set up Positions

9. List Vote Details

10 .Votes Tally



# Steps to Create a “ Python Django Voting System” Project – Let’s Move ahead with this amazing project.

1. Open the terminal fromfolder where we want to create the project. Right click on mouse -> open in terminal -> write “code .” (space is mandatory between code and “.”)

2. Then go to the project folder -> urls.py and inside urls.py of project, do this -> add “include” in import as shown in the code below and add “path(“”, include(“app.urls”))”

3. Create urls.py in the app of Pin your Notes project(urls.py is mandatory in both the project and app).

4. In setting.py, add the ‘app name”.

5. Now runserver to check if everything is working or not. If you see the below image, then we are done with the installation part. To runserver, run command in terminal as follows “py manage.py runserver”.

6. Now, create the models.py using the ORM technique as follows.

# Templates.html -> login.html in django:

<div class="container">

<h1><u>DataFlair Voting System</u></h1>

<div class="form-container">

<h1>Login</h1>

<form action="{% url 'login' %}" method="post">

{% csrf\_token %}

<input type="text" placeholder="Username" name="uname">

<input type="password" placeholder="Password" name="password">

<button type="submit">Login</button>

</form>

<p>Not registered? <a href="{% url 'register' %}">click here</a></p>

</div>

</div>

# Set Up The Project:

Let's get started by installing Django and starting up our project. I already have a folder created with a virtual environment ready to go.

First we need to install Django and then we need to create a Django project.

pip install django

django-admin startproject poll\_project

Now that our project has been created, let's create an app for our poll. We can call this app: poll.

cd poll\_project

python manage.py startapp poll

We can then open up the poll\_project/settings.py file and add poll to the list of apps.

# settings.py

## INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

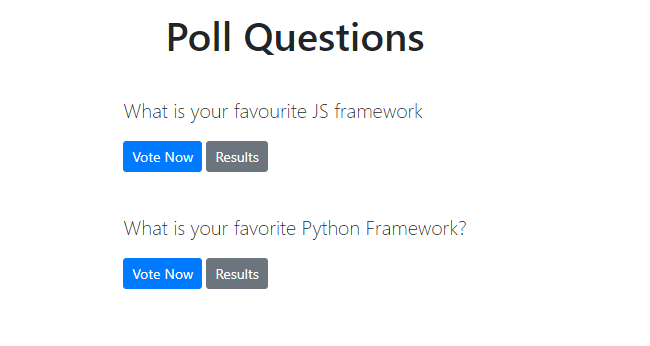
'django.contrib.messages',

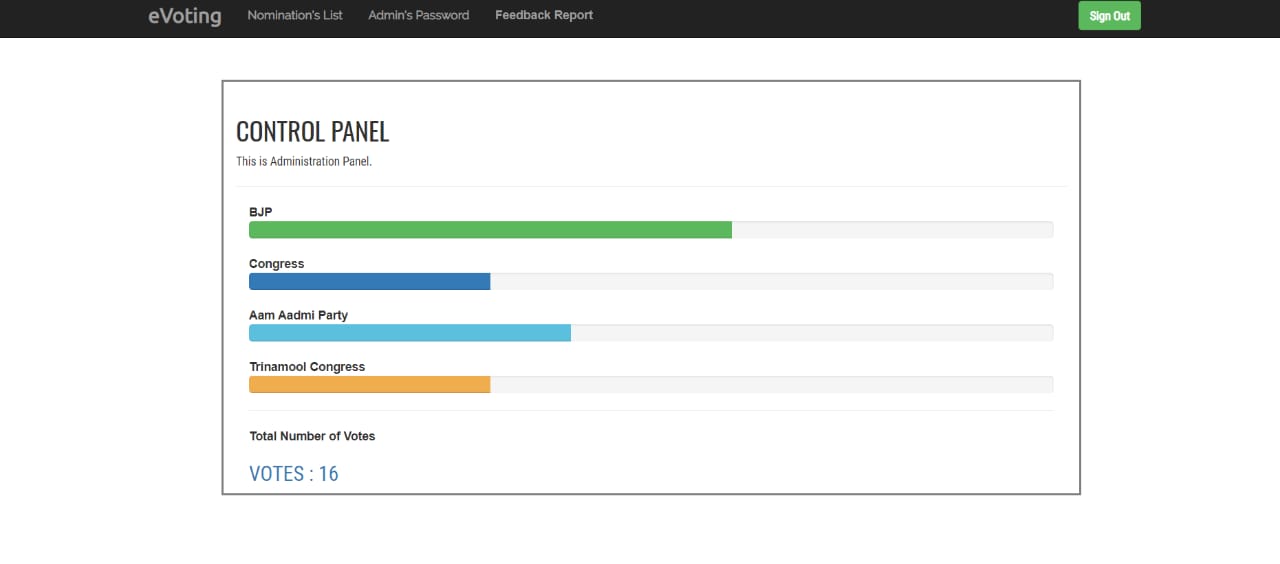
'django.contrib.staticfiles',

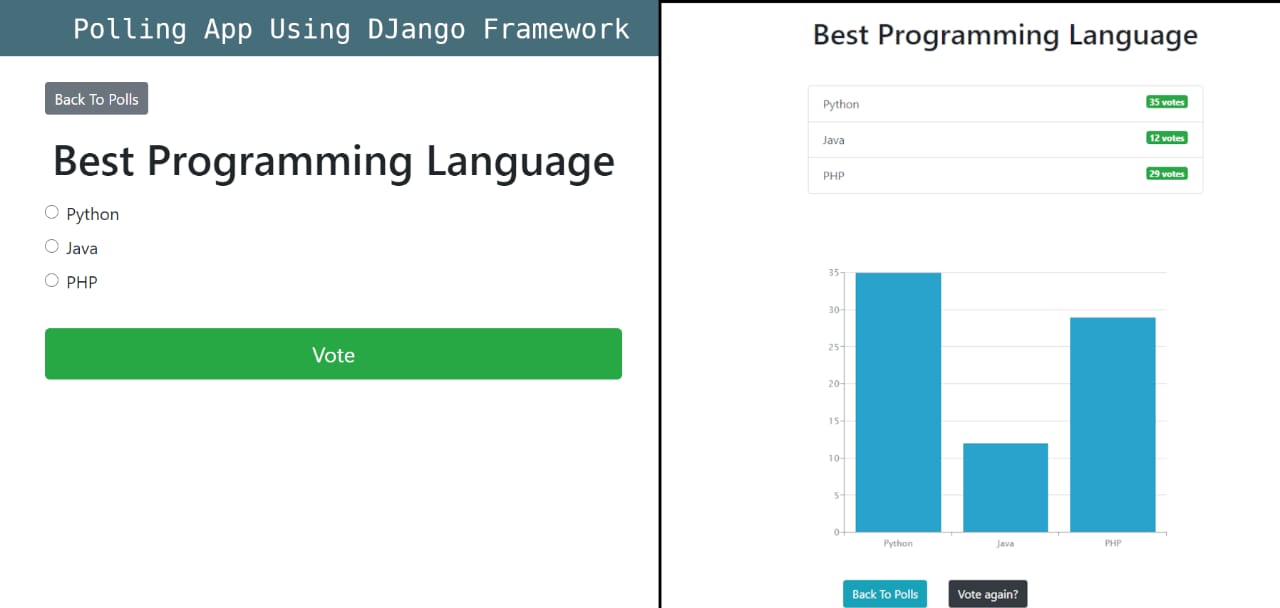
poll',

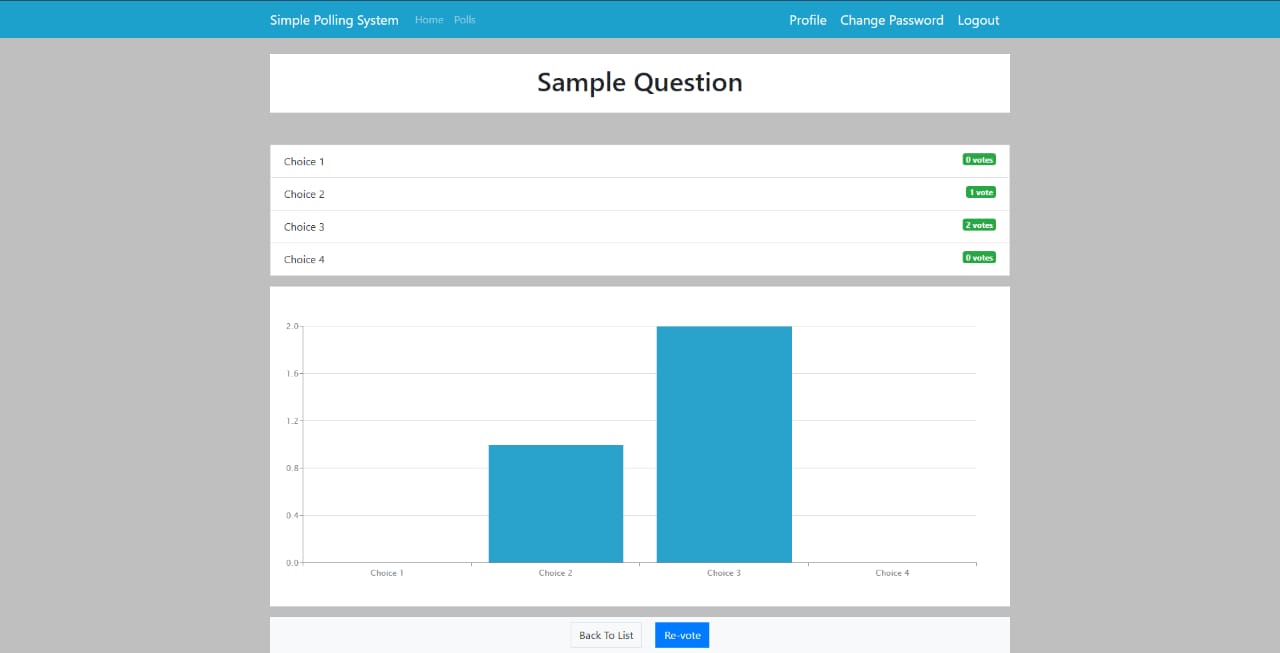
]

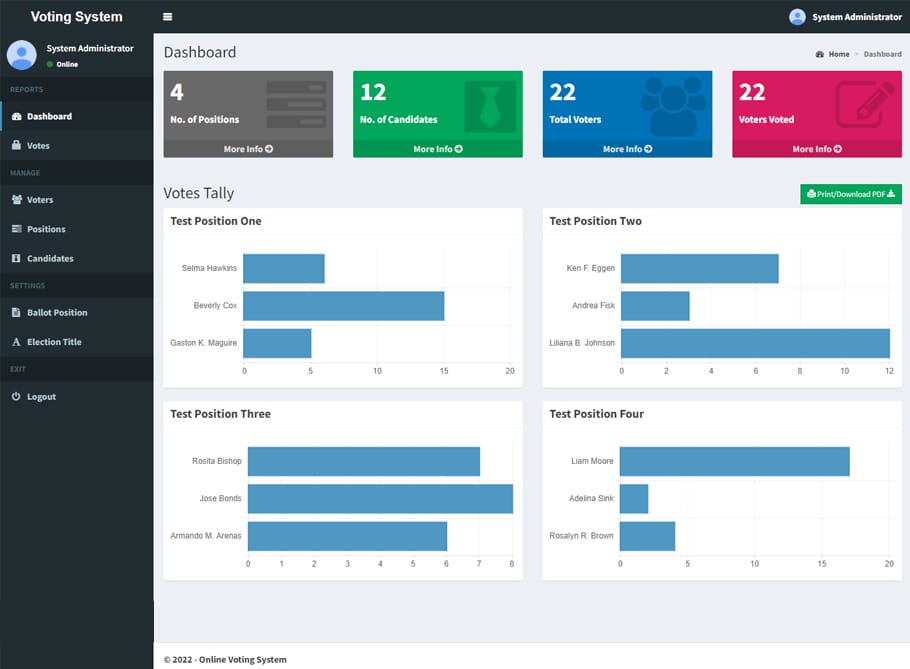
And finally, let's go ahead and migrate the existing migrations. We won't be using any of them, but we'll migrate just to get rid of the warnings later.l











# Steps to Create Django Online Voting System Project – Let’s Move ahead with an amazing project:

### 1.Open the terminal from the folder where we want to create the project.

Right click on mouse -> open in terminal -> write “code .” (space is mandatory between code and “.”)

### 2. Then go to the project folder ->

urls.py and inside urls.py of project, do this -> add “include” in import as shown in the code below and add “path(“”, include(“app.urls”))”

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path("", include("app.urls"))

]

### 3. Create urls.py in an app of wishlistproject

(urls.py is mandatory in both the project and app).

### 4. In setting.py app

add the ‘app name”. In my case, it is the app only as below.

## INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'app'

]

## 5. Now runserver to check, if everything is working or not.

If you see below image, then we are done with the installation part. To runserver, run command in terminal as follows “py manage.py runserver”.(if you see the rocket image, then it’s working fine).

## 6. Now create the models.py using ORM technique as follows.

from django.db import models

from django.contrib.auth.models import User

from django.core.validators import MinValueValidator

# Create your models here.

class Questions(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

ques = models.CharField(max\_length=150)

option1 = models.CharField(max\_length=150)

option2 = models.CharField(max\_length=150)

option3 = models.CharField(max\_length=150)

option4 = models.CharField(max\_length=150)

vote1 = models.IntegerField(default=0)

vote2 = models.IntegerField(default=0)

vote3 = models.IntegerField(default=0)

vote4 = models.IntegerField(default=0)

vote = models.IntegerField(default=False, verbose\_name="How many object created for this questions?")

is\_closed = models.BooleanField(default=False)

@property

def total\_votes(self):

return self.vote1 + self.vote2 + self.vote3 + self.vote4

@property

def get\_winner\_option(self):

options = [self.vote1, self.vote2, self.vote3, self.vote4]

max\_votes = max(options)

winner\_index = options.index(max\_votes)

if options.count(max\_votes) > 1:

return "It's a tie"

else:

if winner\_index == 0:

return self.option1

elif winner\_index == 1:

return self.option2

elif winner\_index == 2:

return self.option3

elif winner\_index == 3:

return self.option4

return

def \_str\_(self) -> str:

return self.ques

class Voted(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

voted\_question = models.ForeignKey(Questions, on\_delete=models.CASCADE)

class UserProfile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

age = models.PositiveIntegerField(validators=[MinValueValidator(0)])

To create the above field in a database, run the following commands as follows:

Py manage.py makemigrations

Py manage.py migrate



# Conclusion :

The provided code represents a basic online voting system implemented using Django framework. It includes registration, login, and voting functionalities. Users can register, log in, view and vote on

questions. The system ensures authentication, prevents duplicate votes, and tracks use acting.