

Borda Count Final Report

Team Members and Roles

Venkata Surya Naga Manoj Sree Harsha Vuppuluri (**Product Owner**)

Manisha Bachu

Harika Gumudavally

Venkata Sai Raghu Ram Reddy Sannapareddy

Revanth Reddy Male

Venkata Bhanu Teja Pallakonda

Mounika Kunduru

Scrum Masters (On rotation):

Iteration 0: Manisha Bachu, Harika Gumudavally

Iteration 1: Revanth Reddy, Raghu Ram Reddy Sannapareddy

Iteration 2: Venkata Bhanu Teja Pallakonda, Mounika Kunduru

Iteration 3: Manisha Bachu, Harika Gumudavally



**DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
TEXAS A&M UNIVERSITY, COLLEGE STATION**

Summary

In this project, we are tasked with developing a responsive website, which could be used for retrieving the optimal poll from the users. This website eliminates the need of sending bulk emails to a large list of users to gather their preferences and having to sort through each person to figure out the best preference.

The application implements an aggregation algorithm called Borda Count, which is a method used for aggregating individual preferences into a social preference. In this algorithm, the administrator creates an event, which is a choice among K objects (e.g. times for a meeting). The N different users can then rank the K objects, which is a strict ordering, and the website uses the Borda Count method to deliver the top choice, and this result is visible to the administrator.

The application is developed using the Django framework and we were able to do both manual and automated testing using jasmine. The story points of the project are monitored using Pivotal *tracker*. It is also deployed on Heroku and uploaded on GitHub. The stakeholders for this project are Dr. Korok Ray, Swapna and Dr. Duncan Walker.

User Stories Implemented

The website will be used by two categories of people : The administrator and the user.

User stories related to administrator:

Feature : Sign Up

As an administrator

So that I can register

I want to enter details and click sign up button

Feature: Login

As an administrator

So that I can log into the application and view the dashboard with created polls

I want to enter details in the login page

Feature: Dashboard View

As an administrator

So that I can view the details of the poll

I want to click on any poll on my dashboard

Feature : Create Poll

As an administrator

So that I can share the poll with users

I want to enter the poll options

User stories related to user:

Feature : Sign Up

As a user

So that I can register

I want to enter details and click sign up button

Feature: Login

As a user

So that I can login into the application and view the dashboard with polls needed to respond.

I want to enter details in the login page

Feature: Dashboard View

As a user

So that I can view the options of the poll and respond

I want to click on any poll on my dashboard

Feature : Respond to Poll

As a user

So that I can give my preferences

I want to select options available based on my priority

Iteration wise progress**Iteration 0:**

- Organized the team and developed a meeting schedule
- Had a meeting with the customer and acquired a better understanding of the project and its requirements.
- Came up with user stories for the application and also developed lofi UI mockups.
- Acquainted ourselves with the Django framework.

Iteration 1: (14 points)

- The initial setup of the project was done.
- The user interface was developed and user sign up and user login features were built.
- A single administrator was created in this project, and an admin login feature was built.
- The admin sign up which was initially thought was removed in this iteration.
- Testing of certain features such as user email validation, password validation and authentication validation, was done using Jasmine.

Iteration 2: (26 points)

- The Create Poll Page feature for the admin is implemented
- The Respond to poll feature for the user is implemented
- The Database Schema is designed
- The Drag and Drop for Poll options, is implemented on the UI
- The Drag and Drop for Poll options, is also implemented on the Backend
- The About Page and User Profile page were also built.
- User Dashboard and Admin Dashboard were also developed.

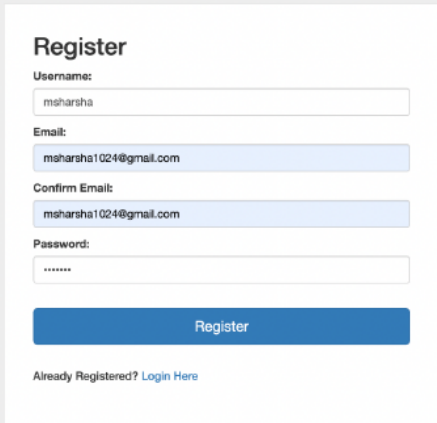
Iteration 3: (28 points)

- The Borda Count algorithm is designed and implemented.
- The feature in which admin could view the results of the polls was also implemented.
- The feature in which the user could see that their poll was answered after they submit the results.
- The testing of the poll created by the admin was implemented, and also submit object, user authorization and borda count validation was tested.
- The user can search the polls with a certain keyword, to find a poll, and the user can also update his profile.
- The user can submit their preferences in a certain order using the Drag and Drop poll option.

We had a final demo with Dr. Ray which includes all the above functionalities.

Development and Testing

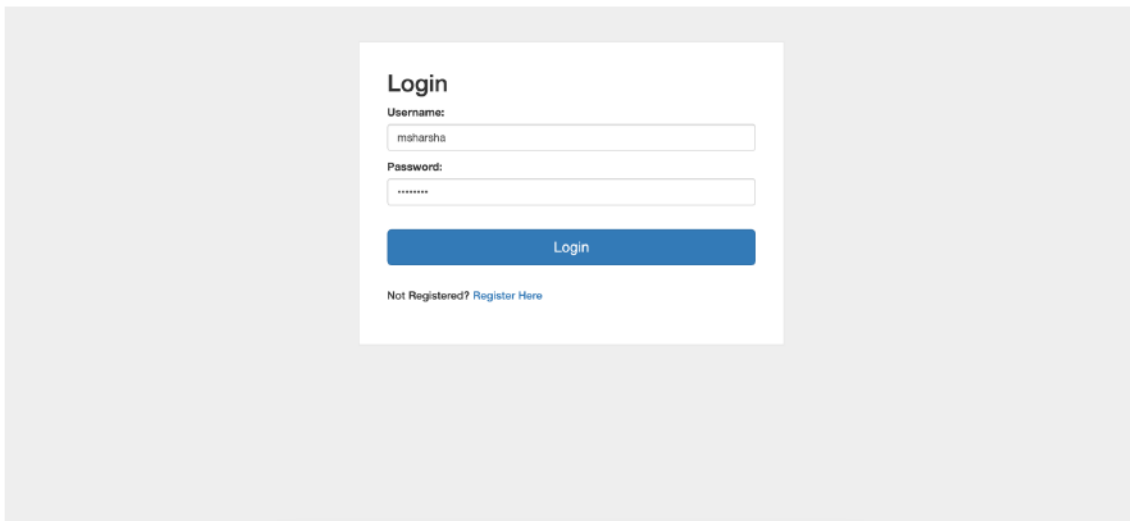
Registration Page



The screenshot displays a registration form titled "Register". It contains the following fields and elements:

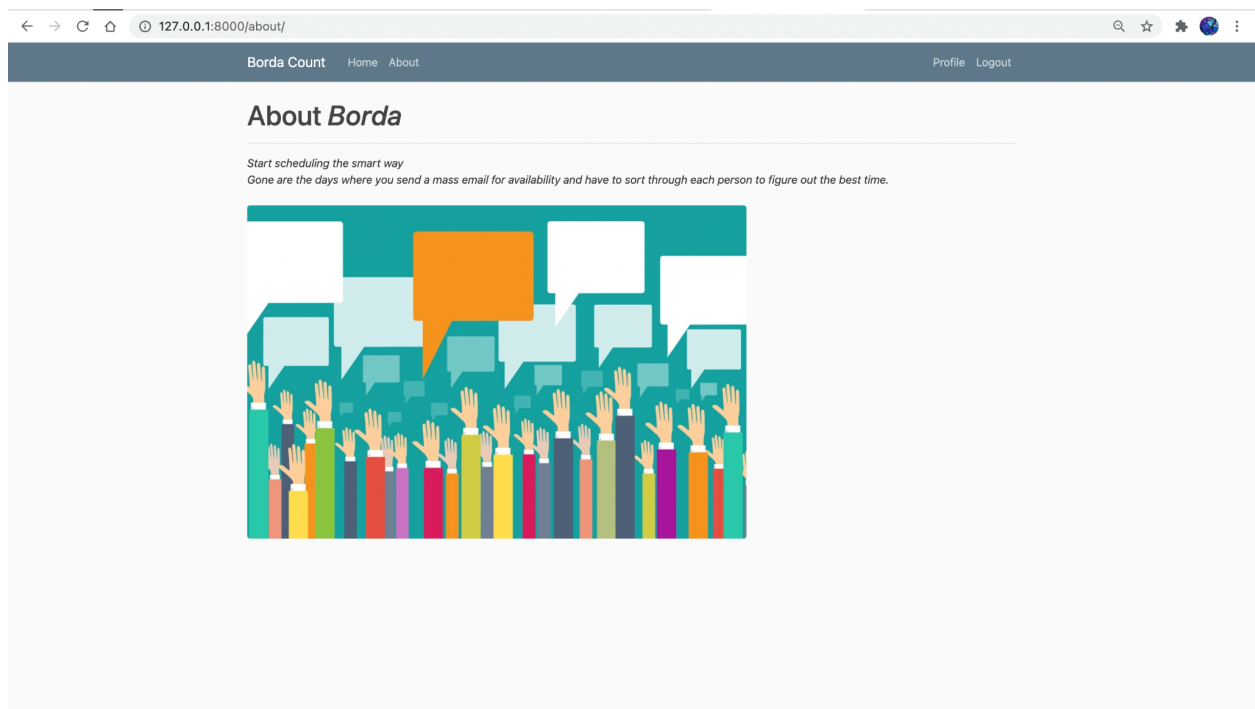
- Username:** A text input field containing the value "msharsha".
- Email:** A text input field containing the value "msharsha1024@gmail.com".
- Confirm Email:** A text input field containing the value "msharsha1024@gmail.com".
- Password:** A text input field with masked characters (dots).
- Register Button:** A blue button with the text "Register".
- Footer:** A link that says "Already Registered? [Login Here](#)".

Login Page



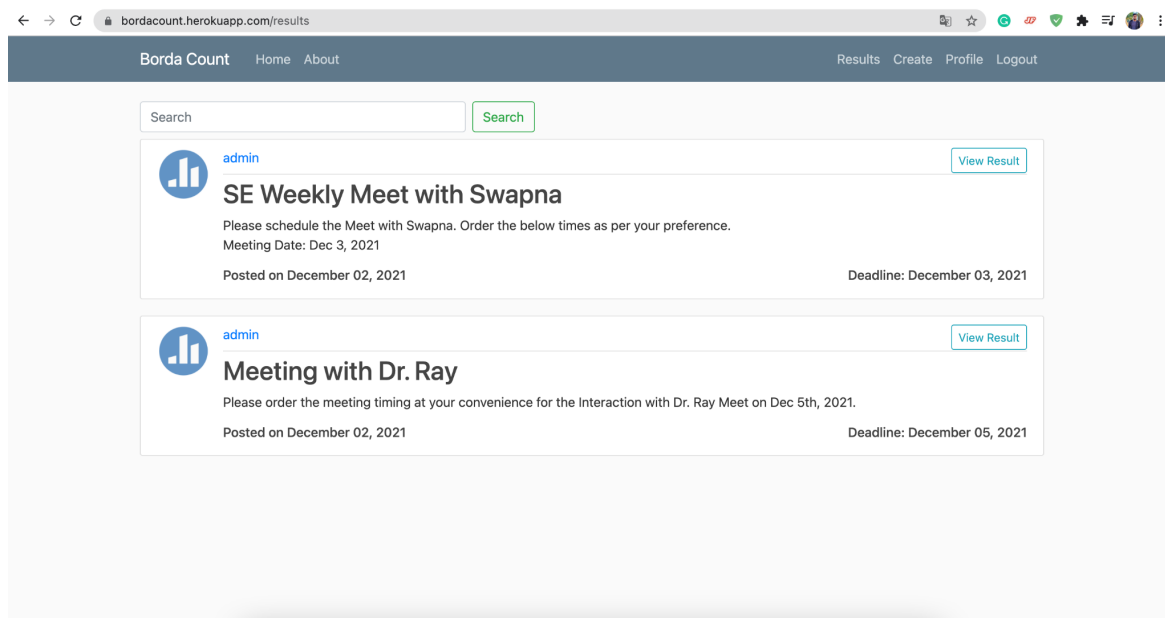
A screenshot of a web application's login page. The page has a light gray background. In the center, there is a white rectangular box containing the login form. The form is titled "Login" in bold black text. Below the title, there are two input fields: "Username:" with the text "maharsha" entered, and "Password:" with a masked password "*****". Below these fields is a blue button with the text "Login" in white. At the bottom of the form, there is a link that says "Not Registered? Register Here".

About Page

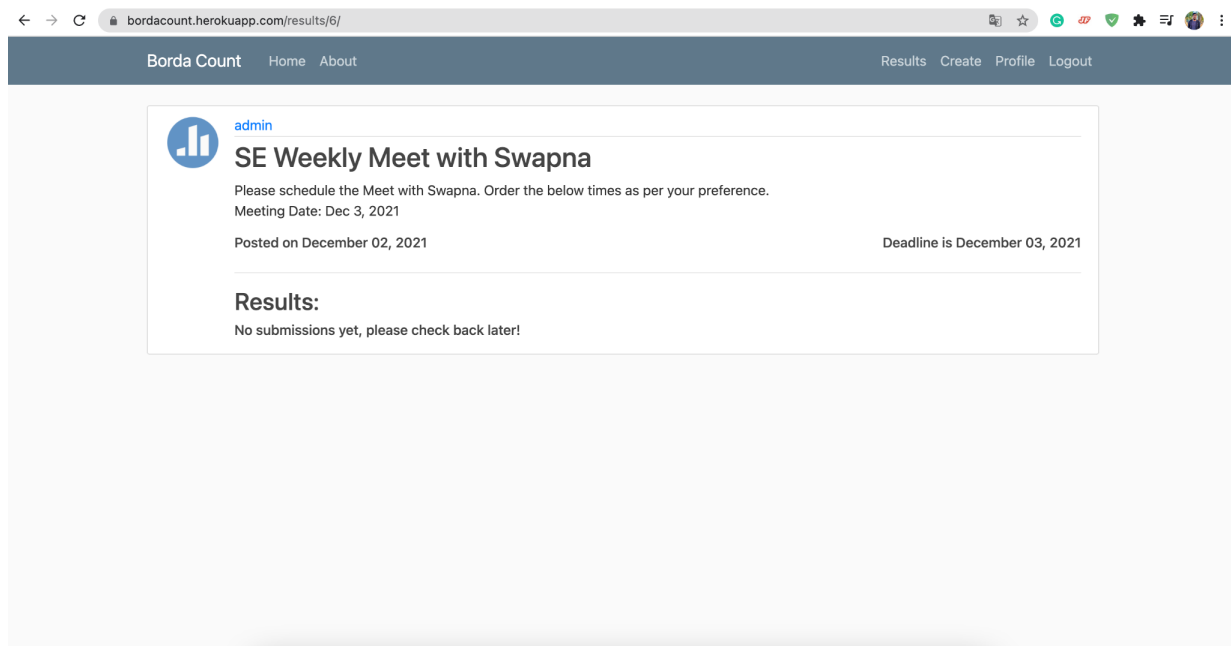


Admin flow

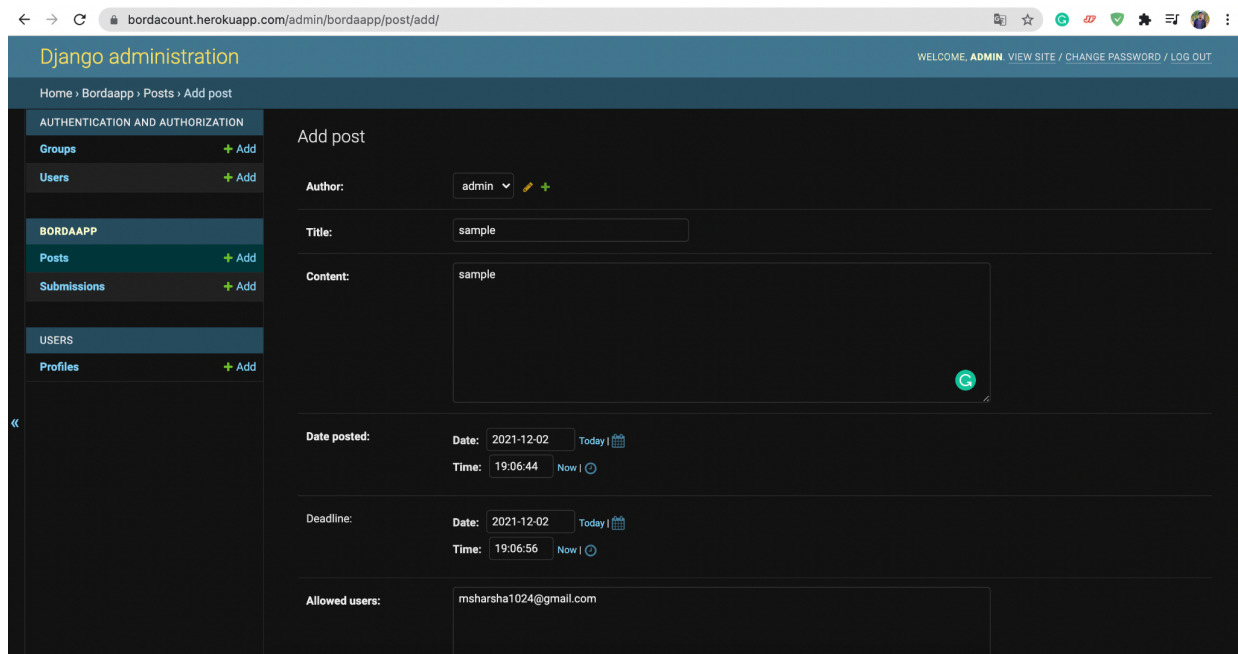
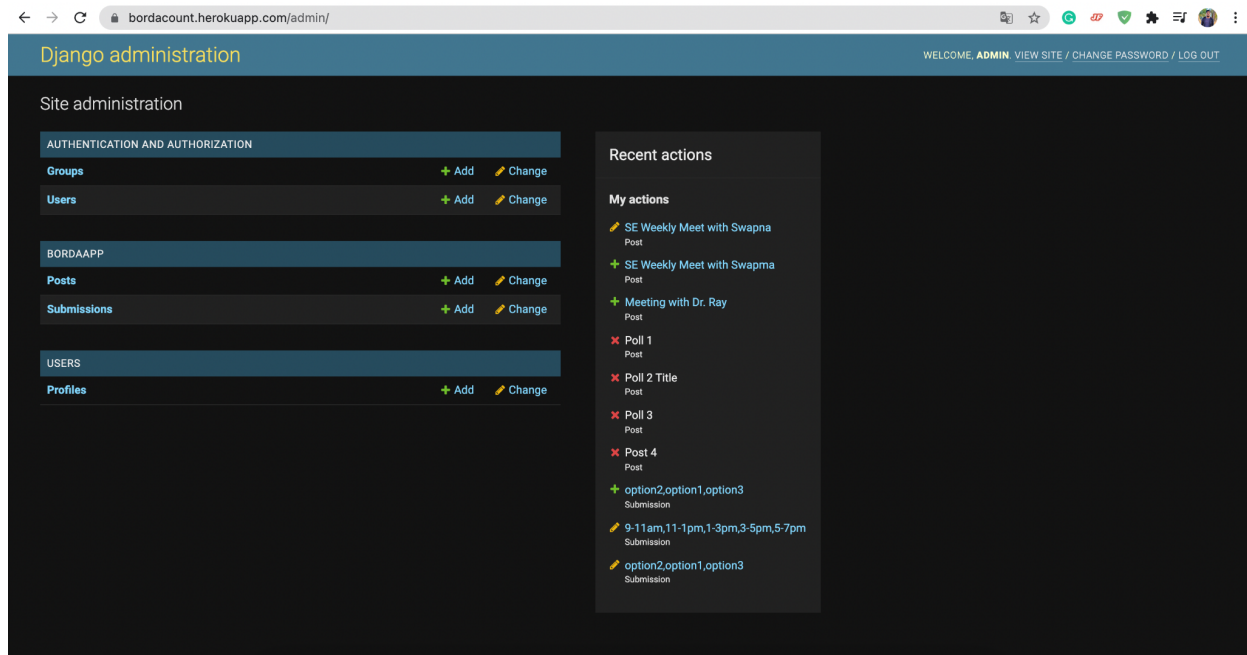
Once they login, upon clicking on **results** he can view all the polls that they created



Upon clicking on the **View Results**, they can view the results for any poll like below. Below case when the admin has created a poll and none of the user hasn't responded yet



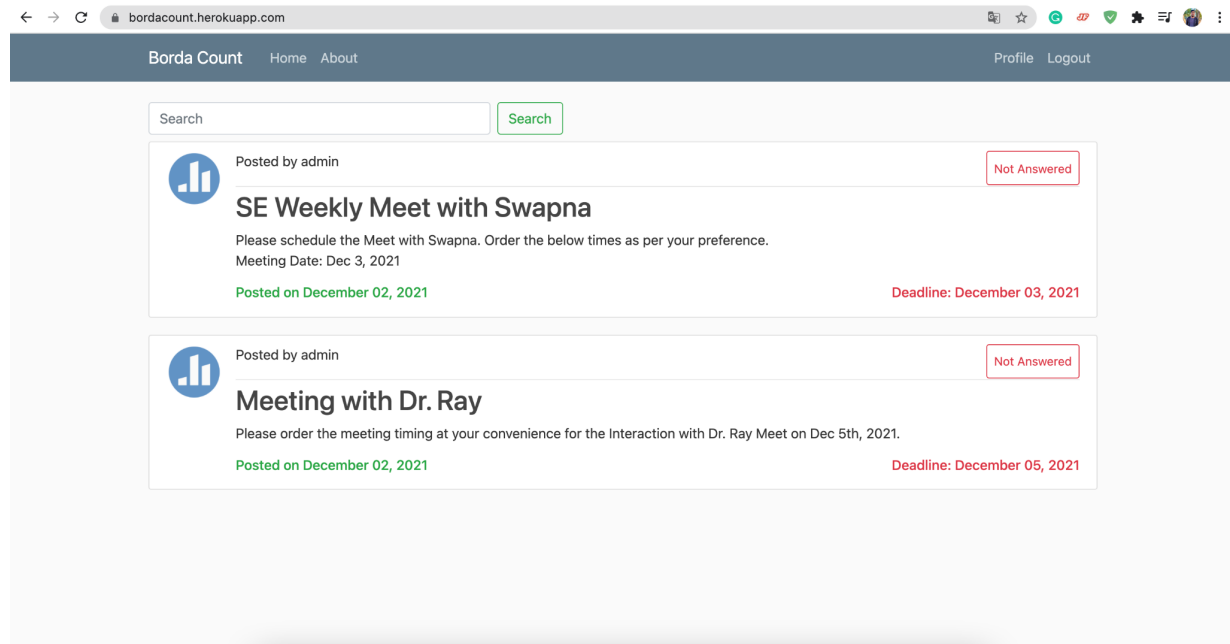
Also, they can create their own poll using **Create** button,



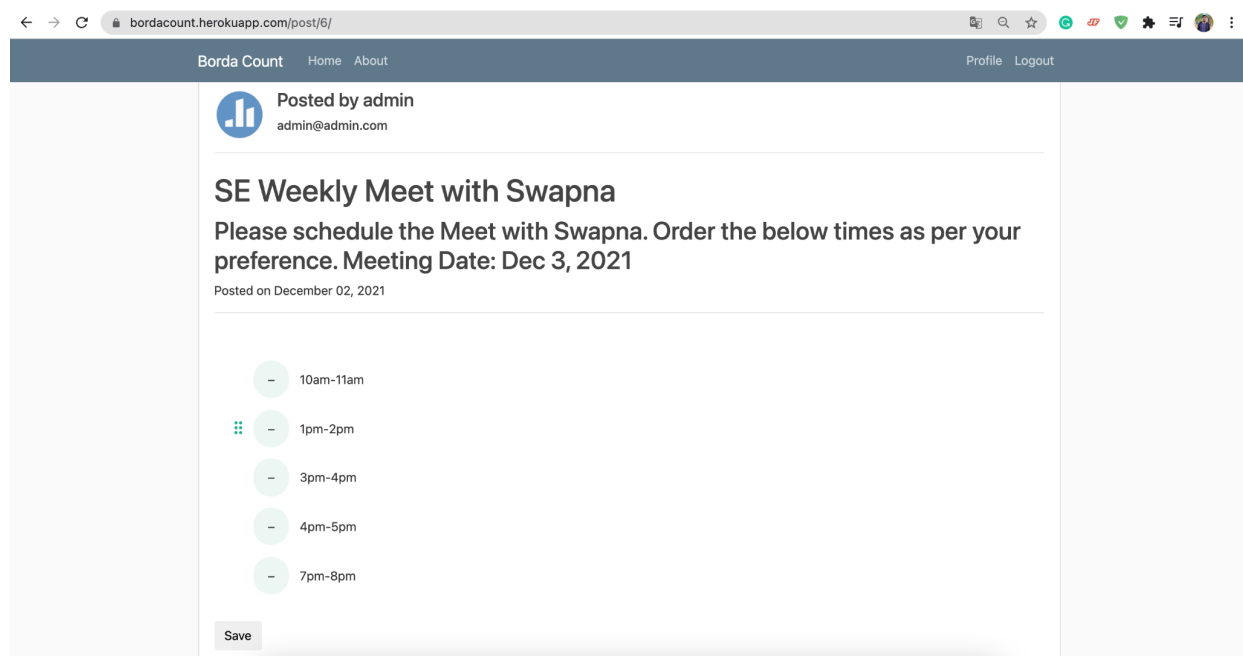
Upon clicking the save button, he can save the poll. At this stage, all the allowed users should be able to see the poll on their dashboard.

User flow

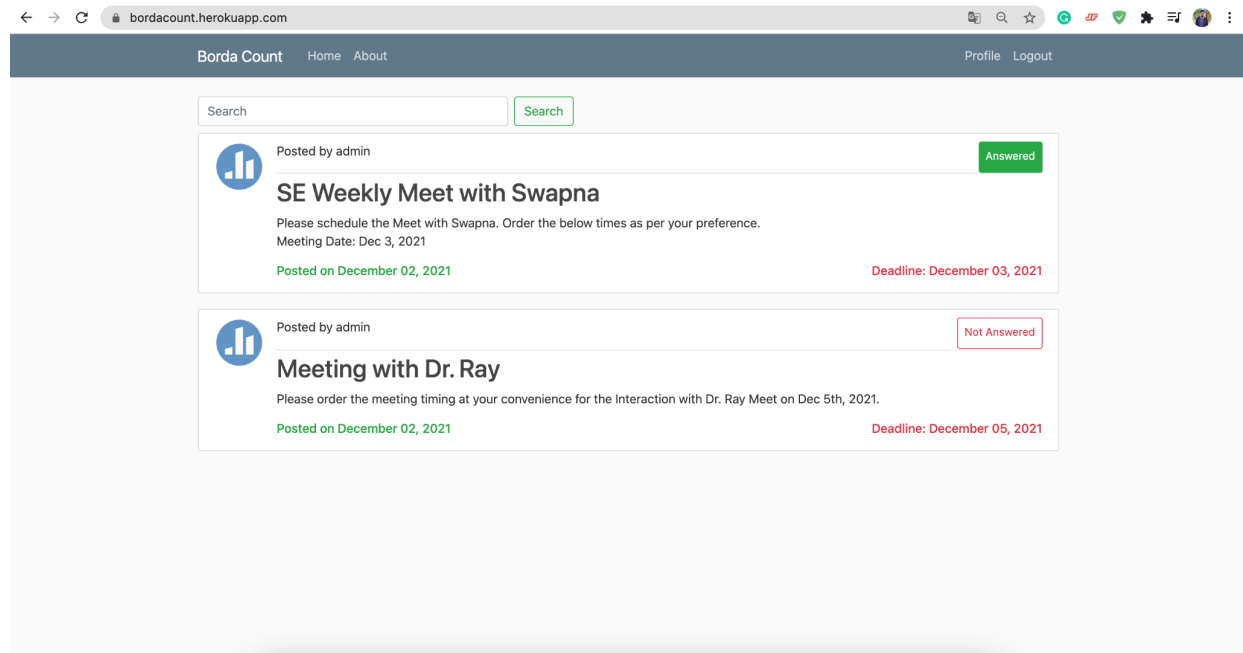
Below image shows the user dashboard with his polls to which they need to respond



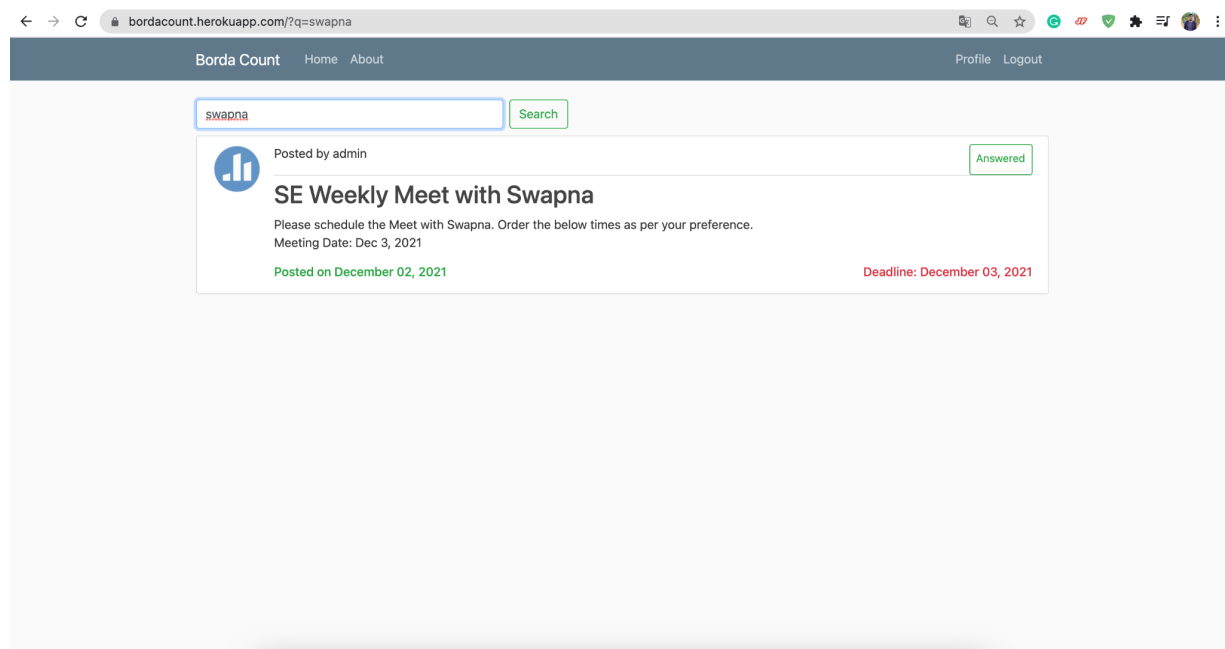
Users can select their own preference for any particular poll. User can drag up and down to finalize his priority order and thereby they can submit their preference by clicking on **save**



Once user submit their preference, they will be redirected to the home page where they can see that the poll has been **Answered**



Users can search their polls by any keyword



Users can update their profile

Borda Count Home About Profile Logout

user1
user1@user.com

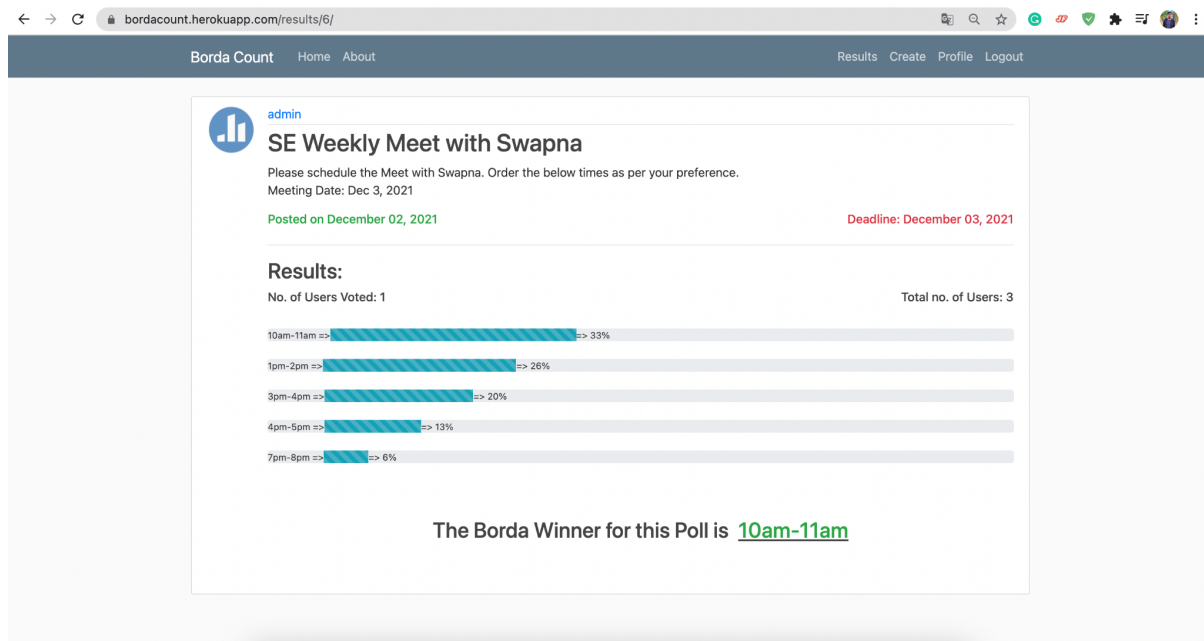
Profile Info

Username*
user1
Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*
user1@user.com

Image*
Currently: [profile_pics/default.jpg](#)
Change:
 No file chosen

Admin can view the results dynamically like how many users voted ? What's their current priority ? etc.



Borda Winner for this poll gives an insight for the admin about the preferred time chosen by all the users

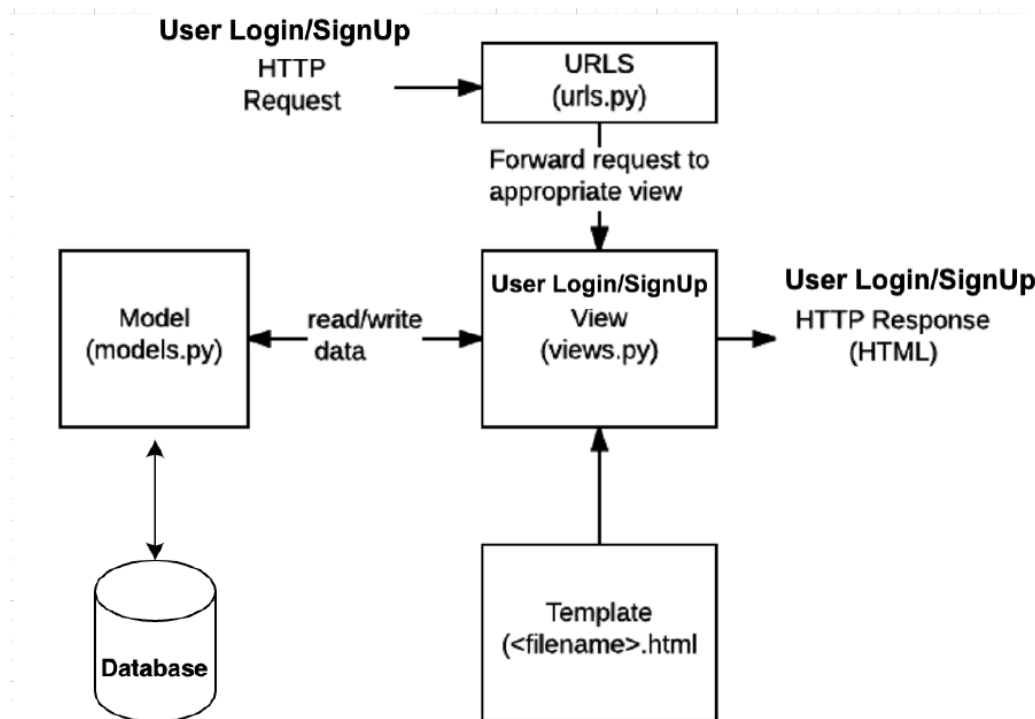
Database design diagram:

AUTH_USER_TABLE

ID	PASSWORD	LAST_LOGIN	IS_SUPERUSER	USERNAME	LAST_NAME	EMAIL	IS_STAFF	IS_ACTIVE	DATE_JOINED	FIRST_NAME

USERS_PROFILE_TABLE

ID	USER_ID	AUTH_USER	IMAGE

Design diagram:

Django uses the Model View Template (MVT) pattern. MVT has views for receiving HTTP requests and returning HTTP responses. When a user signs up or login using his credentials, the respective HTTP requests will be sent. Using urls.py, the application eventually calls the respective view. These views use the models (Login and Sign up models) that interact with the database. In this way, the users can login and sign up into the application.

Testing:

Test case 1: User email validation - Check if the given email is in valid format.

Test case 2: Password validation - Check if the password is meeting the requirements.

Test case 3: Authentication validation- Check if the user is providing the correct password.

Test case 4: Sign up validation - Check if the user is a new user not an existing one.

Test case 5: User sign in validation - Check if the user is providing the correct username.

Code Coverage

```
(myvenv) → Borda-Count git:(it1) × coverage run --source='.' manage.py test accounts
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....

Ran 5 tests in 0.487s

OK
Destroying test database for alias 'default'...
(myvenv) → Borda-Count git:(it1) × coverage report
```

Name	Stmts	Miss	Cover
accounts/__init__.py	0	0	100%
accounts/admin.py	1	0	100%
accounts/apps.py	4	0	100%
accounts/forms.py	52	7	87%
accounts/migrations/__init__.py	0	0	100%
accounts/models.py	1	0	100%
accounts/tests.py	45	2	96%
accounts/urls.py	4	0	100%
accounts/views.py	28	20	29%
bordaapp/__init__.py	0	0	100%
bordaapp/admin.py	5	0	100%
bordaapp/apps.py	4	0	100%
bordaapp/migrations/0001_initial.py	6	0	100%
bordaapp/migrations/__init__.py	0	0	100%
bordaapp/models.py	16	3	81%
bordaapp/tests.py	1	1	0%
bordaapp/urls.py	4	0	100%
bordaapp/views.py	15	8	47%
bordaproject/__init__.py	0	0	100%
bordaproject/asgi.py	4	4	0%
bordaproject/settings.py	21	0	100%
bordaproject/urls.py	6	0	100%
bordaproject/wsgi.py	4	4	0%
manage.py	12	2	83%
TOTAL	233	51	78%

Test case 6 : Post Object validation - Check if the post object is getting created successfully.

Test case 7 : Object creation : Check if Post Object is created with correct values.

Test case 8 : Submission Object validation - Check if the submission object is getting created successfully.

Test case 9 : User Authorization : Check if a user has access to submit preferences to a post.

Test case 10 : Borda Count Validation : Verify that the Borda Count algorithm runs successfully.

Test report:

```
Creating test database for alias 'default'...  
[.....  
-----
```

```
Ran 5 tests in 0.636s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

Challenges faced:

- Initial setup of Heroku deployment was a bit tricky. However, from the subsequent iterations the deployment process went smoothly. We used git as a version control mechanism for the heroku deployment which helped in easy deployment.
- Borda count algorithm is new to everyone across the team, so we had to do some thinking in that area
- Some of our team members were not very well familiarized with the Django framework, so they had to learn initially and increase their pace eventually.

Customer interaction:

- We had a zoom meeting with our customer at 12:30pm on 10/21/2021, in which we discussed the requirements, user flow etc.
- We had a zoom meeting along with a demo with our customer at 11 am on 11/1/2021, where we discussed the progress that we have made, took some feedback, and discussed the future stories.
- We had a zoom meeting along with a demo with our customer at 12:00 pm on 11/19/2021, in which we discussed the progress that we have made, demonstrated the changes, took some feedback, and discussed the future stories.
- We had a zoom meeting along with a demo with our customer at 11:00 am on 11/19/2021, in which we discussed the progress that we have made, took some feedback, and discussed the future stories.
- Had a meeting with Dr.Korok Roy to demonstrate the features implemented in our project and gave a final demo on 12/09/2021.

Software Development Process:

We have followed BDD methodology. Following BDD has helped speed up the development process. With BDD, as its name suggests, focus is on behavior rather than implementation itself. BDD enhances quality of the code, thus reducing the maintenance cost and minimizing project risk.

Configuration Management:

We have created separate branches for each iteration. Deployed the code to Heroku and tested the application by creating 5 users. Demo was presented to the client post every iteration.

Deployment Process:

The web application is deployed to heroku using git as a version control. Following commands are used:

```
heroku login
git add .
git commit -m "final commit"
git push heroku master
```

Future Scope:

- Add a feature to create polls by multiple authors/admins
- Given this polling application is restricted to setup only meetings, then we can add a feature to integrate interactive date pickers and calendars
- Although we have covered the basic input handling at all the places, we can add a feature to do comprehensive input handling
- Add a feature to make UI more interactive
- We can try using other variants of Borda Count algorithm with this application

Project Resources:***Heroku Deployment***

The application till today is deployed on Heroku at this link:

<https://bordacount.herokuapp.com/>

Pivotal Tracker

<https://www.pivotaltracker.com/n/projects/2535906>

Source code:

<https://github.com/msharsha/Borda-Count>

We maintained a different branch for each iteration and tagged them with the name of that respective iteration. Final source code till today can be found under the main branch.

Project Demo and Poster Video:

<https://vimeo.com/656439233>

THANK YOU