An online voting system, also known as e-voting, is a digital platform that allows eligible voters to cast their votes remotely using the internet. It is a modern alternative to traditional paper-based voting methods. With an online voting system, voters can access the voting process through a secure website or a dedicated voting application.

About Python Django Online Voting System

The objective of a Python Django online voting system is to provide a convenient and efficient method for eligible voters to cast their votes remotely using the internet.

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: 4GB or more

Disk Space: 5GB 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.

From django.db import models

From django.contrib.auth.models import User

```
Fromdjango.core.validatorsimportMinValueValidator

#Createyourmodelshere.

ClassQuestions(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=" Howmanyobjectcreated
forthisquestions?" )

  Is_closed=models.Booleanfield(default=False)


@property

Deftotal_votes(self):

  Returnself.vote1+self.vote2+self.vote3+self.vote4


@property

Defget_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' satie"
Else:
  If winner_index==0:
    Returnself.option1
    Elif winner_index==1:
      Returnself.option2
    Elif winner_index==2:
      Returnself.option3
    Elif winner_index==3:
      Returnself.option4
  Return

Def__str__(self)->str:
  Returnself.ques

ClassVoted(models.Model):
  User=models.ForeignKey(User,on_delete=models.CASCADE)
  Voted_question=models.ForeignKey(Questions,on_delete=models.CASCADE)

ClassUserProfile(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:

Pymanage.py makemigrations

Pymanage.py migrate

7. Create a Registration system

```
<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>

<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>

{% endfor %}

Views.py

@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," Youhavealreadyvotedforthisquestion." )
    Returnredirect(" already" )
  Else:
    Selected_option=request.POST.get(' selected_option' )
    If selected_optionin[' 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," Yourvotehasbeenrecorded." )
    Else:
      Messages.warning(request," Invalidvoteselection." )

    Returnredirect(' show' )

@login_required(login_url=" login" )
Def show(request):
New_ques=Questions.objects.all()

#Checkiftheuser' sageislessthan18andshowawarningmessage
User_profile=UserProfile.objects.get(user=request.user)
If user_profile.age<18:
  Messages.warning(request," Voterbelow18agegroupisnotallowed." )
```

Returnrender(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.Forlogoutprocess

Views.py

@login_required(login_url=" login")

Def signout(request):

Logout(request)

Returnredirect(" 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):

This 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 logout 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.

```
.container{
  Background-color:#ffffff;
  Border-radius:5px;
  Box-shadow:02px5pxrgba(0,0,0,0.1);
  Padding:20px;
  Width:300px;
}
```

```
H1{
  Text-align:center;
}

Form{
  Display:flex;
  Flex-direction:column;
}

Input[type=" text" ],
Input[type=" password" ]{
  Padding:10px;
  Margin-bottom:10px;
  Border:1pxsolid#ccc;
  Border-radius:4px;
}

Button{
  Padding:10px;
  Background-color:#4caf50;
  Color:#ffffff;
  Border:none;
  Border-radius:4px;
  Cursor:pointer;
```

```
    }

    Button:hover{
      Background-color:#45a049;
    }
  </style>
</head>
<body>
 <divclass=" container" >
  <h1><u>DataflairVotingSystem</u></h1>
  <divclass=" form-container" >
   <h1>Login</h1>
   <formaction=" {%url' login' %}" method=" post" >
    {%csrf_token%}
    <inputtype=" text" placeholder=" Username" name=" uname" >
    <inputtype=" password" placeholder=" Password" name=" password" >
     <buttontype=" submit" >Login</button>
   </form>
   <p>Notregistered?<ahref=" {%url' register' %}" >clickhere</a></p>
  </div>
 </div>
</body>
```

```
</html>
  #Checkifthe user' sageislessthan18andshowawarningmessage
  User_profile=UserProfile.objects.get(user=request.user)
  If user_profile.age<18:
    Messages.warning(request," Voterbelow18agegroupisnotallowed." )

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

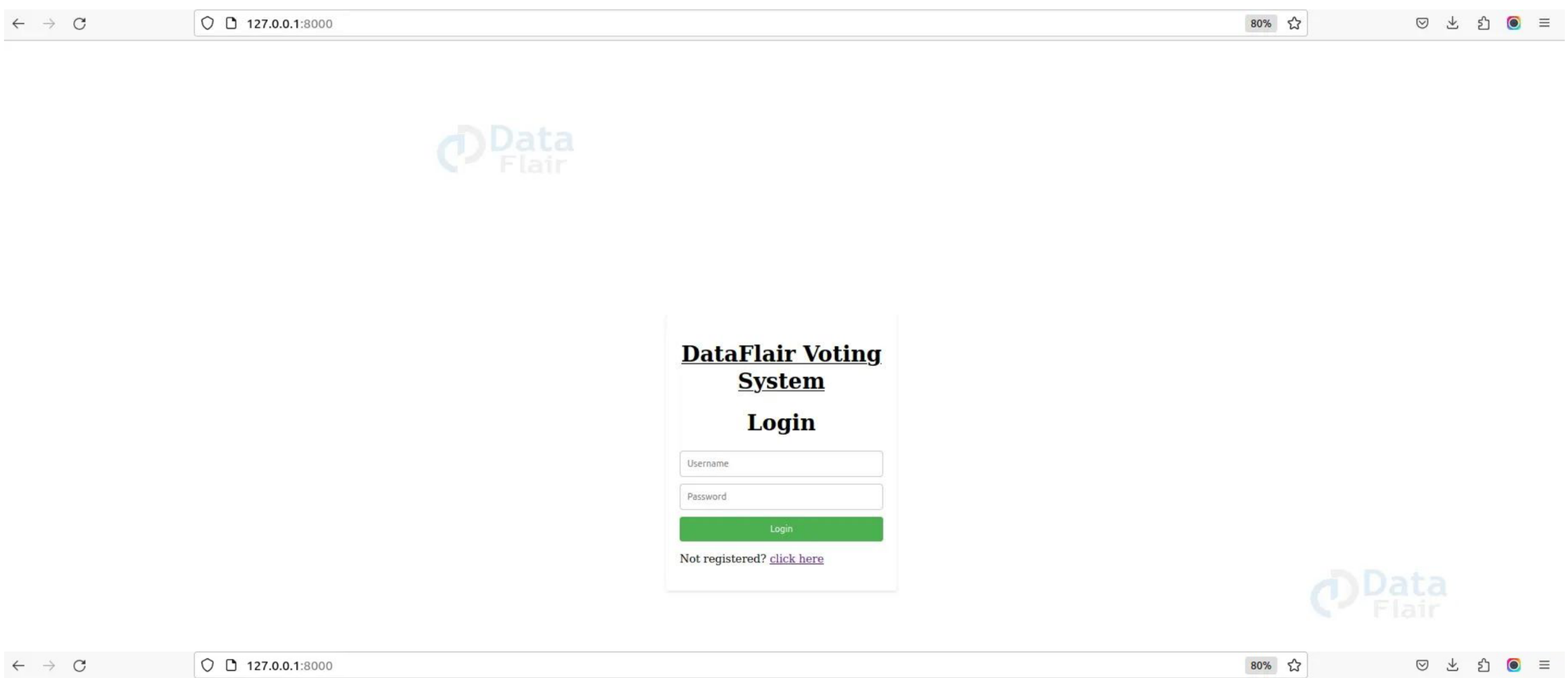
```
Def signout(request):
  Logout(request)
  Returnredirect(" index" )
```

Urls.py

```
From django.urlsimportpath,include
From .importviews
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



127.0.0.1:8000/showques

80%

DataFlair Voting System

Welcome, chirag4 (logout)

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

Vote Now

127.0.0.1:8000/showques

80%

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

Onlinevotingsystemloginoutput

Onlinevotingsystemregistrationoutput

Onlinevotingsystemprojectoutput

Votingpagebelowage18

Adminpanel

Nowhowwouldweadddata,delete,andupdatetotheuserside?Let' scometothedjango inbuiltadminpanel.Soherearethedetailsofthesame:

1.Onceyouhaveyour Djangoprojectsetupandhavecreatedyourmodels,youcanusethe adminpaneltoadddataeasily.Followthesesteps:

2.Runpythonmanage.pycreatesuperusertocreateanadminuserwhocanaccessthe adminpanel.Startthedevelopmentserverusingpythonmanage.pyrunserver.Openyour browserandgototheadminpanelURL(usuallyhttp://127.0.0.1:8000/admin/)Loginusing thesuperusercredentialsyoucreated.

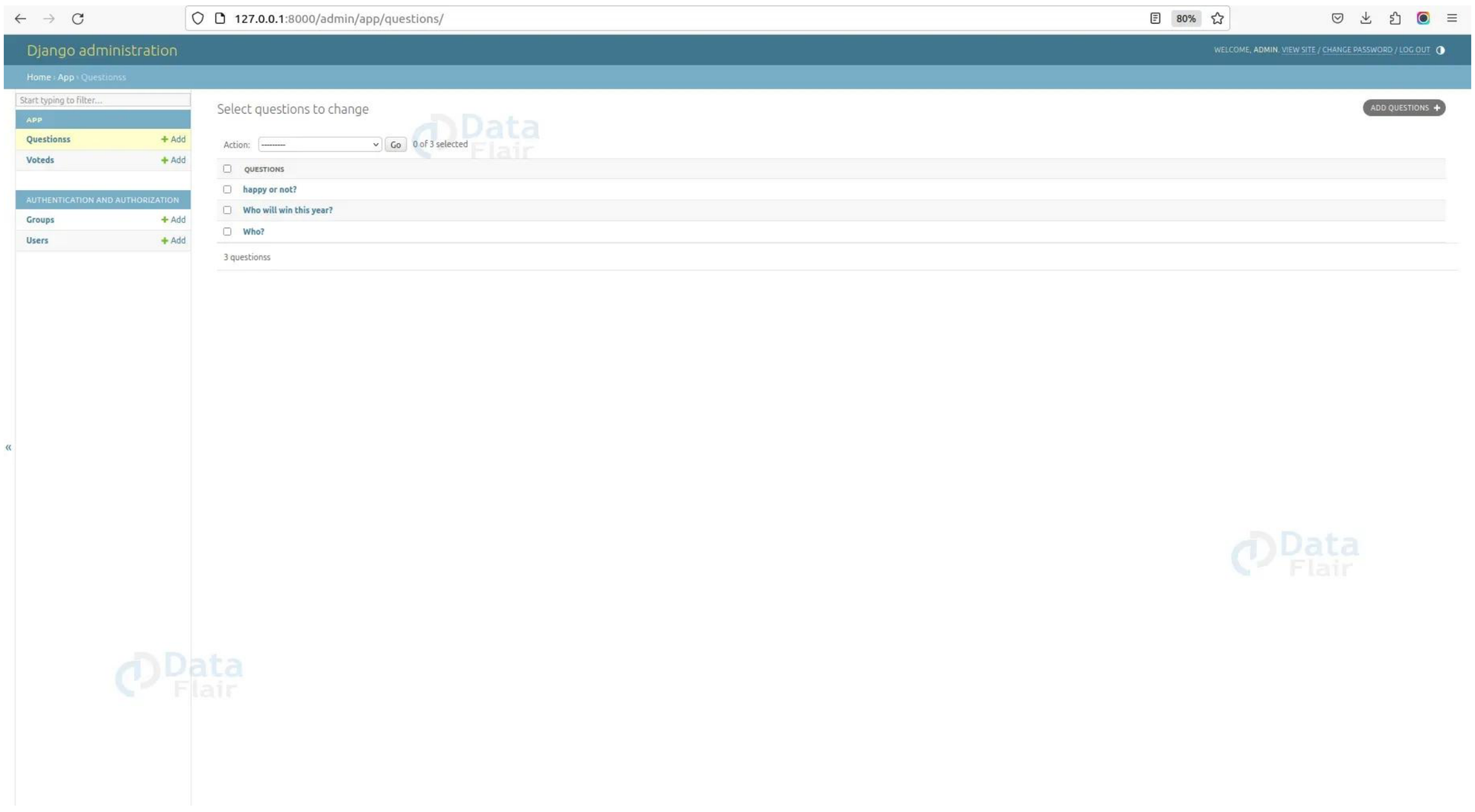
3.Then,Gotoadmin.pyandaddthefollowingthingasfollows:

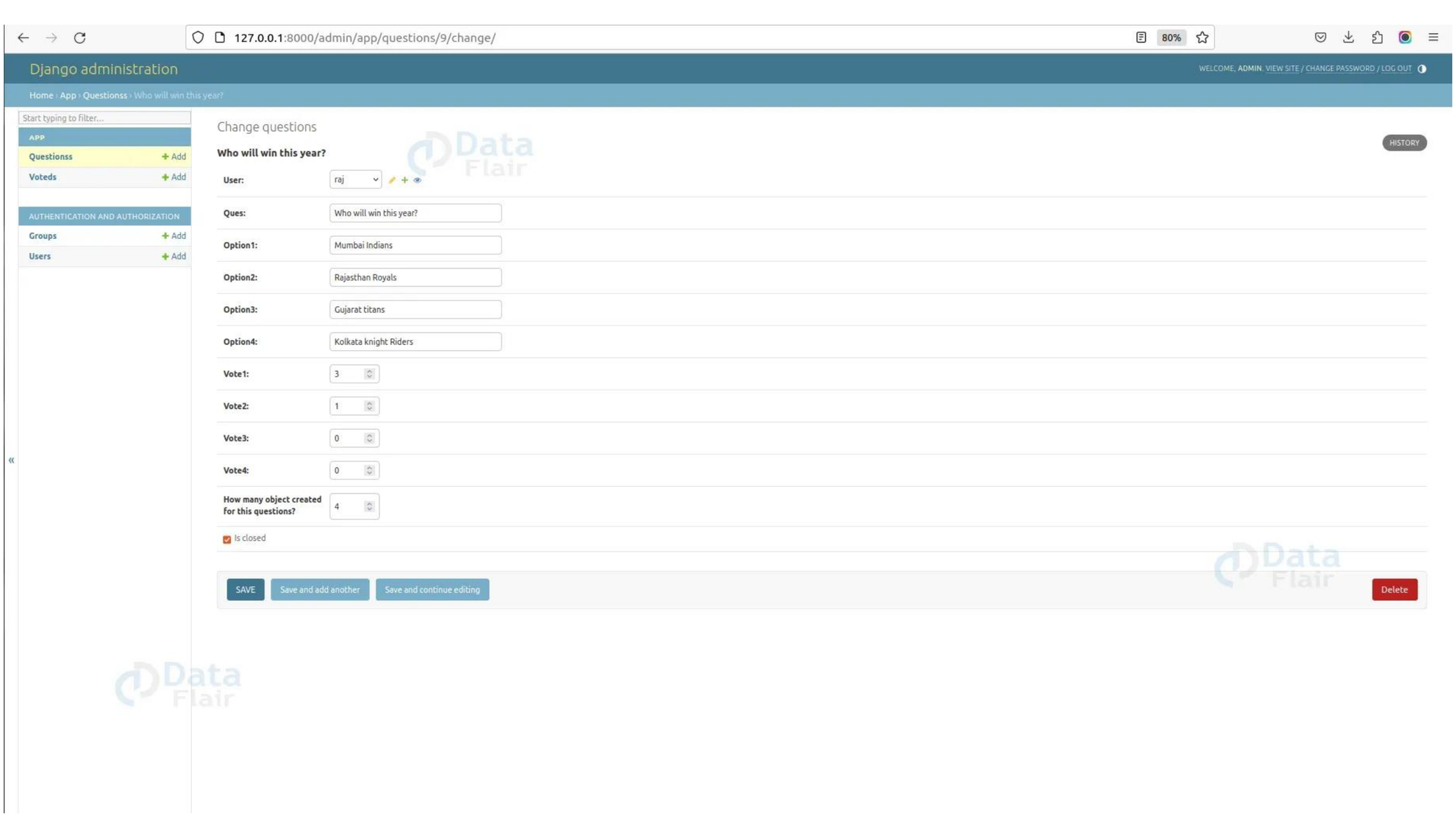
Fromdjango.contribimportadmin

From.modelsimportYourModelName

Admin.site.register(YourModelName)

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





Admin panel show question

Total votes admin page

Summary:

The provided code represents a basic online voting system implemented using Django framework. It includes registration, login, and voting functionalities. Users can register, login, view and vote on questions. The system ensures authentication, prevents duplicate votes, and tracks user activity.