Online Voting system for dijango framework

Prerequisite For Online Voting System Using Python Django

- A solid understanding of the Python programming language and the Django web framework is necessary.
- A strong understanding of HTML, CSS, and JavaScript is required to develop the project's user interface.
- Relational Database: You will need to have a good understanding of relational databases, such as SQLite, MySQL, or PostgreSQL, to create and manage the database for the Online voting system project.

Download Python Django Online Voting System Project

Please download the source code of the Python Django Online Voting System: Python Django Online Voting System Project Code

Project Setup

Minimum system configuration:

- The operating system requirements include Windows 7 or later, macOS 10.11 or later, or a modern operating system.
- Linux distribution.
- Processor: Intel Core i3 or equivalent.
- RAM: 4 GB or more
- Disk Space: 5 GB or more.
- Browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge can be used.

Visual Studio Code can be downloaded from the official website.

On the download page, you can select the suitable installer for your operating system (Windows, macOS, or Linux). After downloading the installer, run it and proceed with the installation instructions to install VS Code on your computer.

Here's a brief explanation of each step, along with the commands to execute:

1. Python should be installed: Download and install the latest version of Python from the official website, following the installation instructions for your operating system.

2. Install pip: Download the get-pip.py script and run python get-pip.py to install pip.

3. Create a virtual environment: Run python -m venv myenv to create a new virtual environment named 'myenv'.

4. Activate the virtual environment: Run source myenv/bin/activate on Linux/Mac or myenv\Scripts\activate on Windows to activate the virtual environment.

5. Install Django: Run pip install django to install the latest stable version of Django.

6. Verify installation: Run python -m django –version to verify that Django is installed correctly.

7. Create a new Django project: Run django-admin startproject project to create a new Django project named 'project'.

8. Start the development server: Run python manage.py runserver to start the development server.

That's it! A working installation of Django should now be in place, and you should be ready to start building your web application

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. create the models.py using the ORM technique as follows.

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

● Py manage.py makemigrations

● Py manage.py migrate

The file structure will look like this for our project:

App(main) -> Templates -> app(folder inside app) -> registration.html, login.html, votingpage.html.

● Now, execute the project step by step with code as follows:

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"))

Create urls.py in an app of wishlistproject(urls.py is mandatory in both the project and app).

In setting.py, 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. 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)

```python
    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

7.Create a Registration system

Templates/app -> registration.html

```
<div class="container">

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

    <div class="form-container">

      <h1>Registration</h1>

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

        {% csrf_token %}

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

        <input type="email" placeholder="Email" name="email">

        <input type="text" placeholder="Age" name="age">

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

        <input type="password" placeholder="Confirm Password" name="cpass">
```

```html
            <button type="submit">Register</button>

        </form>

        <p>Already registered? <a href="{% url 'index' %}">click here</a></p>

    </div>

  </div>
```

Views.py

```python
Def register(request):

  Return render(request, "app/registration.html")

Def Registration(request):

  If request.method == "POST":

    Username = request.POST['uname']

    Email = request.POST['email']

    Password = request.POST['passw']

    Cpassword = request.POST['cpass']

    Age = int(request.POST['age'])  # Convert age to an integer

    If password == cpassword:

      New_user = User.objects.create(

      Username = username,

      Email = email,

      )     New_user.set_password(password)

      New_user.save()

      Return redirect("index")

    Return
```

Urls.py

```python
Path("register", views.register, name="register"),

Path("registration", views.Registration, name="registration"),
```

8.Create a login system Templates.html -> login.html

```html
<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>
```

Views.py

```python
Def Loginview(request):

  If request.method == "POST":

    Username = request.POST['uname']

    Password = request.POST['password']

    User = authenticate(request, username=username, password=password)

   If user is not None:

      Login(request, user)

      Return redirect("show")

    Else:

      Return HttpResponse("invalid credentials")
```

Urls.py

Path("login", views.Loginview, name="login"),

9. Create a main page where voting can be done(one user can vote one time only).

Templates/app -> votingpage.html

```
<center>

    <h3 style="color: tomato;">

      {% if messages %}

      {% for message in messages %}

      <div>{{ message }}</div>

      {% endfor %}

      {% endif %}

    </h3>

  </center>

  <br><br>

 {% for item in new_ques %}

  <div class="container">

    <form method="post" action="{% url 'votingpage' pk=item.pk %}">

      {% csrf_token %}

      <h2>{{ item.ques }}</h2>

      <label><input type="radio" name="selected_option" value="1">{{ item.option1 }}</label><br>

      <label><input type="radio" name="selected_option" value="2">{{ item.option2 }}</label><br>

      <label><input type="radio" name="selected_option" value="3">{{ item.option3 }}</label><br>

      <label><input type="radio" name="selected_option" value="4">{{ item.option4 }}</label><br>
```

```html
<h4>Total Votes: {{item.total_votes}}</h4>

{% if item.is_closed %}

<p><strong>Winner: {{ item.get_winner_option }}</strong></p>

<button class="vote-button" type="button" disabled>

    Voting Closed

</button>

{% else %}

<button class="vote-button" type="submit">

    {% if user_profile.age > 18 %}

    Vote Now

    {% else %}

    Not allowed

    {% endif %}

</button>

    {% endif %}

  </form>

  <br>

</div>

{% endfor %}
```

Views.py

```python
@login_required(login_url="login")

Def Voting(request, pk):

  User = request.user

  Ques = get_object_or_404(Questions, pk=pk)

  # Check if the user is below 18 years old

  User_profile = UserProfile.objects.get(user=user)
```

```
If user_profile.age < 18:

    Messages.warning(request, "Voters below 18 years of age are not allowed to vote.")

    Return redirect("votingpage")

# Check if the user has already voted for this question

If Voted.objects.filter(user=user, voted_question=ques).exists():

    Messages.warning(request, "You have already voted for this question.")

    Return redirect("already")

Else:

    Selected_option = request.POST.get('selected_option')

    If selected_option in ['1', '2', '3', '4']:

        Setattr(ques, f'vote{selected_option}', getattr(ques, f'vote{selected_option}') + 1)

        Ques.save()

        Voted.objects.create(user=user, voted_question=ques)

        Messages.success(request, "Your vote has been recorded.")

    Else:

        Messages.warning(request, "Invalid vote selection.")

    Return redirect('show')

@login_required(login_url="login")

Def show(request):

    New_ques = Questions.objects.all()

    # Check if the user's age is less than 18 and show a warning message

    User_profile = UserProfile.objects.get(user=request.user)

    If user_profile.age < 18:

        Messages.warning(request, "Voter below 18 age group is not allowed.")

    Return render(request, "app/votingpage.html", {"new_ques": new_ques, "user_profile": user_profile})
```

Urls.py

Path("home", views.home, name="home"),

Path("votingpage/<int:pk>", views.Voting, name="votingpage"),

Path("showques", views.show, name="show"),

10. For logout process

Views.py

@login_required(login_url="login")

Def signout(request):

   Logout(request)

   Return redirect("index")

Urls.py

Path("logout", views.signout, name="logout")

Explanation of the above snippets:

Certainly! Here's an explanation of each view in the provided code:

1.Index(request):

- This view renders the login page (login.html).

2.Register(request):

- This view renders the registration page (registration.html).

3.Registration(request):

- view handles the registration process when the registration form is submitted.
- It first checks if the passwords provided by the user match.
- If the passwords match, a new User object is created with the provided username and email.
- The password is set and encrypted using the set_password method.

- The new user is saved to the database, and the user is redirected to the login page (index).

4.Loginview(request):

- This view handles the login process when the login form is submitted.
- It retrieves the username and password entered by the user.
- The authenticate function is used to verify the credentials. If the authentication is successful, the user is logged in using the login function, and they are redirected to the "show" page.
- If the authentication fails, an "invalid credentials" message is displayed.

5.Home(request):

- This view renders the main page (votingpage.html) after the user has successfully logged in.
- The @login_required decorator ensures that only authenticated users can access this page. If a user is not logged in, they are redirected to the login page.

6.Voting(request, pk):

- This view handles the process of voting for a specific question.
- It first retrieves the authenticated user and the question (specified by its primary key pk) using get_object_or_404.
- It checks If the user has already voted for the question by searching for an existing record in the Voted model.
- If the user has not voted for the question, the vote count for the question is incremented, and the vote is saved.
- A new record is created in the Voted model to indicate that the user has voted for that question.
- The user is then redirected to the "show" page.

7.Show(request):

- This view retrieves all the questions from the Questions model and renders the "votingpage.html" template.
- The retrieved questions are passed to the template as the context variable new_ques.

8.Signout(request):

- This view handles the logout process.
- The logout function is called to log out the user.
- The user is then redirected to the login page (index).
- These views, along with the corresponding URL patterns defined in the urlpatterns list, form the routing and logic for the online voting system.

Here is the complete code for Python Django Online Voting System as follows:

Templates/app -> registration.html

<!DOCTYPE html>

<html>

<head>

  <title>Registration</title>

  <style>

    Body {

      Background-color: #ffffff;

      Display: flex;

      Justify-content: center;

      Align-items: center;

      Height: 100vh;

      Margin: 0;

    }

    .container {

      Background-color: #ffffff;

      Border-radius: 5px;

```css
  Box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  Padding: 20px;

  Width: 300px;

}

H1 {

  Text-align: center;

}

Form {

  Display: flex;

  Flex-direction: column;

}

Input[type="text"],

Input[type="email"],

Input[type="password"] {

  Padding: 10px;

  Margin-bottom: 10px;

  Border: 1px solid #ccc;

  Border-radius: 4px;

}

Button {

  Padding: 10px;

  Background-color: #4caf50;

  Color: #ffffff;

  Border: none;

  Border-radius: 4px;

  Cursor: pointer;
```

```html
        }
        Button:hover {
          Background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="container">
      <h1><u>DataFlair Voting System</u></h1>
      <div class="form-container">
        <h1>Registration</h1>
        <form action="{% url 'registration' %}" method="post">
          {% csrf_token %}
          <input type="text" placeholder="Name" name="uname">
          <input type="email" placeholder="Email" name="email">
          <input type="text" placeholder="Age" name="age">
          <input type="password" placeholder="Password" name="passw">
          <input type="password" placeholder="Confirm Password" name="cpass">
          <button type="submit">Register</button>
        </form>
        <p>Already registered? <a href="{% url 'index' %}">click here</a></p>
      </div>
    </div>
</body>
</html>
```

Templates/ app -> votingpage.html

```html
<!DOCTYPE html>

<html>


<head>

  <link rel="stylesheet" type="text/css" href="style.css">

  <style>

    Nav {

      Background-color: #f2f2f2;

      Height: 60px;

      Display: flex;

      Justify-content: space-between;

      Align-items: center;

      Padding: 0 20px;

    }

    .logo img {

      Height: 40px;

    }

    .username {

      Font-size: 18px;

    }

    .container {

      Max-width: 400px;

      Margin: 0 auto;

      Padding: 20px;

      Background-color: #f2f2f2;
```

```css
    Border-radius: 5px;

}

H2 {

    Text-align: center;

    Margin-bottom: 20px;

}

.option {

    Display: flex;

    Align-items: center;

    Margin-bottom: 10px;

}

Input[type="radio"] {

    Margin-right: 10px;

}

Label {

    Font-size: 16px;

}

.vote-button {

    Display: block;

    Width: 100%;

    Margin-top: 20px;

    Padding: 10px;

    Font-size: 16px;

    Text-align: center;

    Background-color: #4caf50;

    Color: white;
```

```
      Border: none;

      Border-radius: 5px;

      Cursor: pointer;

    }

    .vote-button:hover {

      Background-color: #45a049;

    }

  </style>

</head>


<body>

  <nav>

    <div class="logo">

      <h1>DataFlair Voting System</h1>

    </div>

    <div class="username">

      {% if user.is_authenticated %}

      <span>Welcome, {{ user.username }} <a href="{% url 'logout'
%}">(logout)</a></span>

      {% endif %}

    </div>

  </nav>

  <br><br>

  {% for item in new_ques %}

  <div class="container">

    <form method="post" action="{% url 'votingpage' pk=item.pk %}">
```

```html
    {% csrf_token %}

  <h2>{{ item.ques }}</h2>

    <button class="vote-button" type="submit">{% if vote.is_valid %} voted {% else %}
vote {% endif %}</button>

  </form>

  <br>

 </div>

 {% endfor %}


</body>

</html>
```

Urls.py

```python
From django.urls import path,include

From . import views

Urlpatterns = [

  Path("", views.Index, name="index"),

  Path("register", views.register, name="register"),

  Path("registration", views.Registration, name="registration"),

  Path("home", views.home, name="home"),

  Path("login", views.Loginview, name="login"),

  Path("votingpage/<int:pk>", views.Voting, name="votingpage"),

  Path("showques", views.show, name="show"),

  Path("logout", views.signout, name="logout"),

  Path("successfully", views.successfully, name="successfully"),

  Path("already", views.already, name="already")

]
```

Python Django Online Voting System Output:

DataFlair Voting System

Login

Username

Password

Login

Not registered? click here

DataFlair Voting System Welcome, raje (logout)

Voter below 18 age group is not allowed.

Who?

○ Virat kohli
○ Dhoni
○ Sachin
○ Hardik Pandya

Total Votes: 2

Winner: It's a tie

Voting Closed

Who will win this year?

○ Mumbai Indians
○ Rajasthan Royals
○ Gujarat titans
○ Kolkata knight Riders

Total Votes: 1

Winner: Rajasthan Royals

Voting Closed

happy or not?

○ Javscript
○ python
○ ubuntu
○ html

Total Votes: 1

Not allowed

Admin panel

Now how would we add data, delete, and update to the user side? Let's come to the django inbuilt admin panel. So here are the details of the same:
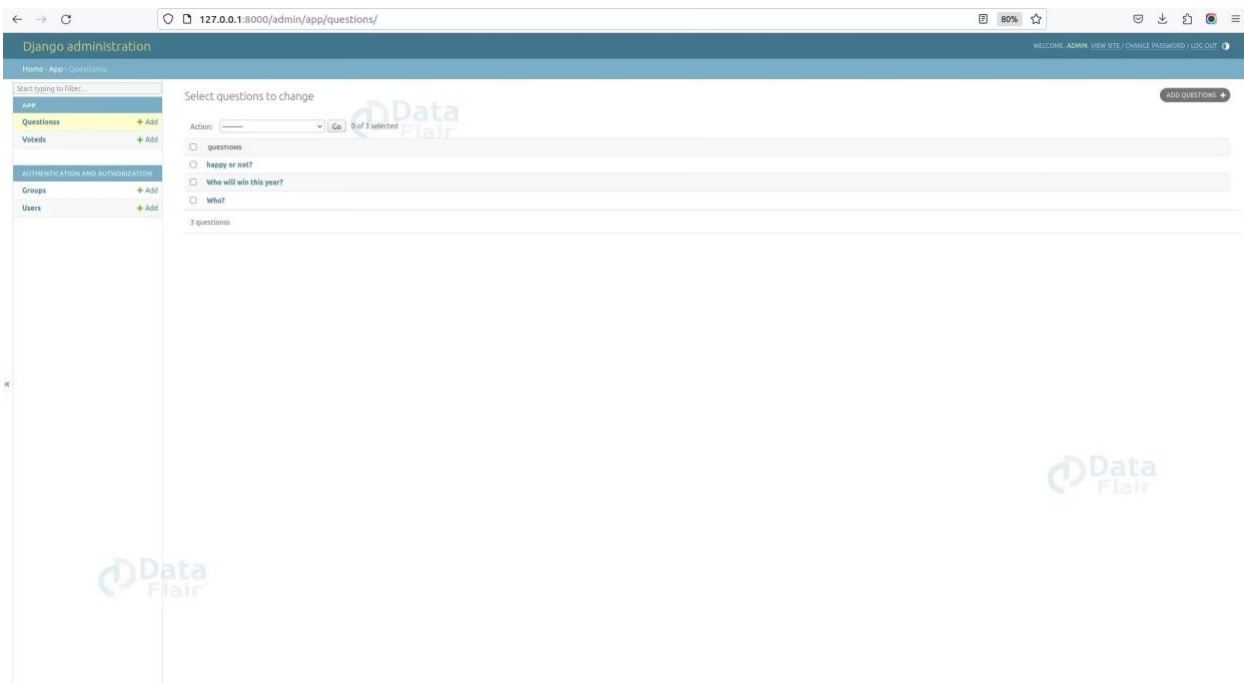
1. Once you have your Django project set up and have created your models, you can use the admin panel to add data easily. Follow these ssteps

2. Run python manage.py createsuperuser to create an admin user who can access the admin panel. Start the development server using python manage.py runserver. Open your browser and go to the admin panel URL (usually http://127.0.0.1:8000/admin/) Log in using the superuser credentials you created.

3. Then, Go to admin.py and add the following thing as follows:

From django.contrib import admin

From .models import YourModelName

Admin.site.register(YourModelName)

   Note: To display the model on the admin panel, you need to follow the third point.

Thank you!