# Django Third Assignment

##### This Assignment will cover: Django Models, and GET/POST/PUT/DELETE request

##### Assignment includes web interface UI and backend development

Consider you have to Develop a login page for user who can give polls for various questions and later can view the scores and check results for the same. The website is divided into 4 categories Login, register, Profile Page and Home Page. User would first register to manage the following features for each category:<br />

TEMPLATE 1

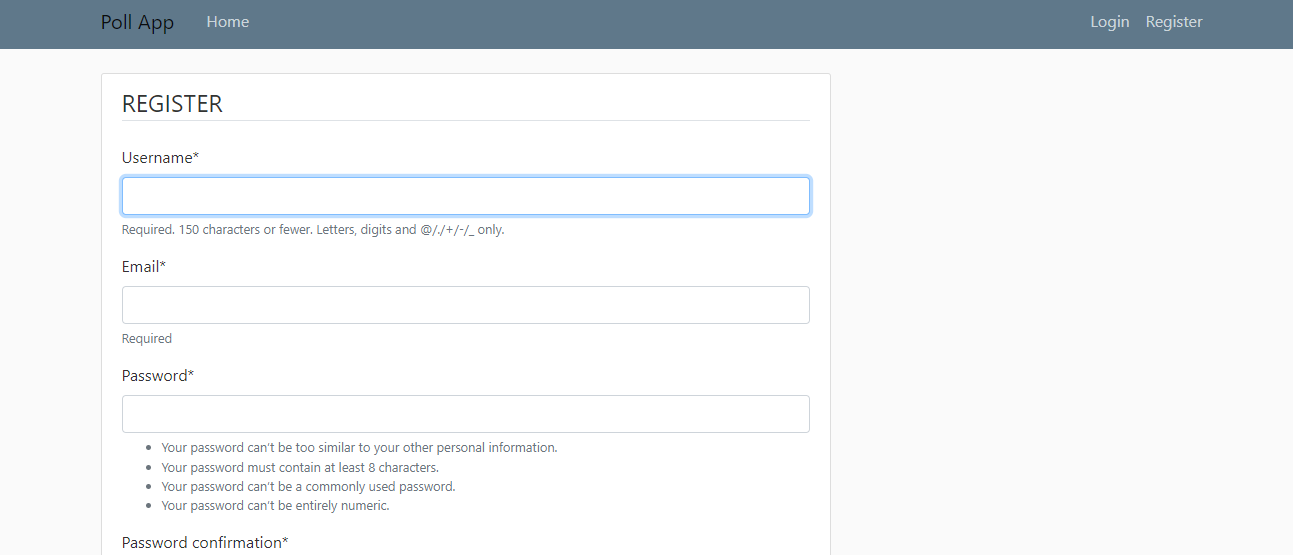
Register page<br />

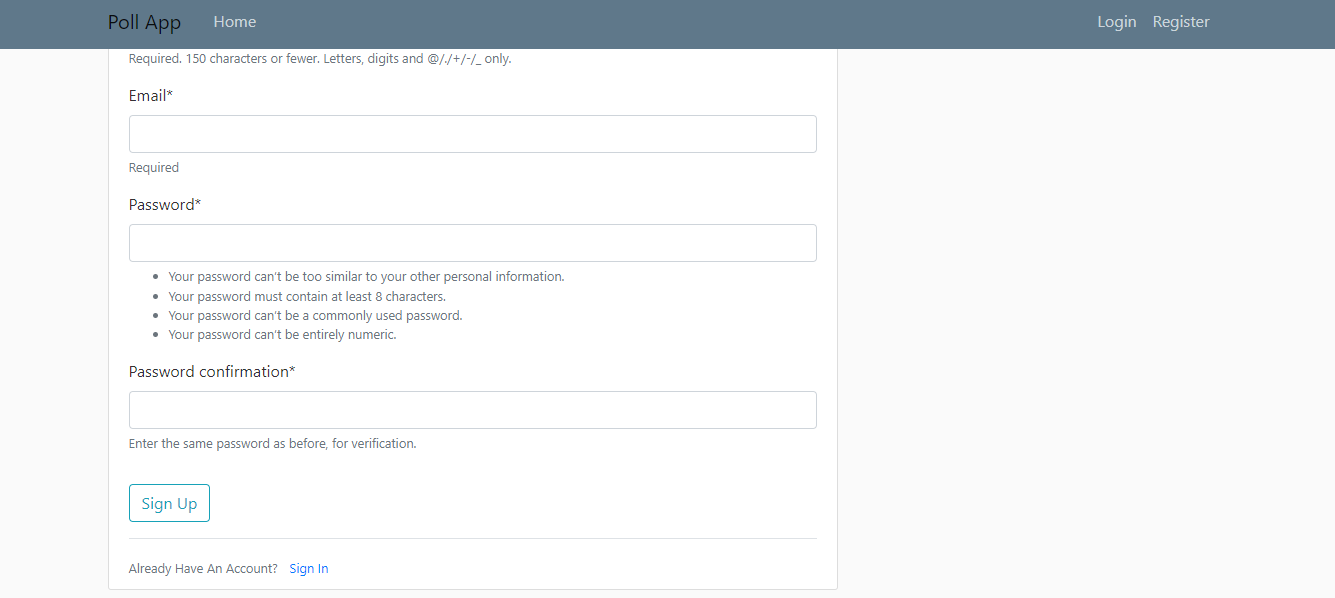
1. Name (name) -> String<br />

2. Email(email) -> String<br />

3. Password (\*) -> string<br />

4. Confirm Password (\*)-> Mix<br />





path('register/', user\_views.RegisterView.as\_view(), name='register')

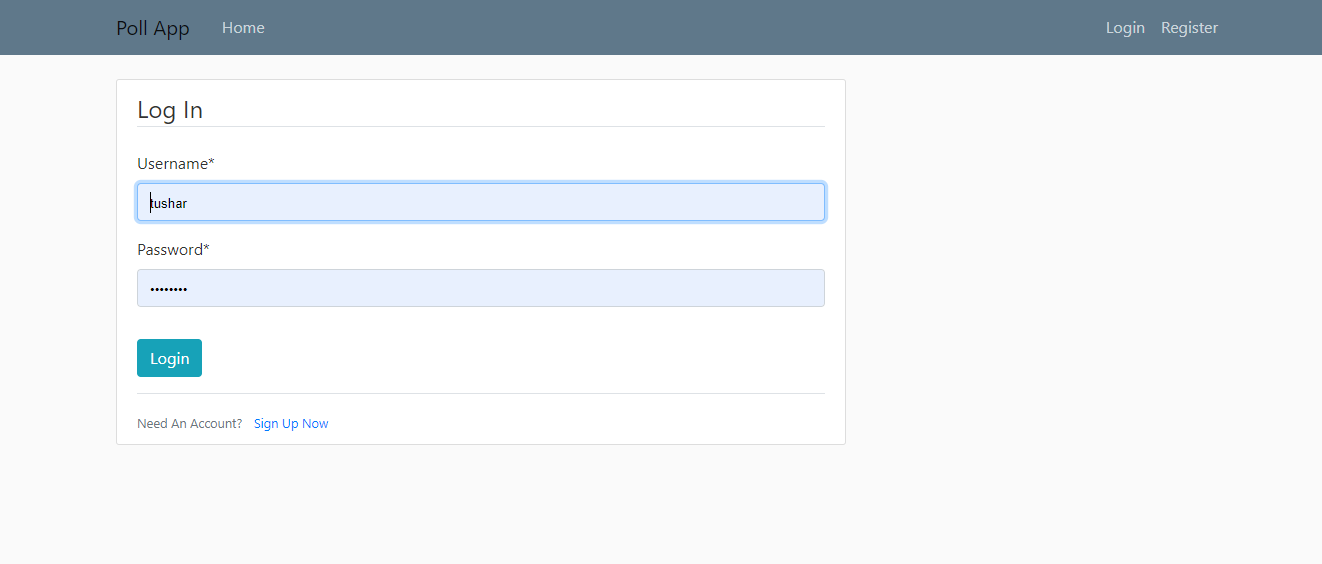
Login<br />

path('login/', auth\_views.LoginView.as\_view(template\_name='users/login.html'), name='login'),

1. Name (name) -> String<br />

2. Password(\*) -> String<br />

3. Login Button



TEMPLATE 2

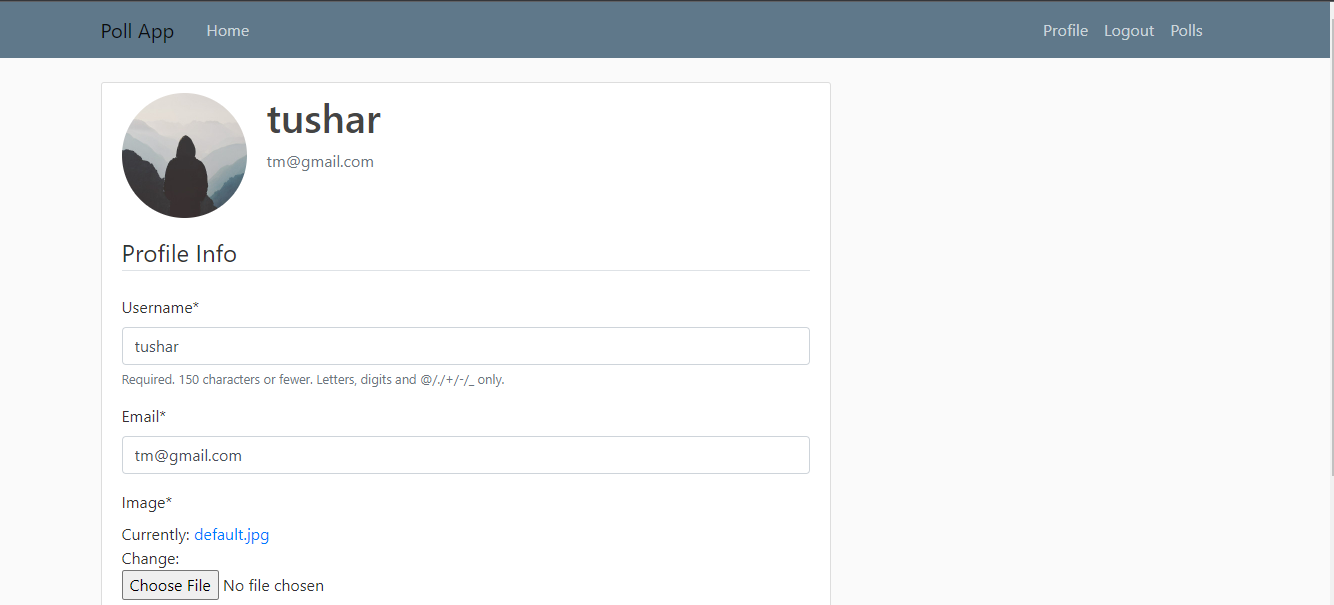
Profile<br />

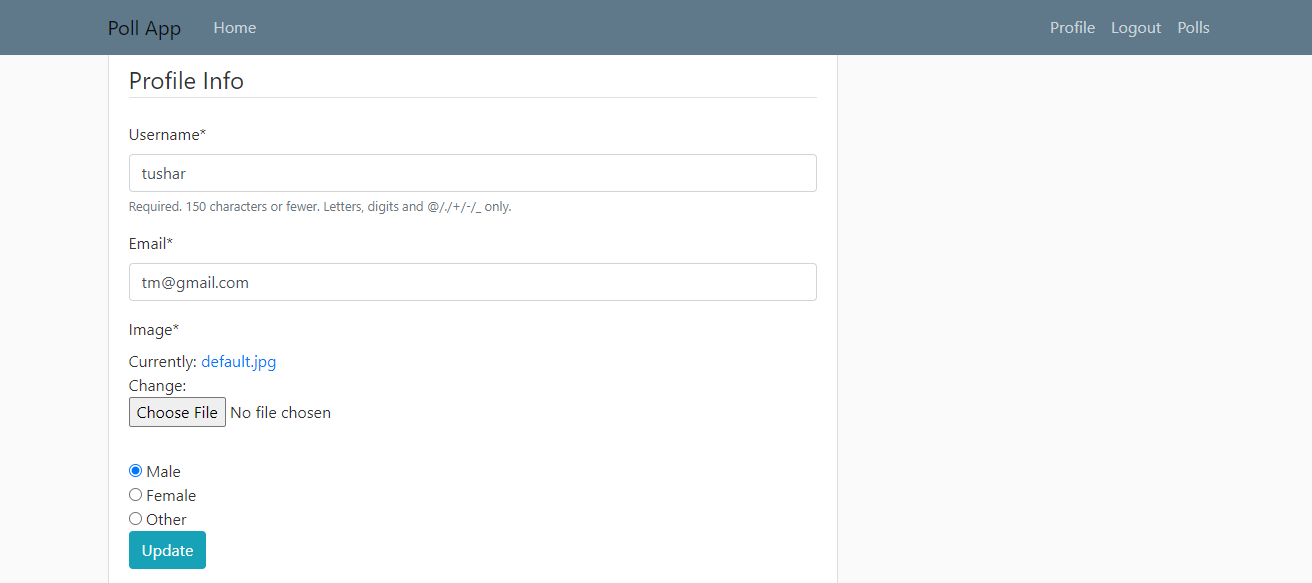
1. Name (name) -> String<br />

2. Email(email) -> String<br />

3. Image(\*) -> jpg<br />

4. Gender (\*)-> Mix<br />

5. Update button

TEMPLATE 3

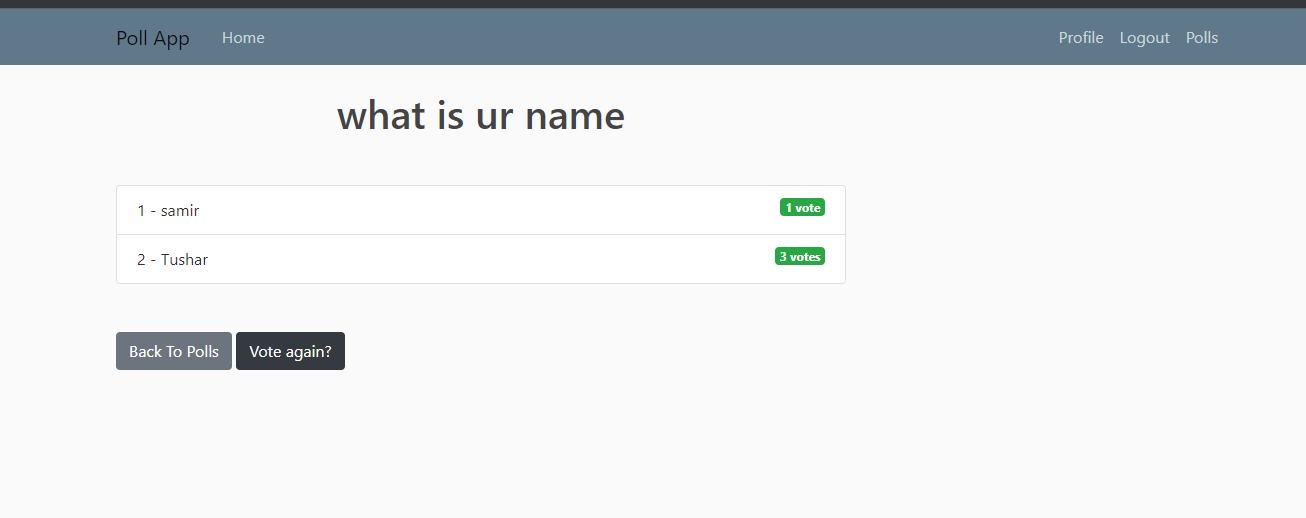
Polls<br />

1. Question <br />

2. Multiple choices (Button) <br />

- Vote now and Check results button

1. Vote button (\*) -> jpg<br />
2. Multiple options for the user to click for the answer
3. Back to Polls button to redirect to home page.



\*\*Only authenticated users can access the home page\*\*

Thus , after Login , any User can further view and update their Profile page , same needs to be shown on the Home page as well

### Assignment:

1. Create a Django app named login\_user\_app<br />

2. Create models in login\_user\_app for ur Templates-> Home, Profile and Register , login , logout ,Polls with features mentioned above<br />

3. Do not change Database changes in settings.py<br />

4. Run makeMigrations command<br />

5. Finally, run migrate command to create tables in the database<br />

6. Create views to Get and POST Data for the users , Profile, Polls Models Hint: Use ORM<br />

7. For Views and Urls More Details are given below <br />

##### Some Guide for urls.py:<br />

1. Name in Django URL -> name is used for accessing that URL from your Django / Python code.<br />

2. For example you have <http://127.0.0.1:8000/polls/> for ur Polls app<br />

 path('polls/', views.index, name='index')

3. In point 2, "index" is the name of this URL

4. URL namespaces -> URL namespaces allow you to uniquely reverse named URL patterns even if different applications use the same URL names.

5. For this assignment's test cases we have used some predefined names and namespace

6. Those names are defined below in the problem statement describes as "Django Url name"

7. Please make sure you use those names in Django URLs or else your assignment's test will fail

Summary: \*\*Namespace\*\* is "polls" set in app/urls.py file and \*\*name\*\* is to be set for each URL in app/urls which is stated in the problem statement

1. Make sure User can get question\_id directly from the url when it is passed in the Headers

Eg '<int:question\_id>/'

#### Views: Expected I/O

\*\*Part 1: GET and POST request for fetching POLLS\*\*

\*\*Django URL name: polls\*\*<br />

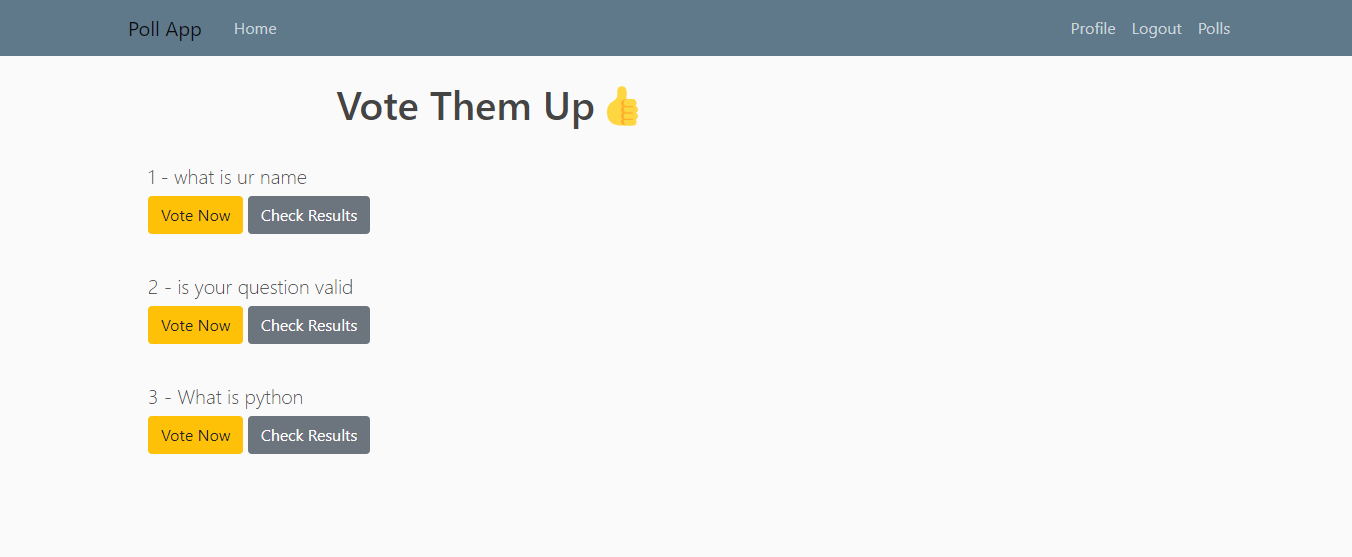
\*\*Actions\*\*:<br />

1. GET - get list of all questions mentioned in the poll<br />

2. POST- Vote Now<br />

3. POST- Check Results<br />

<br />



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

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

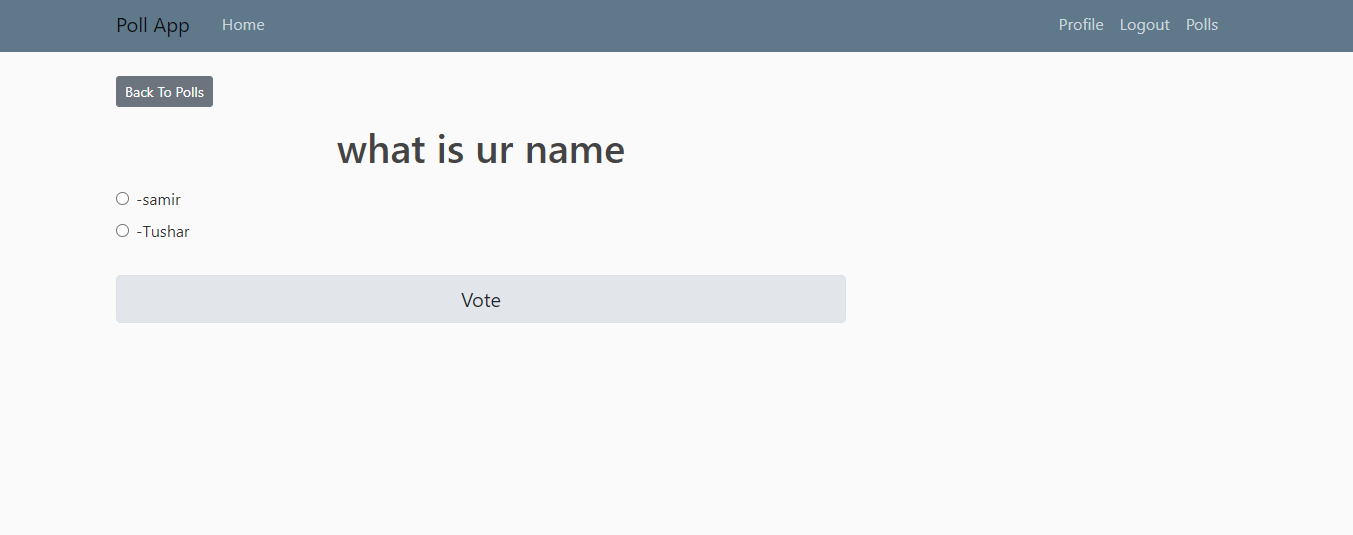
]

2. POST

Response Data format Button click on url index to question\_id

Eg <http://127.0.0.1:8000/1/>

####Here user can post their answers for the questions and vote for it

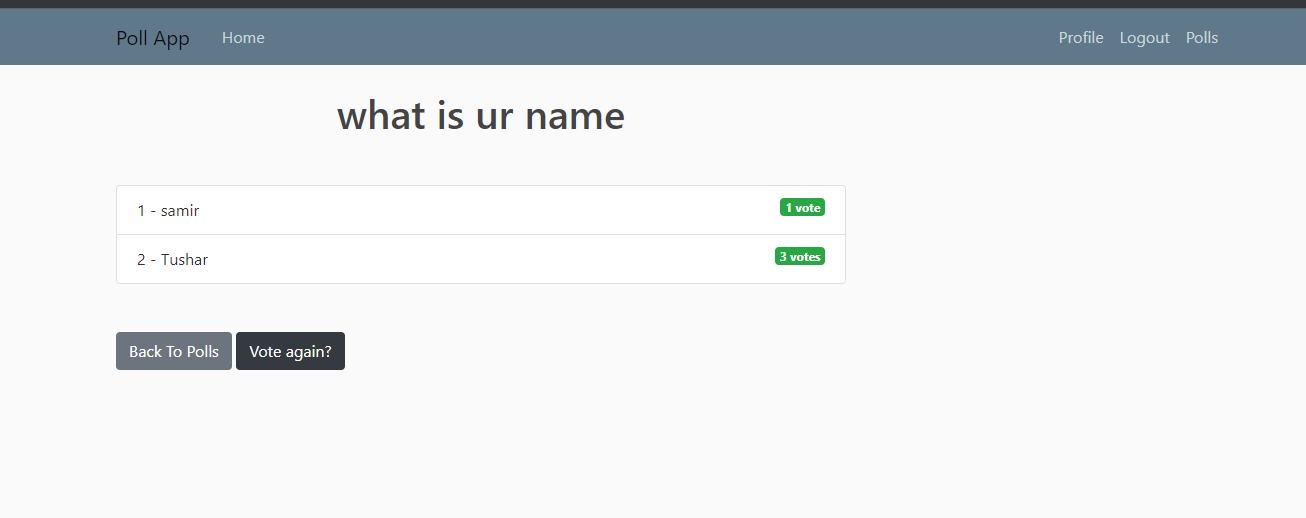


3 GET

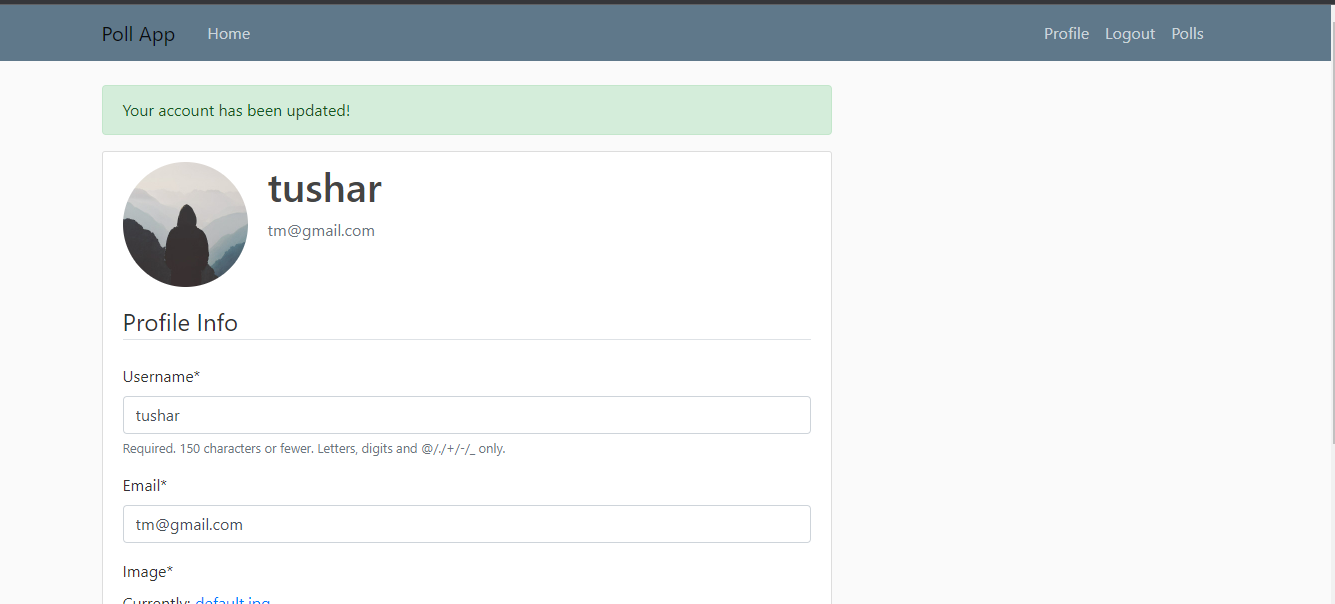
After that User can check results for the number of posts he created

And we will provide multiple options to user and allow him to vote again

[ HINT - no need to blacklist any IP address so any user can post as many votes as he wants .]



\*\*Part 2: UPDATE PROFILE<br />\*\*



\*\*Django URL name: profile\*\*<br />

No parameter<br />

\*\*Actions\*\*

1. PUT- Update from list of all User info<br />

2. POST - Add a new email to the Profile<br />

<br />

1. GET:

Response Data format

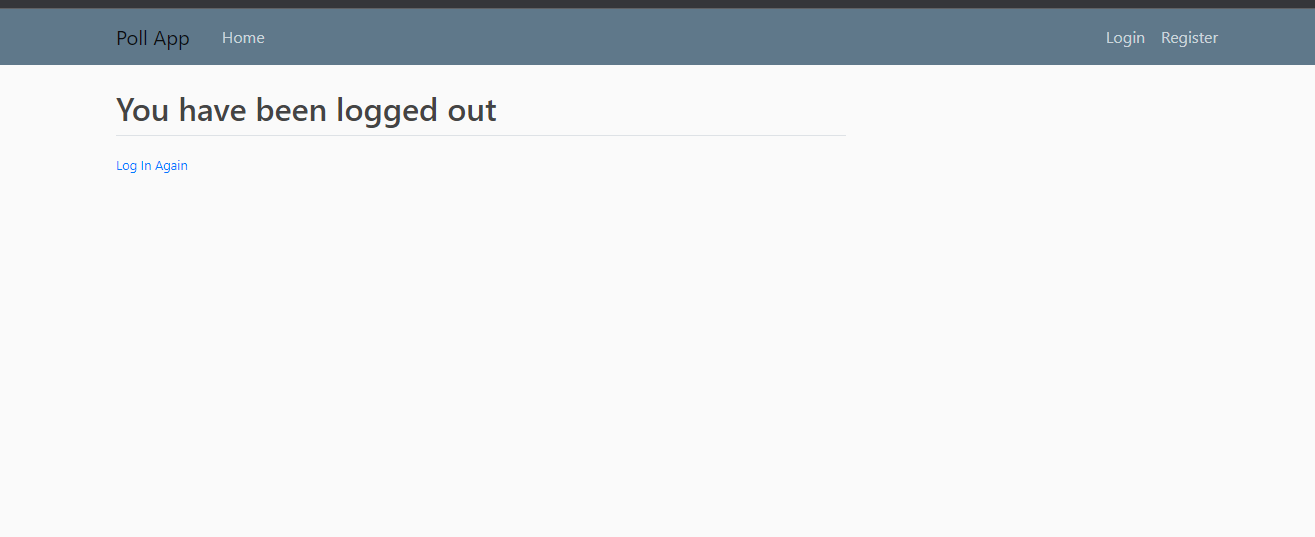
If no data is present

/ Response -> return Home page

\*\*Part 3: Logout<br />\*\*

<br />

<h3>Once logout , user cant login see his profile if he cant login back again</h3>



\*\*Django URL name: logout\*\*<br />

No parameter<br />

path('logout/', auth\_views.LogoutView.as\_view(template\_name='users/logout.html'), name='logout'),

\*\*Actions\*\*

1. GET - redirect to main screen <br />

2. POST -here user can again login to view the home page or register a new user <br />

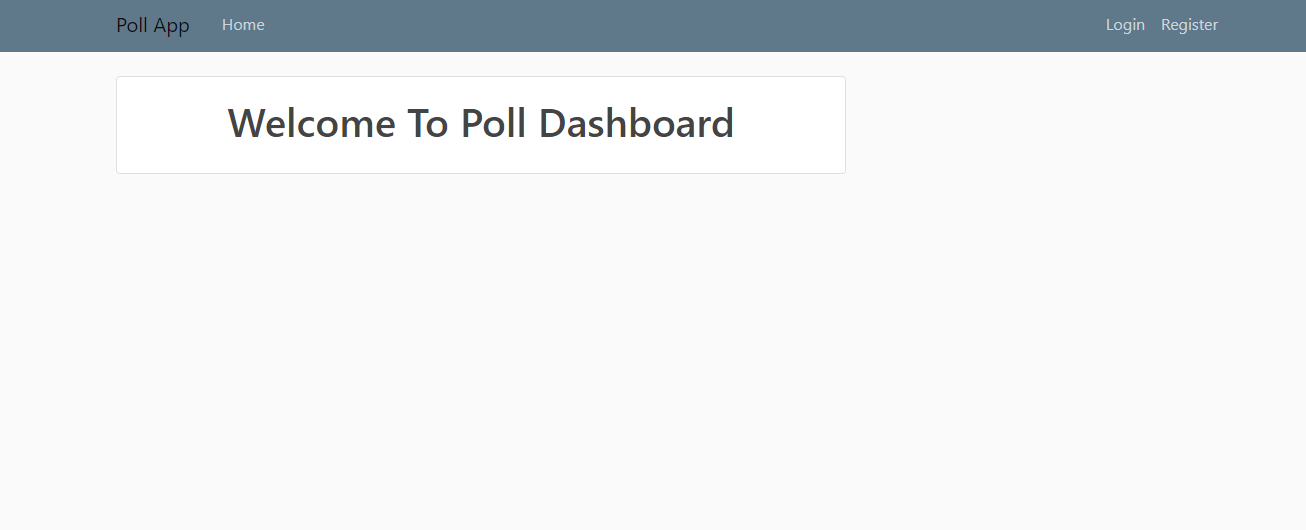
<br />

1. POST:

Response Data format /home

Here user can now check and login back to check the landing pages

1. Home page



## Installation Steps

1. Open up your Terminal / Command Line

2. git clone the repository

3. cd into the directory of the step (the one you just git cloned)

4. make sure you have python3 installed

5. Install virtualenv by following the steps

```

Mac

python3 -m pip install --user virtualenv

python3 -m venv env

source env/bin/activate

Windows

py -m pip install --user virtualenv

py -m venv env

.\env\Scripts\activate

```

6. Run

```

python -m pip install -r requirements/requirements.txt

```

6. No Database setup needs to be done. Do not edit Database settings in settings.py

7. If everything runs fine till here, create the application using the command

```

python manage.py startapp apps

```

Now Create File app/urls.py and add lines

```

from django.urls import path

from . import views

urlpatterns = []

```

8. Run

```

python manage.py makemigrations

```

9. Now Run the Server using command

```

python manage.py runserver

```

10. Great! Start Coding the assignment for above usecase

11. After Finishing the assignment, you can test them locally using command

```

coverage run --source='.' manage.py test --no-input

```

1. Push the code into the repo using git, Check Pull Requests Tab.
2. Check if all the tests are Passed
3. Make sure your Django version is above 3 or 4.0