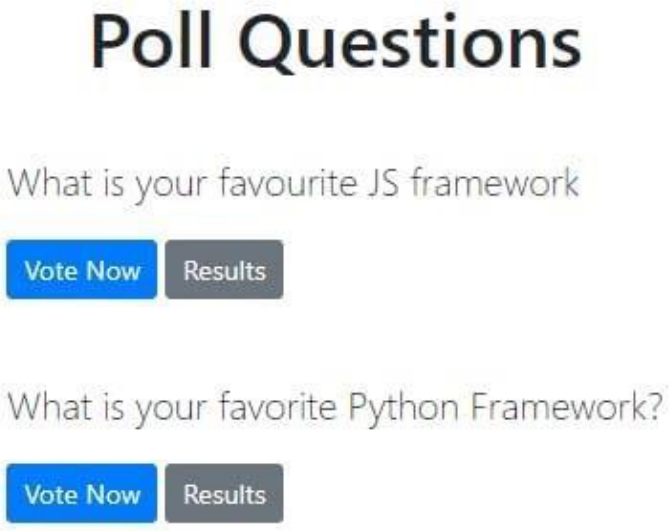
Voting System Project Using Django Framework

Project Title: Pollster (Voting System) web application using Django framework Type of Application (Category): Web application.

Poll questions:



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.

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 command.

Pip install pipenv

Pipenv shell

Step-3: A Pipfile will be created in your folder from the above step. Now install Django in your folder using the following command.

Pipenv install django

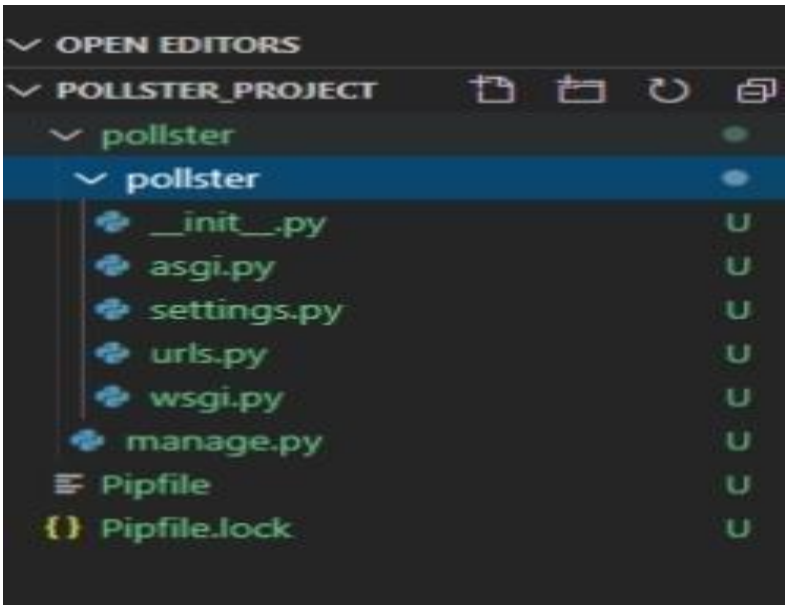
Step-4: Now we need to establish the Django project. Run the following command in your folder and initiate a Django project.

Django-admin startproject 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.

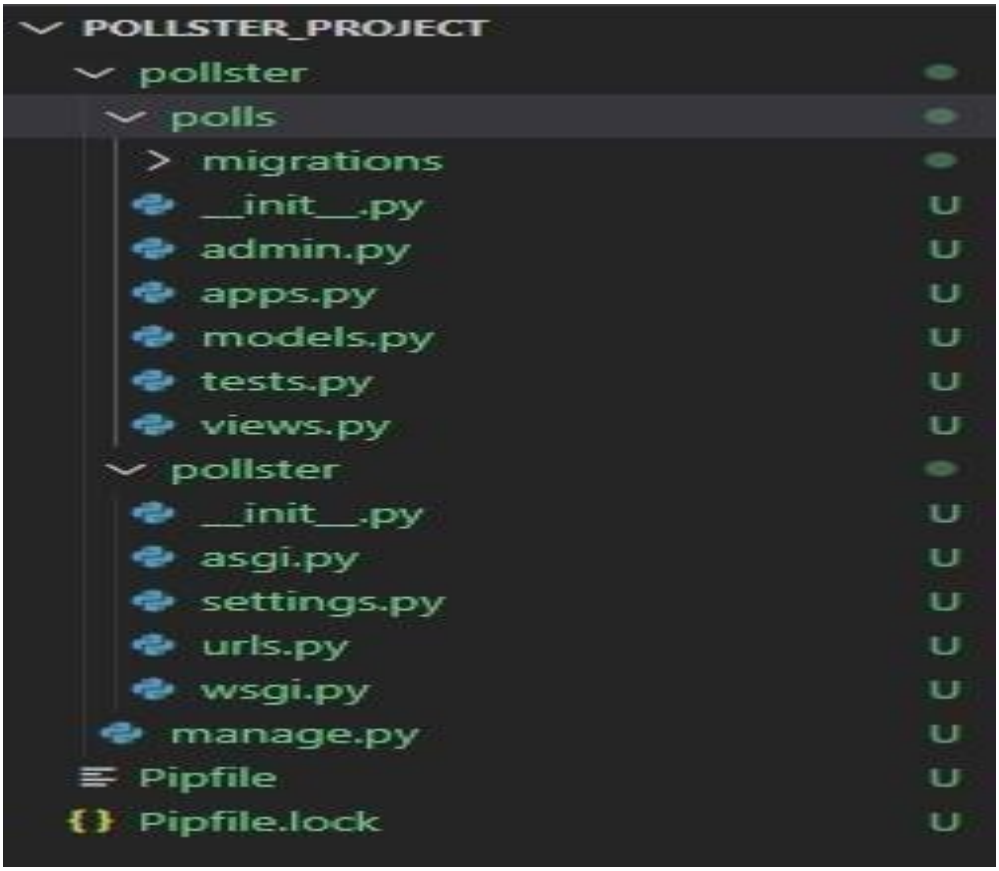


Python manage.py runserver

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

Python manage.py startapp polls

Below is the folder structure after creating ”polls’ app in the project.


POLLSTER_PROJECT
pollster
polls
migrations
__init__.py U
admin.py U
apps.py U
models.py U
tests.py U
views.py U
pollster
__init__.py U
asgi.py U
settings.py U
urls.py U
wsgi.py U
manage.py U
Pipfile U
Pipfile.lock U

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)
   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”>
<button type=”submit”>Register</button>
      </form>
      <p>Already registered? <a href=”{% url ‘index’ %}”>click here</a></p>
    </div>
  </div>
```

Views.py

```
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

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

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

```
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>
      <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

```
@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;
    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;
    }
    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

```
<!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;
    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 %}
```

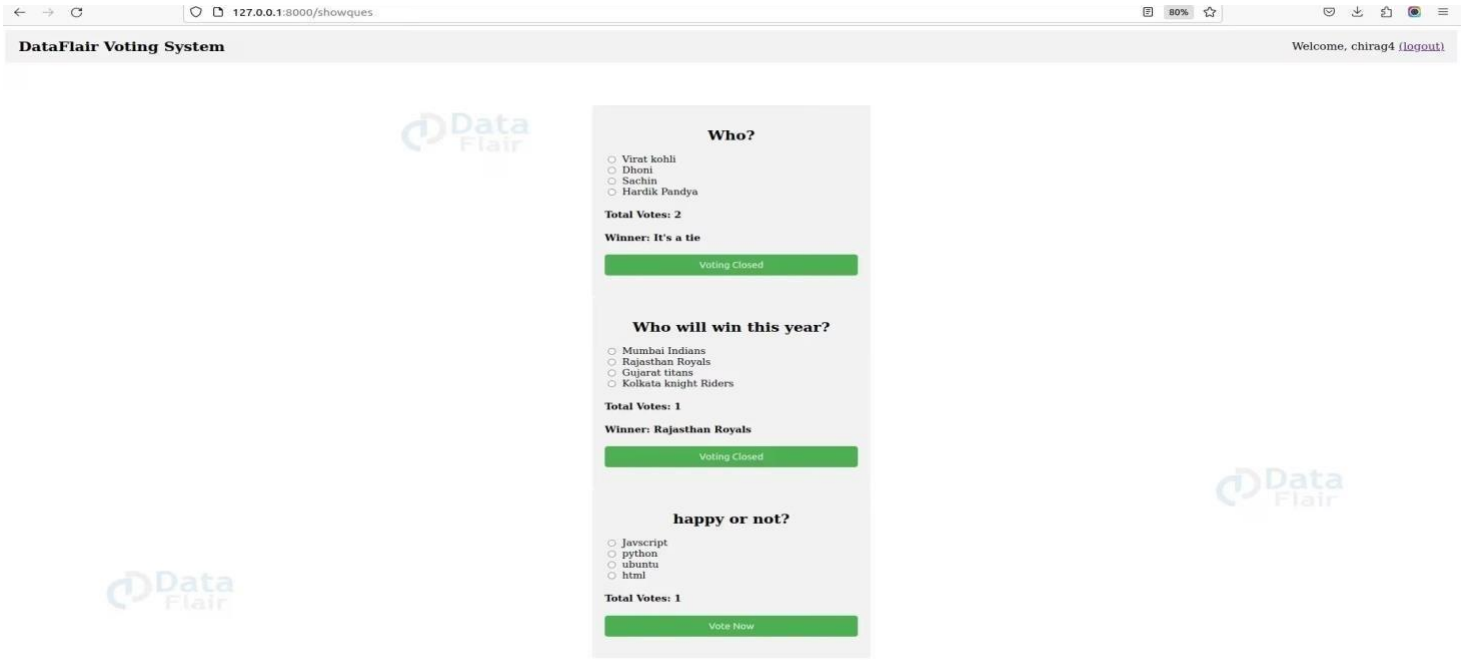
```
    </div>
  </nav>
  <br><br>
  {% for item in new_ques %}
  <div class="container">
    <form method="post" action="{% url 'votingpage' pk=item.pk %}"> {% csrf_token %}
    <h2>{{ item.ques }}</h2>
      <button class="vote-button" type="submit">{% if vote.is_valid %} voted {% else %}
vote {% endif %}</button>
    </form>
    <br>
  </div>
  {% endfor %}


</body>
</html>
```

Urls.py

```
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"),
```

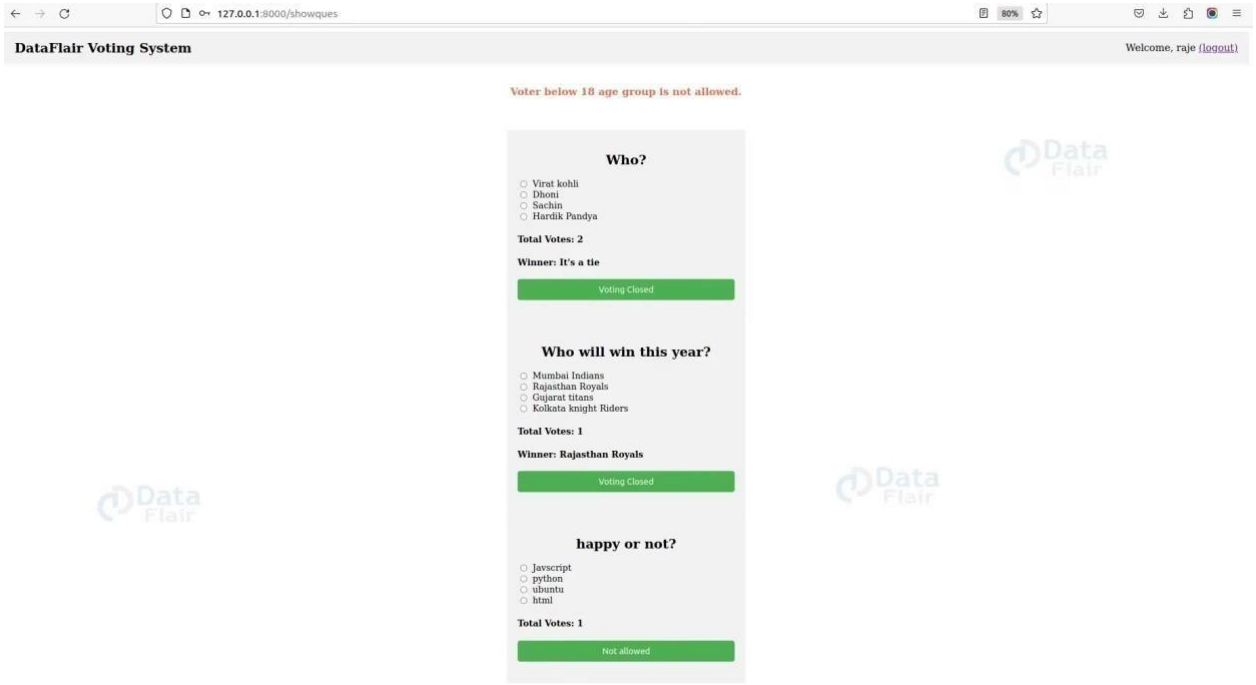
```
    Path("showques", views.show, name="show"),
    Path("logout", views.signout, name="logout"),
    Path("successfully", views.successfully, name="successfully"),
    Path("already", views.already, name="already")
]
```

Python Django Online Voting System Output:




127.0.0.1:8000
80%


DataFlair Voting System
Login
Username
Password
Login
Not registered? click here


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

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:

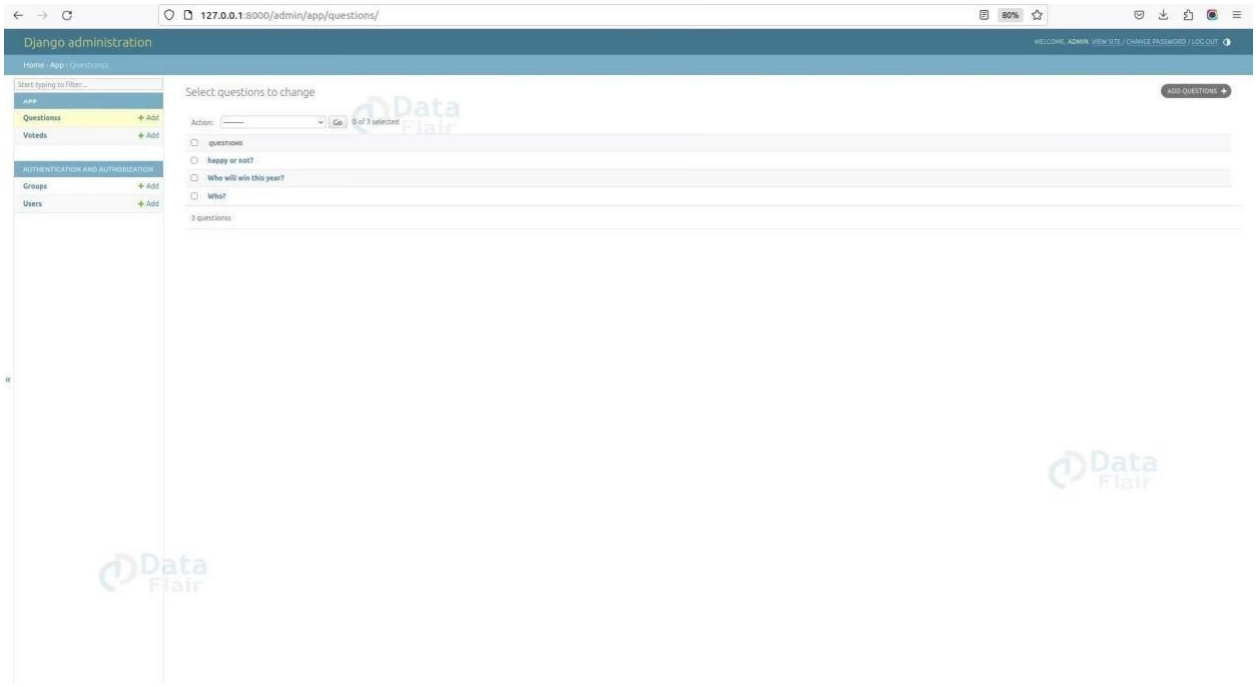
1. Once you have your Django project set up and have created your models, you can use the admin panel to add data easily. Follow these ssteps
2. Run python manage.py createsuperuser to create an admin user who can access the admin panel. Start the development server using python manage.py runserver. 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:

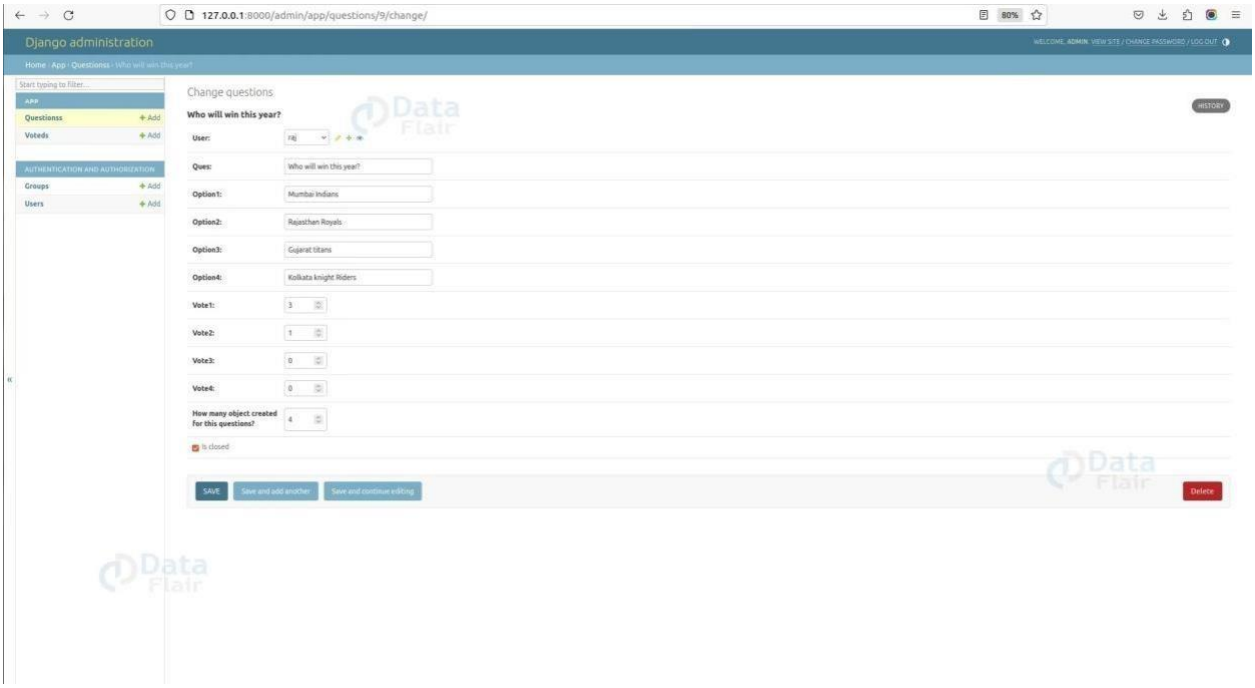
From django.contrib import admin

From .models import YourModelName

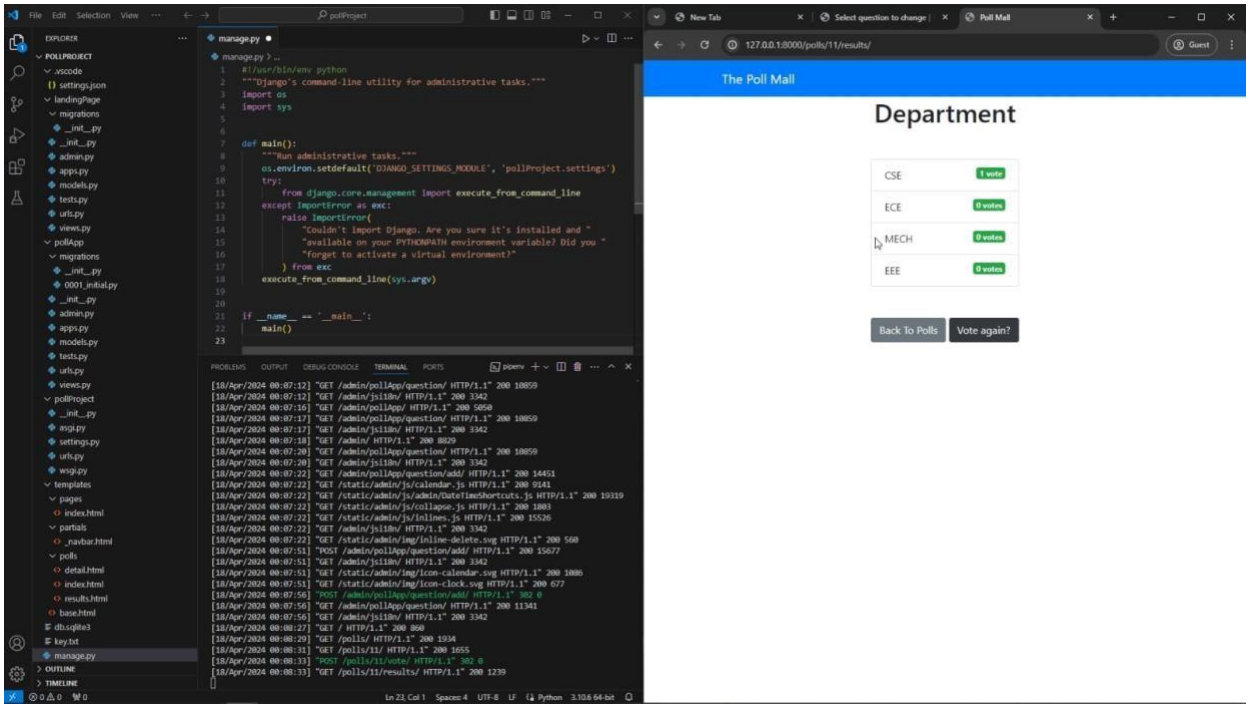
Admin.site.register(YourModelName)

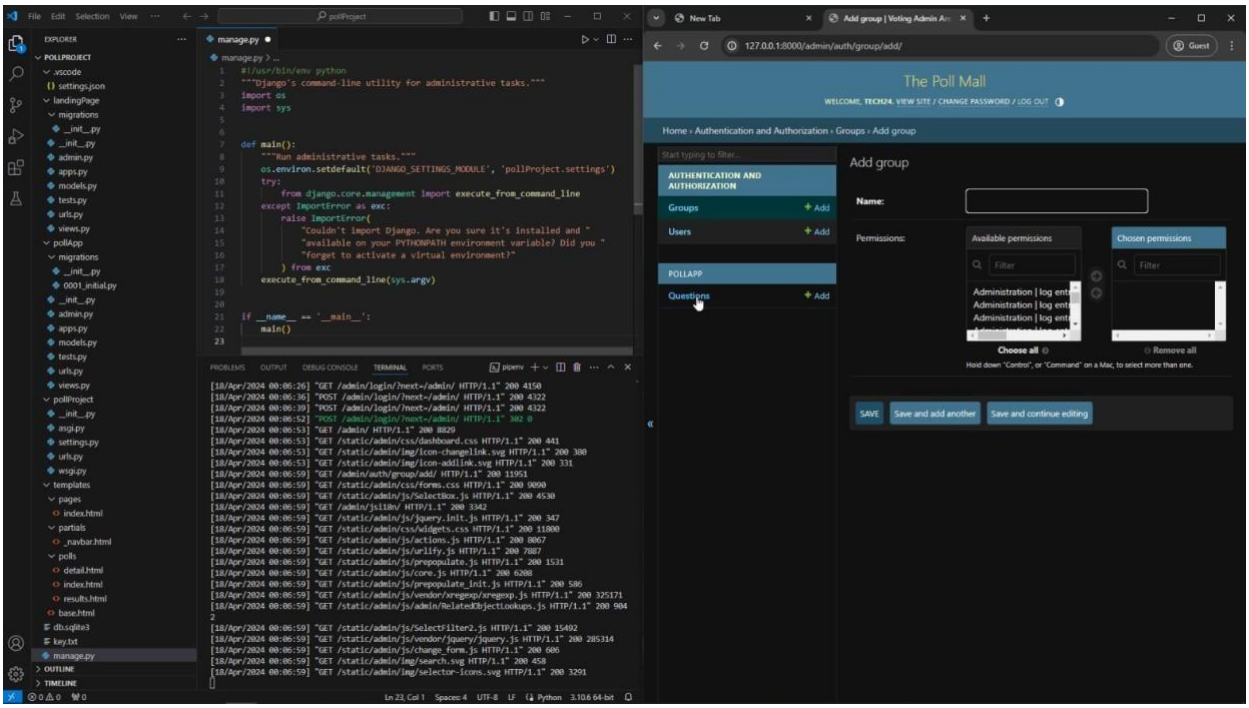
Note: To display the model on the admin panel, you need to follow the third point.

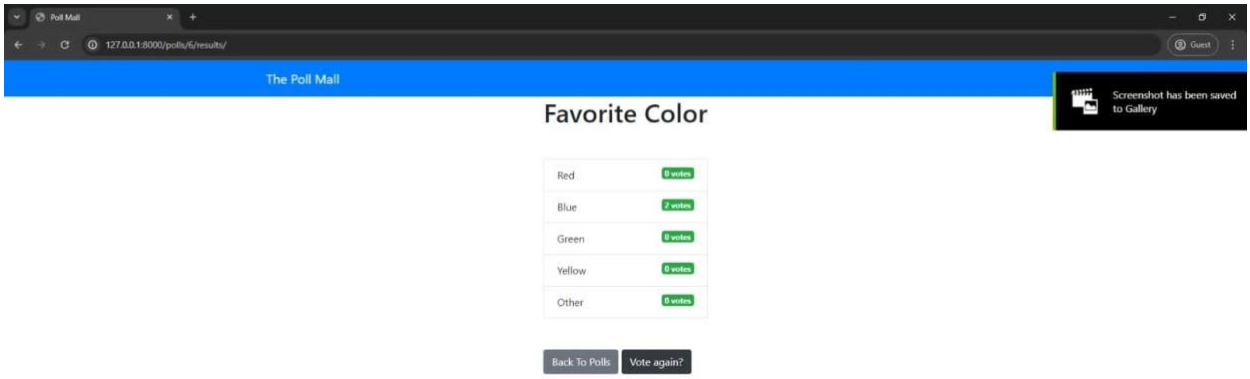


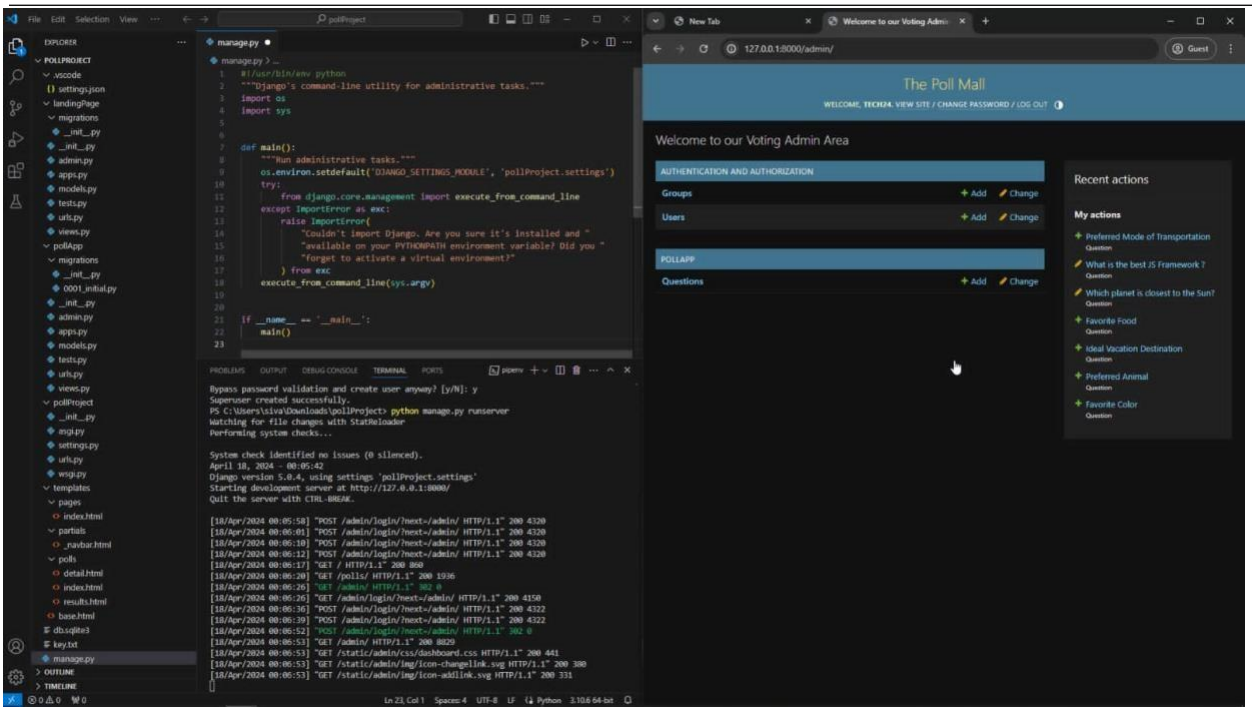


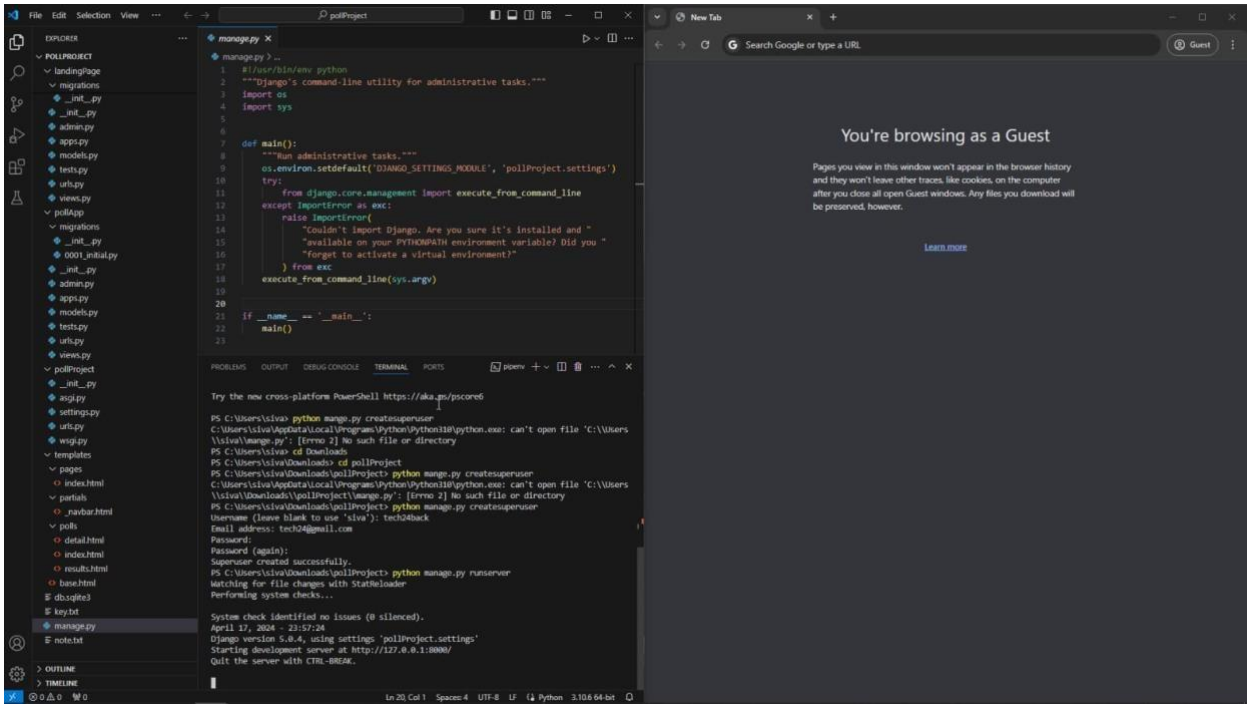
Outputs and execution:

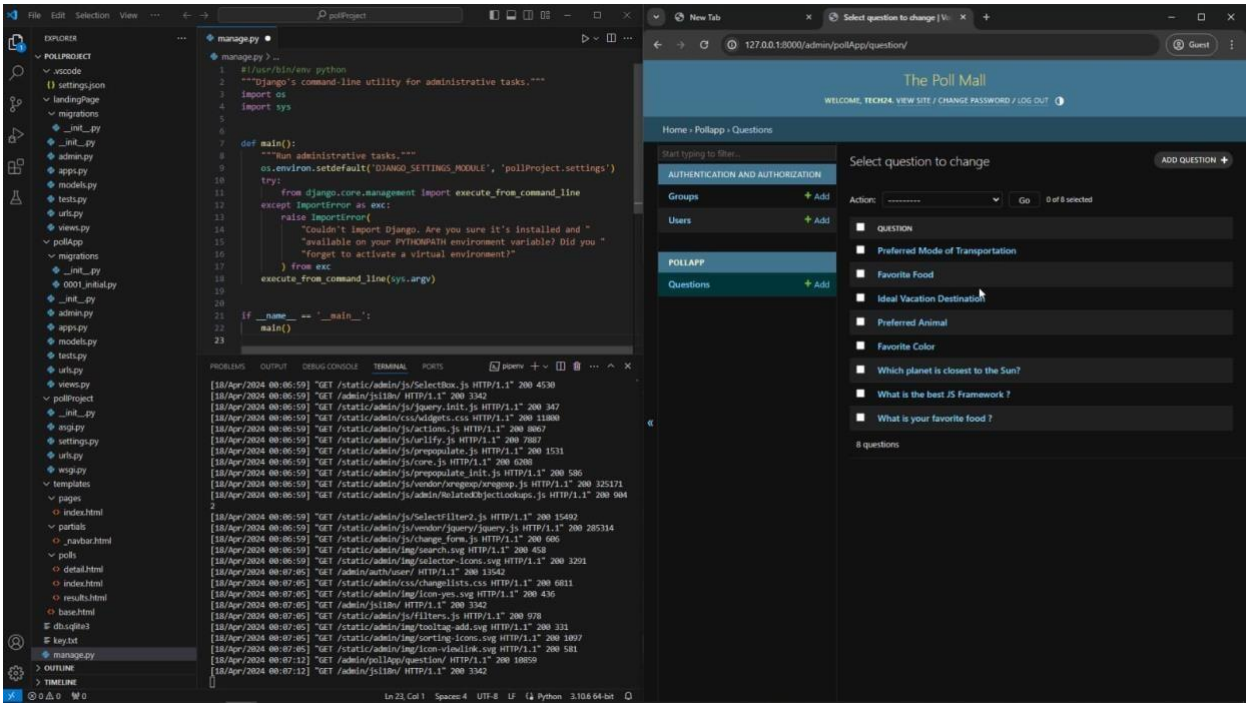

manage.py
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
"""Run administrative tasks."""
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'pollProject.settings')
try:
from django.core.management import execute_from_command_line
except ImportError as exc:
raise ImportError(
"Couldn't import Django. Are you sure it's installed and "
"available on your PYTHONPATH environment variable? Did you "
"forget to activate a virtual environment?"
) from exc
execute_from_command_line(sys.argv)
if __name__ == '__main__':
main()
[18/Apr/2024 00:07:12] "GET /admin/pollApp/question/ HTTP/1.1" 200 10859
[18/Apr/2024 00:07:12] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:07:16] "GET /admin/pollApp/ HTTP/1.1" 200 5050
[18/Apr/2024 00:07:17] "GET /admin/pollApp/question/ HTTP/1.1" 200 10859
[18/Apr/2024 00:07:17] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:07:18] "GET /admin/ HTTP/1.1" 200 8829
[18/Apr/2024 00:07:20] "GET /admin/pollApp/question/ HTTP/1.1" 200 10859
[18/Apr/2024 00:07:20] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:07:22] "GET /admin/pollApp/question/add/ HTTP/1.1" 200 14451
[18/Apr/2024 00:07:22] "GET /static/admin/js/calendar.js HTTP/1.1" 200 9141
[18/Apr/2024 00:07:22] "GET /static/admin/js/admin/DateTimeShortcuts.js HTTP/1.1" 200 19319
[18/Apr/2024 00:07:22] "GET /static/admin/js/collapse.js HTTP/1.1" 200 1803
[18/Apr/2024 00:07:22] "GET /static/admin/js/inlines.js HTTP/1.1" 200 15526
[18/Apr/2024 00:07:22] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:07:22] "GET /static/admin/img/inline-delete.svg HTTP/1.1" 200 560
[18/Apr/2024 00:07:51] "POST /admin/pollApp/question/add/ HTTP/1.1" 200 15677
[18/Apr/2024 00:07:51] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:07:51] "GET /static/admin/img/icon-calendar.svg HTTP/1.1" 200 1086
[18/Apr/2024 00:07:51] "GET /static/admin/img/icon-clock.svg HTTP/1.1" 200 677
[18/Apr/2024 00:07:56] "POST /admin/pollApp/question/add/ HTTP/1.1" 302 0
[18/Apr/2024 00:07:56] "GET /admin/pollApp/question/ HTTP/1.1" 200 11341
[18/Apr/2024 00:07:56] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:08:27] "GET / HTTP/1.1" 200 860
[18/Apr/2024 00:08:29] "GET /polls/ HTTP/1.1" 200 1934
[18/Apr/2024 00:08:31] "GET /polls/11/ HTTP/1.1" 200 1655
[18/Apr/2024 00:08:33] "POST /polls/11/vote/ HTTP/1.1" 302 0
[18/Apr/2024 00:08:33] "GET /polls/11/results/ HTTP/1.1" 200 1239
The Poll Mall
Department
CSE 1 vote
ECE 0 votes
MECH 0 votes
EEE 0 votes
Back To Polls
Vote again?

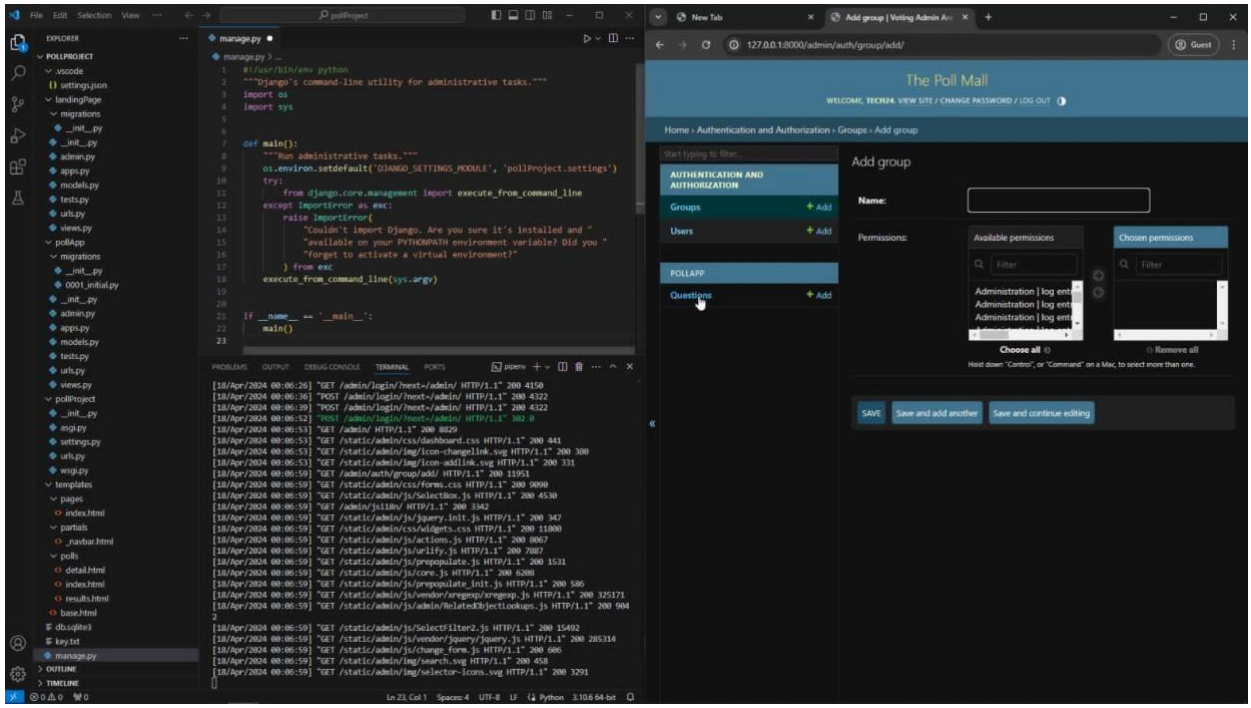

manage.py
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
"""Run administrative tasks."""
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'pollProject.settings')
try:
from django.core.management import execute_from_command_line
except ImportError as exc:
raise ImportError(
"Couldn't import Django. Are you sure it's installed and "
"available on your PYTHONPATH environment variable? Did you "
"forget to activate a virtual environment?"
) from exc
execute_from_command_line(sys.argv)
if __name__ == '__main__':
main()
[18/Apr/2024 00:06:26] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 4150
[18/Apr/2024 00:06:36] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4322
[18/Apr/2024 00:06:39] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4322
[18/Apr/2024 00:06:52] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[18/Apr/2024 00:06:53] "GET /admin/ HTTP/1.1" 200 8829
[18/Apr/2024 00:06:53] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 441
[18/Apr/2024 00:06:53] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
[18/Apr/2024 00:06:53] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
[18/Apr/2024 00:06:59] "GET /admin/auth/group/add/ HTTP/1.1" 200 11951
[18/Apr/2024 00:06:59] "GET /static/admin/css/forms.css HTTP/1.1" 200 9090
[18/Apr/2024 00:06:59] "GET /static/admin/js/SelectBox.js HTTP/1.1" 200 4530
[18/Apr/2024 00:06:59] "GET /admin/jsi18n/ HTTP/1.1" 200 3342
[18/Apr/2024 00:06:59] "GET /static/admin/js/jquery.init.js HTTP/1.1" 200 347
[18/Apr/2024 00:06:59] "GET /static/admin/css/widgets.css HTTP/1.1" 200 11800
[18/Apr/2024 00:06:59] "GET /static/admin/js/actions.js HTTP/1.1" 200 8067
[18/Apr/2024 00:06:59] "GET /static/admin/js/urlify.js HTTP/1.1" 200 7887
[18/Apr/2024 00:06:59] "GET /static/admin/js/prepopulate.js HTTP/1.1" 200 1531
[18/Apr/2024 00:06:59] "GET /static/admin/js/core.js HTTP/1.1" 200 6208
[18/Apr/2024 00:06:59] "GET /static/admin/js/prepopulate_init.js HTTP/1.1" 200 586
[18/Apr/2024 00:06:59] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 200 325171
[18/Apr/2024 00:06:59] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 200 804
2
[18/Apr/2024 00:06:59] "GET /static/admin/js/SelectFilter2.js HTTP/1.1" 200 15492
[18/Apr/2024 00:06:59] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 200 285314
[18/Apr/2024 00:06:59] "GET /static/admin/js/change_form.js HTTP/1.1" 200 686
[18/Apr/2024 00:06:59] "GET /static/admin/img/search.svg HTTP/1.1" 200 458
[18/Apr/2024 00:06:59] "GET /static/admin/img/selector-icons.svg HTTP/1.1" 200 3291
The Poll Mall
WELCOME, TECH24. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Authentication and Authorization › Groups › Add group
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
POLLAPP
Questions + Add
Add group
Name:
Permissions:
Available permissions
Chosen permissions
Choose all
Remove all
Hold down "Control", or "Command" on a Mac, to select more than one.
SAVE
Save and add another
Save and continue editing

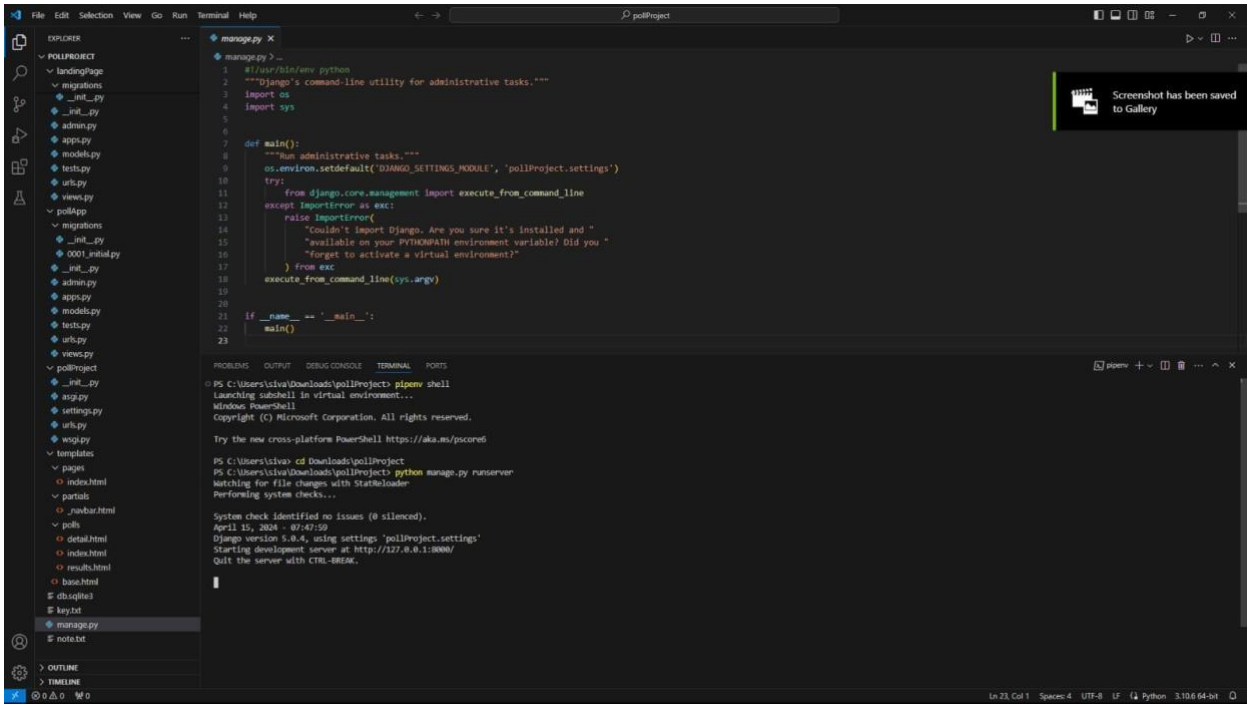

Poll Mall
127.0.0.1:8000/polls/6/results/
Guest
The Poll Mall
Screenshot has been saved to Gallery
Favorite Color
Red
0 votes
Blue
2 votes
Green
0 votes
Yellow
0 votes
Other
0 votes
Back To Polls
Vote again?

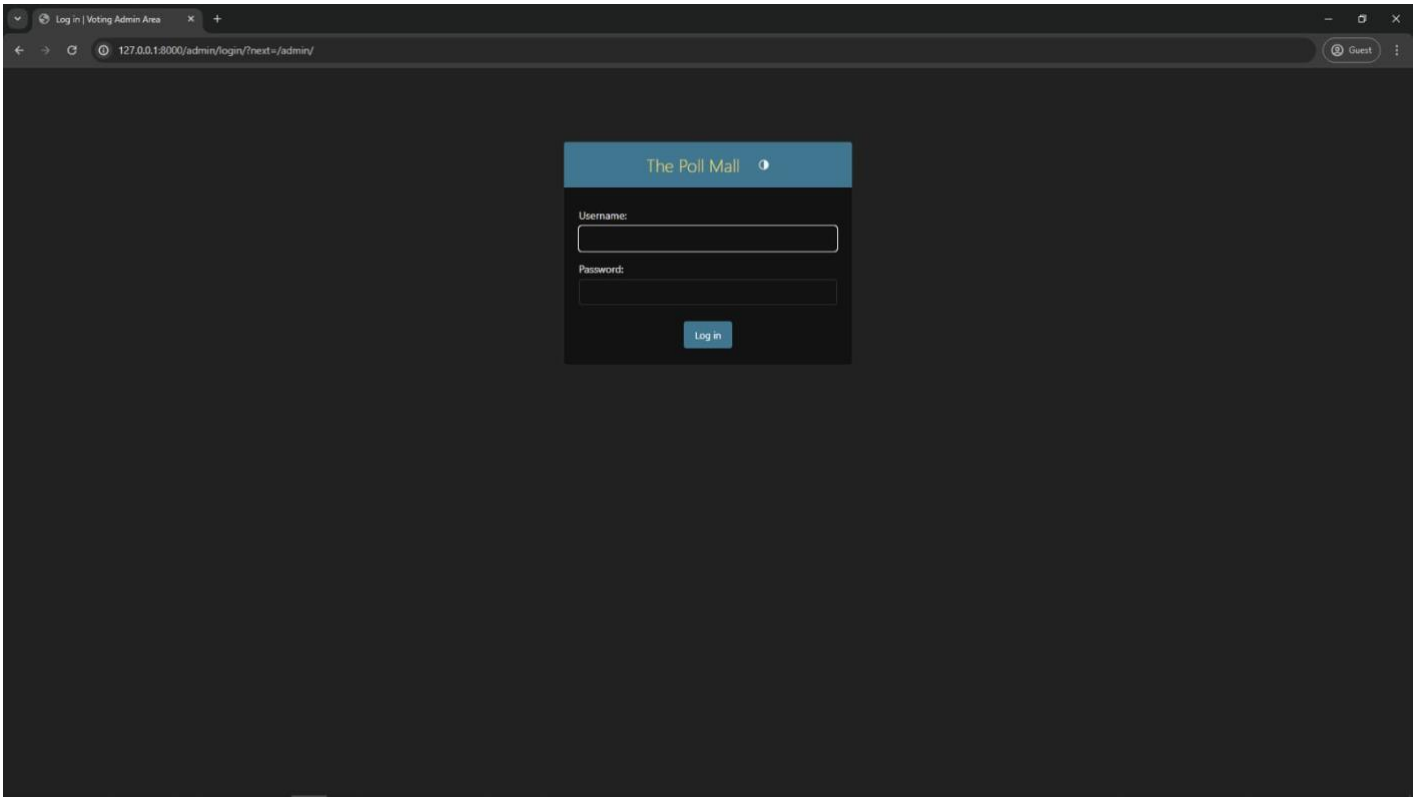

manage.py
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
"""Run administrative tasks."""
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'pollProject.settings')
try:
from django.core.management import execute_from_command_line
except ImportError as exc:
raise ImportError(
"Couldn't import Django. Are you sure it's installed and "
"available on your PYTHONPATH environment variable? Did you "
"forget to activate a virtual environment?"
) from exc
execute_from_command_line(sys.argv)
if __name__ == '__main__':
main()
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS C:\Users\siva\Downloads\pollProject> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 18, 2024 - 00:05:42
Django version 5.0.4, using settings 'pollProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[18/Apr/2024 00:05:58] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4320
[18/Apr/2024 00:06:01] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4320
[18/Apr/2024 00:06:10] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4320
[18/Apr/2024 00:06:12] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4320
[18/Apr/2024 00:06:17] "GET / HTTP/1.1" 200 860
[18/Apr/2024 00:06:20] "GET /polls/ HTTP/1.1" 200 1936
[18/Apr/2024 00:06:26] "GET /admin/ HTTP/1.1" 302 0
[18/Apr/2024 00:06:26] "GET /admin/login/?next=/admin/ HTTP/1.1" 200 4150
[18/Apr/2024 00:06:36] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4322
[18/Apr/2024 00:06:39] "POST /admin/login/?next=/admin/ HTTP/1.1" 200 4322
[18/Apr/2024 00:06:52] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[18/Apr/2024 00:06:53] "GET /admin/ HTTP/1.1" 200 8029
[18/Apr/2024 00:06:53] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 441
[18/Apr/2024 00:06:53] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
[18/Apr/2024 00:06:53] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
127.0.0.1:8000/admin/
The Poll Mall
WELCOME, TECH24. VIEW SITE / CHANGE PASSWORD / LOG OUT
Welcome to our Voting Admin Area
AUTHENTICATION AND AUTHORIZATION
Groups
Add
Change
Users
Add
Change
POLLAPP
Questions
Add
Change
Recent actions
My actions
Preferred Mode of Transportation
Question
What is the best JS Framework ?
Question
Which planet is closest to the Sun?
Question
Favorite Food
Question
Ideal Vacation Destination
Question
Preferred Animal
Question
Favorite Color
Question

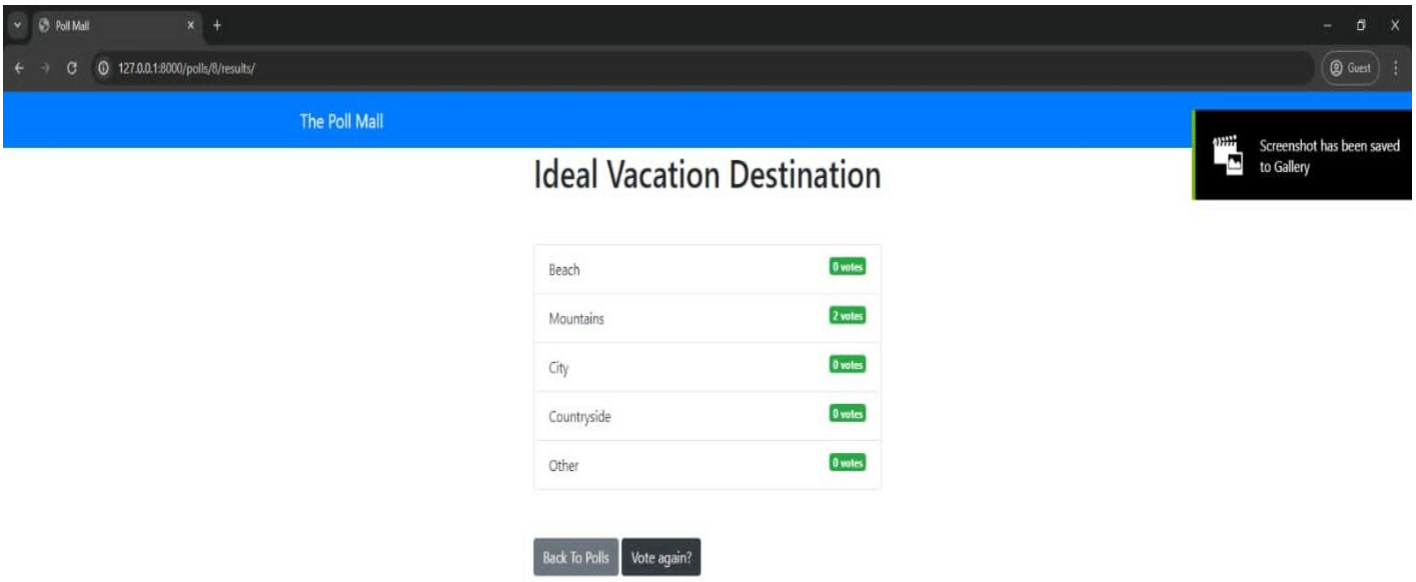

You're browsing as a Guest
Pages you view in this window won't appear in the browser history and they won't leave other traces, like cookies, on the computer after you close all open Guest windows. Any files you download will be preserved, however.
Learn more

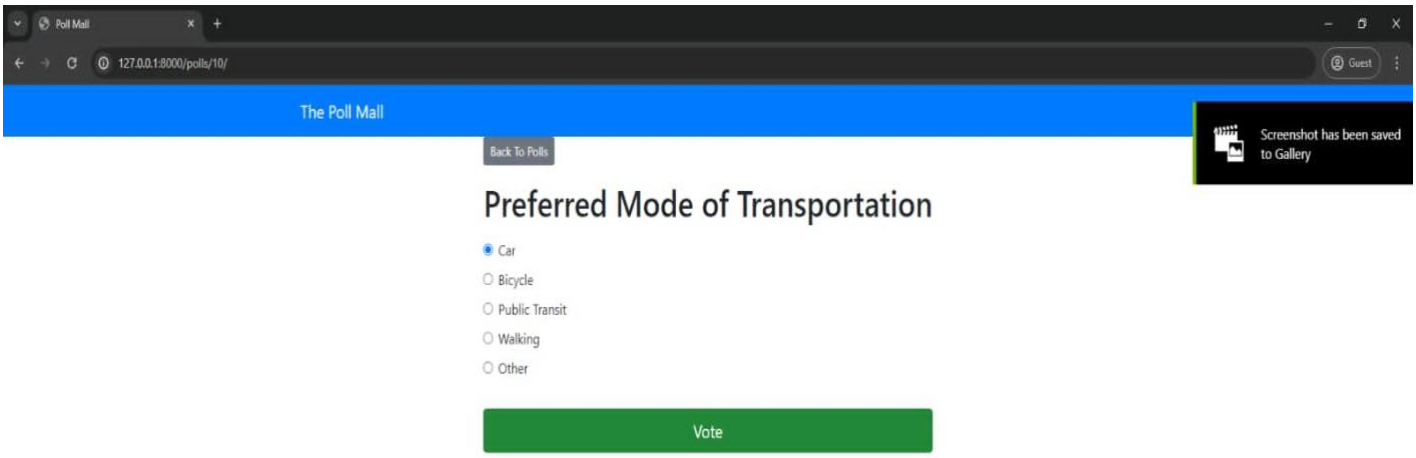

The Poll Mall
Select question to change
ADD QUESTION
Preferred Mode of Transportation
Favorite Food
Ideal Vacation Destination
Preferred Animal
Favorite Color
Which planet is closest to the Sun?
What is the best JS Framework ?
What is your favorite food ?
8 questions


The Poll Mall
Add group
Name:
Permissions:
Available permissions
Chosen permissions
SAVE
Save and add another
Save and continue editing


Screenshot has been saved to Gallery


Log in | Voting Admin Area
127.0.0.1:8000/admin/login/?next=/admin/
Guest
The Poll Mall
Username:
Password:
Log in


Poll Mall
127.0.0.1:8000/polls/8/results/
Guest
The Poll Mall
Screenshot has been saved to Gallery
Ideal Vacation Destination
Beach
0 votes
Mountains
2 votes
City
0 votes
Countryside
0 votes
Other
0 votes
Back To Polls
Vote again?


Poll Mall
127.0.0.1:8000/polls/10/
Guest
The Poll Mall
Screenshot has been saved to Gallery
Back To Polls
Preferred Mode of Transportation
Car
Bicycle
Public Transit
Walking
Other
Vote