

Borda Count Final Report

Team Members and Roles

Prof. Korok Ray	Client
Amith Gopi	Supervisor (SPOC)
Sneha Singh	Team Member (Scrum Master)
Shaista Ambreen	Team Member (Product Owner)
Sunanda Saha	Team Member
Upasana Mishra	Team Member



**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. This is an area of Economics called Social Choice Theory, which is connected to Game Theory. 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.

It is a legacy project wherein the Fall 2021 CSCE 606 class performed the initial implementation of the website using Python Django. They had implemented a single-admin system with a sum-based algorithm to output the group's final preferences. As part of the expansion, the requirements for our team were to replace the existing Borda Count algorithm to Cumulative Voting Algorithm and to implement the multi-admin functionality to accommodate multiple admins.

The application is developed using the Django framework. 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, Amith Gopi and Dr. Philip Ritchey.

User Stories Implemented

Below are the user stories and the sub-tasks that we planned and completed successfully.

User stories related to Multi-Admin Functionality:

Feature: Multi Admin Login (Points - 4)

- As multiple administrators,
- So that anyone can log into the application and view/create polls on the dashboard
- I want to allow multiple admins to enter personal credentials on the login page

Feature: Adding New Admin (Points - 4)

- As an existing administrator,
- So that multiple admins can manage the application
- I want to add a new admin with random credentials which can be reset later.

Title	Multi Admin Login	Adding New Admin
Tasks	<div>TASKS (10/10)</div> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> UI wire-framing <input checked="" type="checkbox"/> Code walkthrough <input checked="" type="checkbox"/> Architecture overview <input checked="" type="checkbox"/> Understanding Existing Test Case framework <input checked="" type="checkbox"/> DB compatibility check and setup <input checked="" type="checkbox"/> Implementing multiple admins login <input checked="" type="checkbox"/> Implementing Unit/Integration Tests <input checked="" type="checkbox"/> E2E Testing <input checked="" type="checkbox"/> Cloud Deployment <input checked="" type="checkbox"/> Peer Review 	<div>TASKS (10/10)</div> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> UI Wireframing <input checked="" type="checkbox"/> DB Compatibility and Setup <input checked="" type="checkbox"/> Implementing add admin feature <input checked="" type="checkbox"/> Implementing Unit/Integration Test Cases <input checked="" type="checkbox"/> E2E Testing <input checked="" type="checkbox"/> Cloud deployment <input checked="" type="checkbox"/> Peer Review <input checked="" type="checkbox"/> Add create dropdown with two options: Poll and Admin <input checked="" type="checkbox"/> Notify on Dashboard on successful Admin Creation <input checked="" type="checkbox"/> Admin Register Screen changes (Add username to existing user register)

User stories related to Cumulative Borda Count Algorithm:

Feature: Cumulative Voting Backend Changes (Points - 4)

- As an admin,
- So that I can get the highest preference,
- I want to replace the existing algorithm with Cumulative Voting.

Feature: Cumulative Voting UI Changes (Points - 4)

- As a user,
- So that I can give a number of points to each alternative,
- I want to replace the ranking with the intensity of preferences.

Title	Cumulative Voting Backend Changes	Cumulative Voting UI Changes
Tasks	<div data-bbox="451 751 875 779">TASKS (8/8)</div> <div data-bbox="459 789 867 1255"> <div data-bbox="459 789 867 837">✓ Code Walkthrough</div> <div data-bbox="459 848 867 896">✓ Understanding Existing Test Case Framework</div> <div data-bbox="459 907 867 955">✓ DB Compatibility Setup</div> <div data-bbox="459 966 867 1014">✓ Backend Code Changes</div> <div data-bbox="459 1024 867 1073">✓ Unit and Integration Test Cases</div> <div data-bbox="459 1083 867 1131">✓ E2E Testing</div> <div data-bbox="459 1142 867 1190">✓ Cloud Deployment</div> <div data-bbox="459 1201 867 1249">✓ Peer review</div> </div>	<div data-bbox="907 751 1308 779">TASKS (10/10)</div> <div data-bbox="915 789 1308 1318"> <div data-bbox="915 789 1308 837">✓ Code Walkthrough</div> <div data-bbox="915 848 1308 896">✓ Understanding Existing UI Test case Framework</div> <div data-bbox="915 907 1308 955">✓ UI Wireframing</div> <div data-bbox="915 966 1308 1014">✓ UI changes</div> <div data-bbox="915 1024 1308 1073">✓ Saving preferences to DB</div> <div data-bbox="915 1083 1308 1131">✓ Exploring Survey Monkey</div> <div data-bbox="915 1142 1308 1190">✓ Unit and Integration Test cases</div> <div data-bbox="915 1201 1308 1249">✓ E2E Testing</div> <div data-bbox="915 1260 1308 1308">✓ Cloud Deployment</div> <div data-bbox="915 1318 1308 1367">✓ Peer Review</div> </div>

Iteration wise progress

Iteration 0: (2 points)

- Organized the team and developed a meeting schedule
- Scheduled meeting with the client and supervisor to discuss on the user requirements for the project

Iteration 1: (2 points)

- Had meeting with the client to acquire better understanding of the project and its requirements.
- Code walkthrough and completed the initial local setup of the project
- Came up with user stories for the application and developed UI wireframes.
- Presented client with the live demo of the wireframe and working prototype of the multi admin functionality.
- Acquainted ourselves with the Django framework.

Iteration 2: (4 points)

- Developed both the user interface and backend for implementing multi admin feature.
- Delivered multi admin feature via two different methods mentioned below:
 - The existing admin creates a new admin with username and password. The newly created admin can then login and change their passwords.
 - Second approach is that the new admin registers as a user with their preferred credentials and then an admin gives them admin permissions.
- Created interface for admin to add a new admin and to give admin permissions to a user
- Implemented UI changes to deliver notification after a new admin is added.
- Performed unit and integration testing of the implementation and deployed the changes to the Heroku.
- Demonstrated the actual working multi admin functionality to the client on Heroku.

Iteration 3: (4 points)

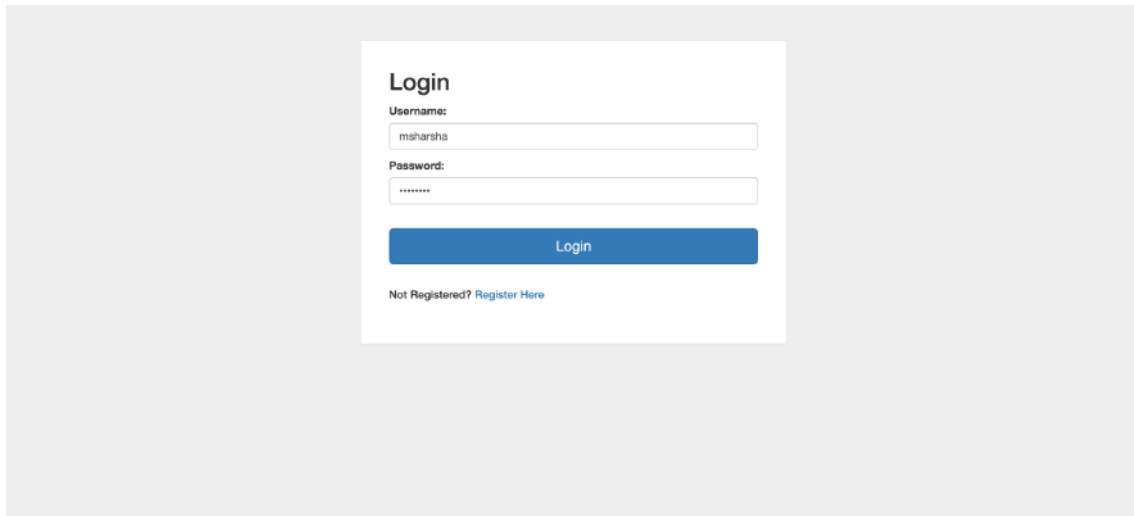
- Discussed the queries with supervisor for implementing the new Cumulative Borda-Count Algorithm.
- Discussed the validations for the new algorithm with the supervisor.
- Scheduled a meeting with previous team to discuss their approach to update the preference collecting mechanism.
- Generated the UI wireframes to collect the preferences and made backend changes to update the aggregator.
- The Database Schema for the new changes is designed.

Iteration 4: (4 points)

- Implemented the new Cumulative Algorithm with the required backend and UI changes.
- Added the necessary backend validations for the new algorithm.
- Added the UI implementations for the necessary validations implemented in the backend.
- Implemented Text Box Preferences on UI as well as Backend.
- The user can submit their preferences by filling in their choices in the text boxes.
- Performed unit and integration testing and deployed the changes to Heroku.
- Merged all the changes to the parent repository.
- Presented client with the live demo of the working prototype.

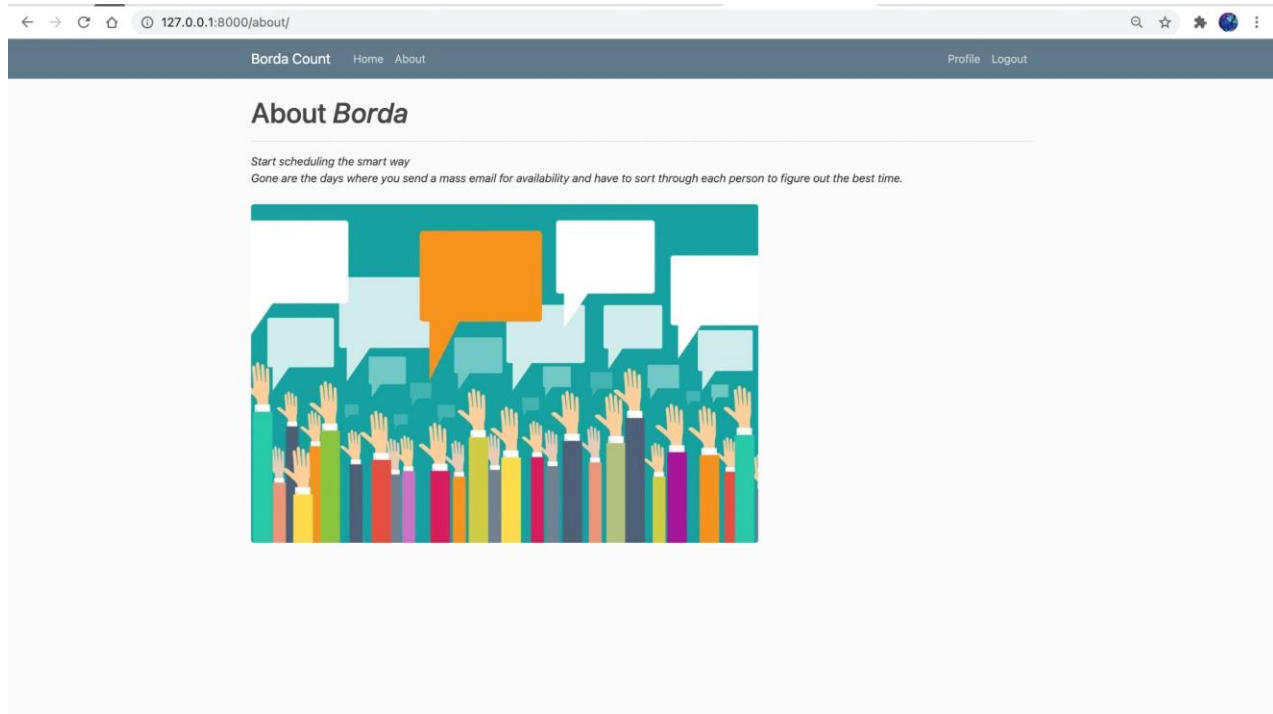
User Interface

Login Page



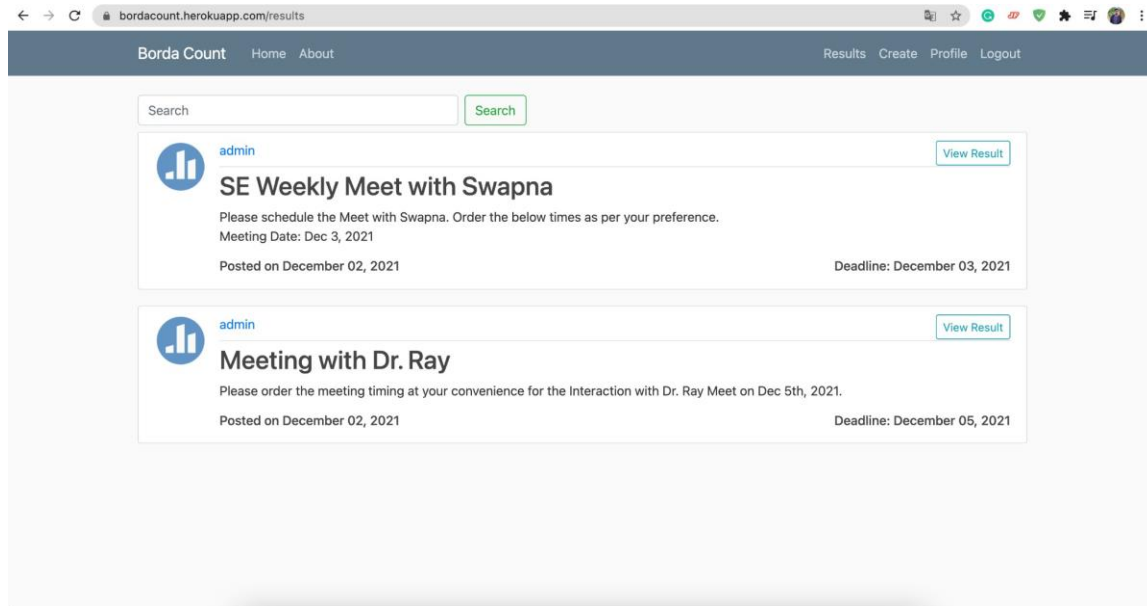
A login form titled "Login" is centered on a light gray background. The form is white and contains the following elements: a "Username:" label followed by a text input field containing "maharsha"; a "Password:" label followed by a password input field with masked characters "*****"; a blue "Login" button; and a link "Not Registered? Register Here" below the button.

About Page

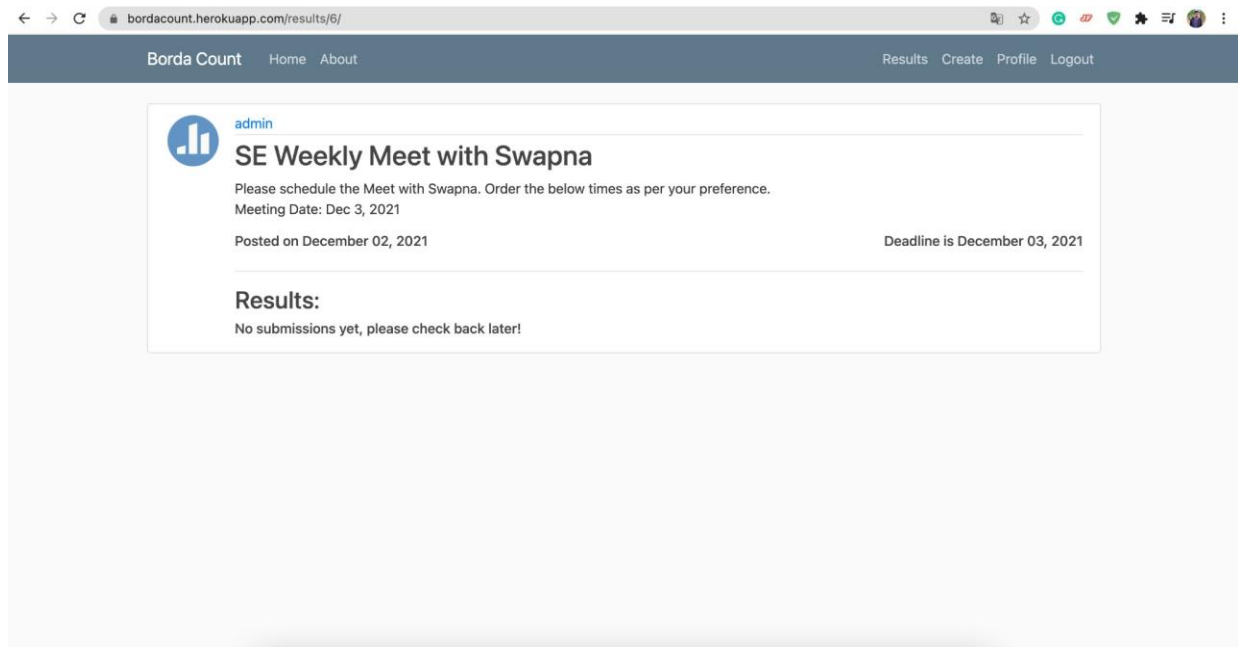


Admin flow

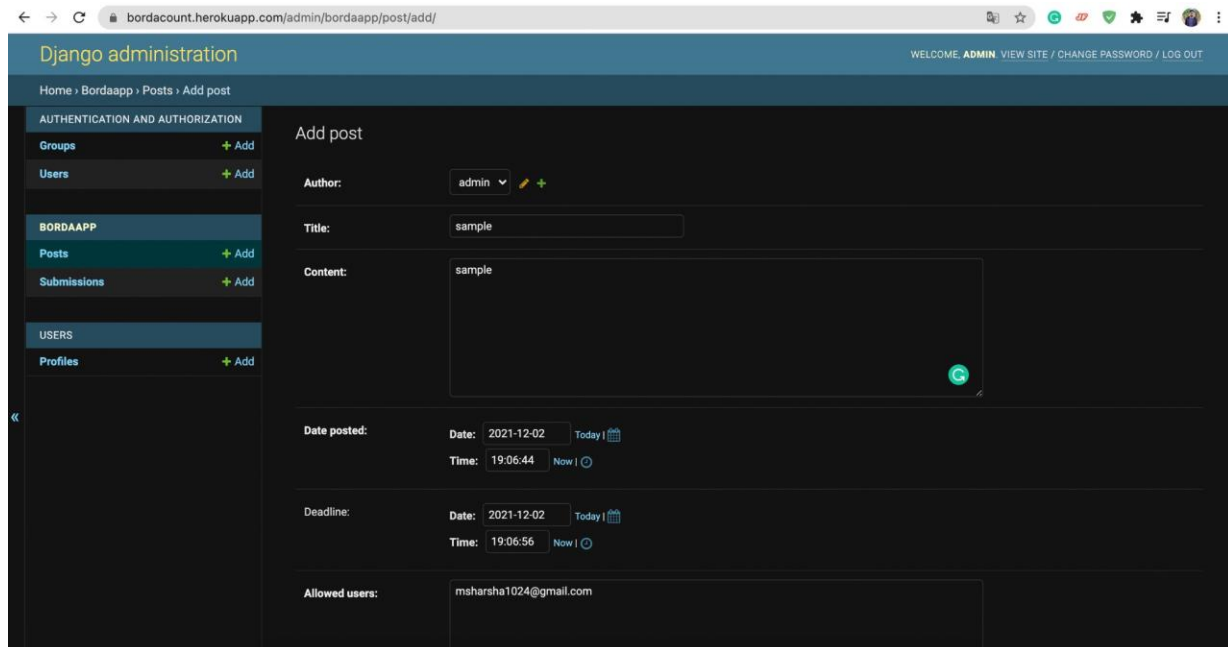
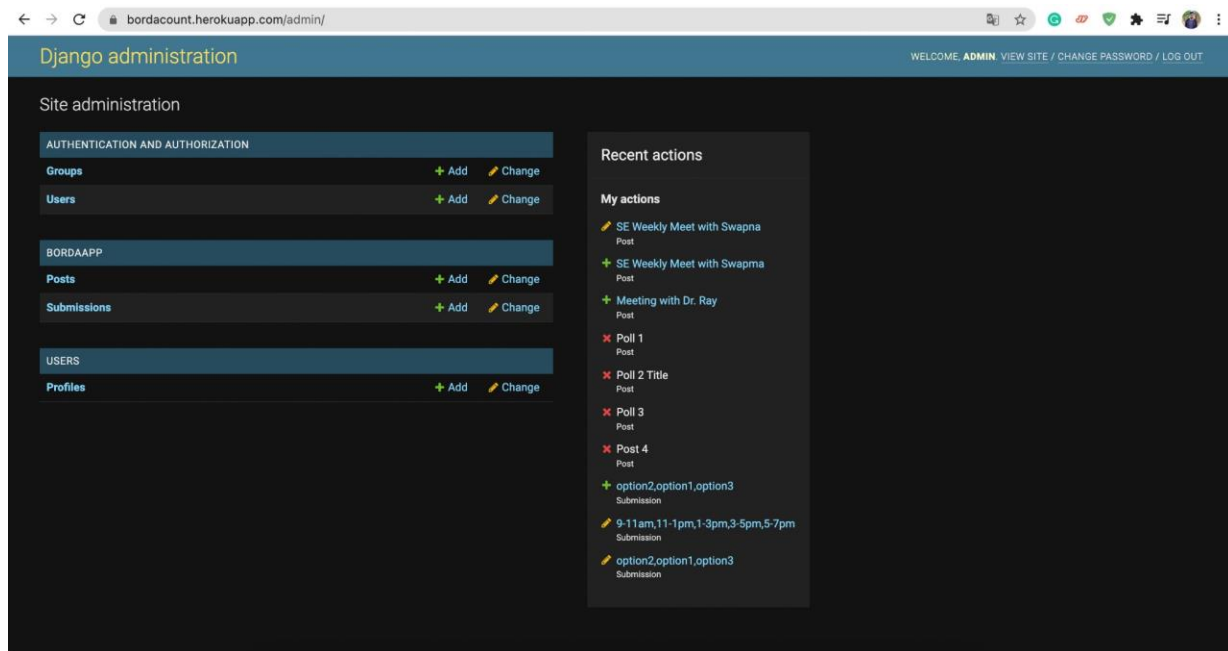
Once the login, upon clicking on **results** he can view all the polls that admins created.



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 have responded.



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



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

Multi-Admin Functionality Flow:

First Approach (Admin creates another admin):

Admin – Login

Click “Add” to add new user

Admin gives credentials for new user

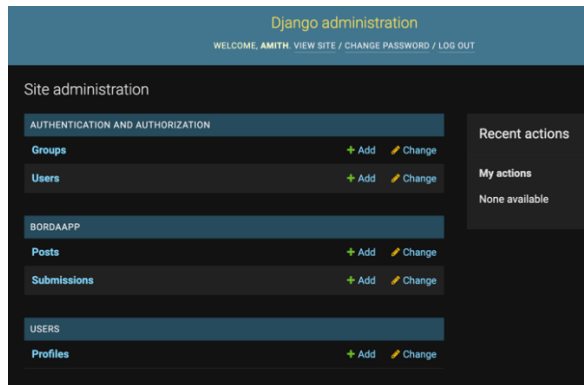
User Added

Added user appears on list

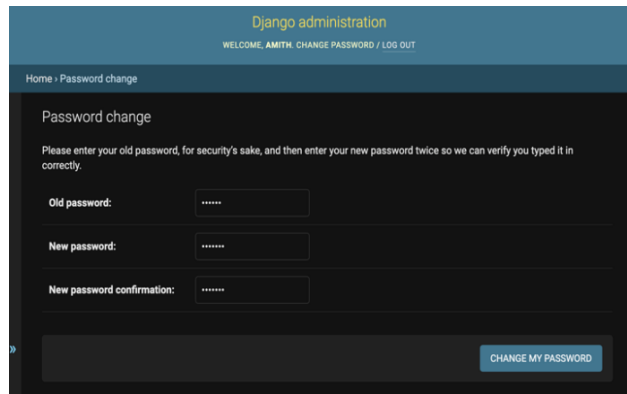
USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
admin	admin@admin.com			✓
new_admin	amith.gopi@tamu.edu	Amith	Gopi	✓
user1	user1@user.com			✗

User logs in using added credentials

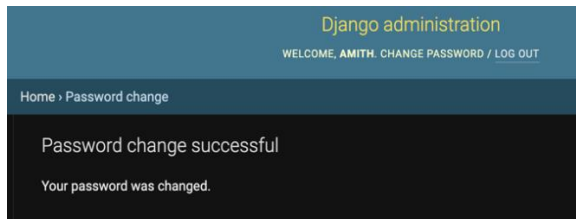
User homepage



User changes the password



Password changed successfully



Second Approach (Admin grants existing user admin permissions):

New User SignUp

Join Today

Username*

user_to_admin

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

Email*

user_to_admin@email.com

Password*

.....

Password confirmation*

.....

Enter the same password as before, for verification.

[Sign Up](#)

Already Have An Account? [Sign In](#)

User Login

Your account has been created! You are now able to login.

Log In

Username*

user_to_admin

Password*

.....

[Login](#) [Forgot Password?](#)

Need An Account? [Sign Up Now](#)

User present in the list of users

Homepage

Borda Count Home About Profile Logout

Search [Search](#)

No Polls

Login to see the polls

Django administration

WELCOME, AMITH. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Users

Select user to change

Search

0 of 4 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
admin	admin@admin.com			Yes
new_admin	amith.gopi@tamu.edu	Amith	Gopi	Yes
user1	user1@user.com			No
user_to_admin	user_to_admin@email.com			No

4 users

FILTER

By staff status

All Yes No

By superuser status

All Yes No

By active

All Yes No

Change user info page

Django administration

WELCOME, AMITH. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Users > user_to_admin

Change user

user_to_admin

Username:

user_to_admin

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

Password:

algorithm: pbkdf2_sha256 iterations: 260000 salt: KdHfS3***** hash: jDgk4*****

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email address:

user_to_admin@email.com

Permissions

☒ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ Staff status

Designates whether the user can log into this admin site.

☐ Superuser status

Designates that this user has all permissions without explicitly assigning them.

Add the user as admin

Django administration

WELCOME, AMITH. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Users > user_to_admin

Change user

user_to_admin

Username:

user_to_admin

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

Password:

algorithm: pbkdf2_sha256 iterations: 260000 salt: KdHfS3***** hash: jDgk4*****

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email address:

user_to_admin@email.com

Permissions

☒ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

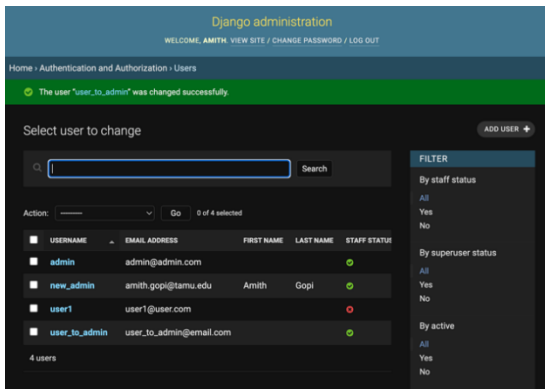
☐ Staff status

Designates whether the user can log into this admin site.

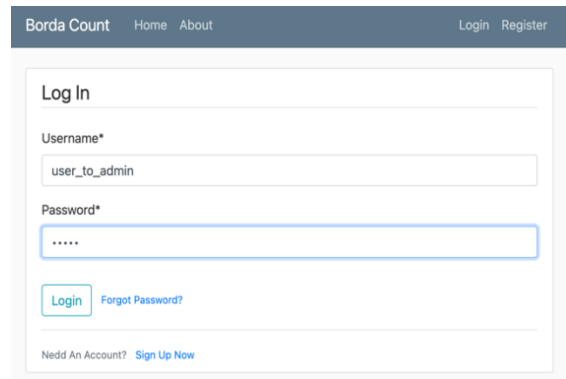
☒ Superuser status

Designates that this user has all permissions without explicitly assigning them.

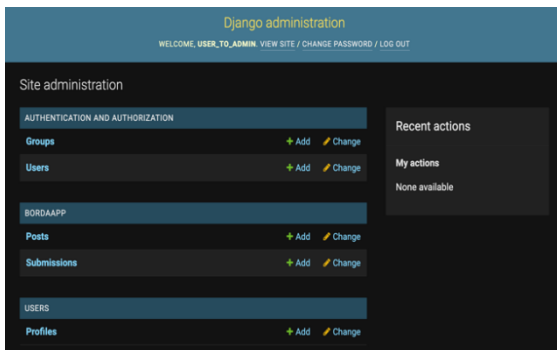
User changed to admin successfully



User can now log in as admin

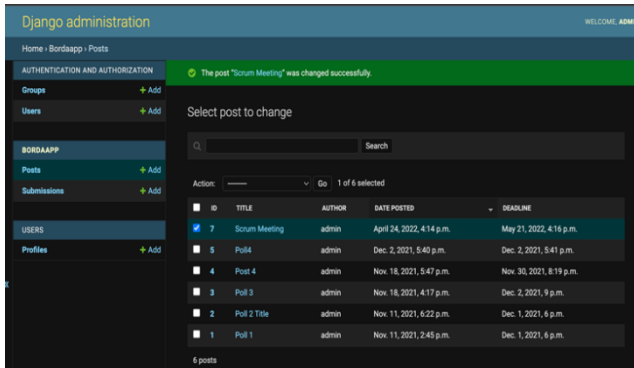


Admin Homepage



Cumulative Borda Count Implementation Flow:

Poll is created by admin



User logs in to take the poll

Poll as it appears to the user

Poll as visible on the user dashboard

Comparison between old and new UI:

Previous screen with old Borda Count Algorithm

Posted by admin
admin@admin.com

Scrum Meeting

Please give your preferred timing for scrum meetings among the below options.

Posted on April 24, 2022

Poll-features

- Monday 10:30AM
- Monday 11:00AM
- Monday 11:30AM
- Monday 12:00PM
- Friday 10:30AM
- Friday 11:00AM
- Friday 11:30AM
- Friday 12:00PM

Save

New Screen with Cumulative Borda Count Algorithm

Posted by admin
admin@admin.com

Scrum Meeting

Please give your preferred timing for scrum meetings among the below options.

Posted on April 24, 2022

Please tell us your preferred timing

	Preference Points *
Monday 10:30AM	
Monday 11:00AM	
Monday 11:30AM	
Monday 12:00PM	
Friday 10:30AM	
Friday 11:00AM	
Friday 11:30AM	
Friday 12:00PM	

Total Points: 0

Save

Previously user needed to drag and drop preferences

Posted by admin
admin@admin.com

Scrum Meeting

Please give your preferred timing for scrum meetings among the below options.

Posted on April 24, 2022

Poll-features

- Monday 10:30AM
- Friday 11:30AM
- Friday 10:30AM
- Friday 12:00PM
- Monday 11:30AM
- Monday 12:00PM
- Monday 11:00AM
- Friday 11:00AM

Save

Currently user needs to allocate points to the preferences

Posted by admin
admin@admin.com

Scrum Meeting

Please give your preferred timing for scrum meetings among the below options.

Posted on April 24, 2022

Please tell us your preferred timing

	Preference Points *
Monday 10:30AM	0
Monday 11:00AM	0
Monday 11:30AM	2
Monday 12:00PM	0
Friday 10:30AM	1
Friday 11:00AM	2
Friday 11:30AM	0
Friday 12:00PM	3

Total Points: 8

Save

Required validations:

When total number of points exceeds the total points allocated to the user (Total points is equal to the number of alternatives)

Maximum points allowed: 8 ; Current points: 10

Scrum Meeting
Please give your preferred timing for scrum meetings among the below options.
Posted on April 24, 2022

Please tell us your preferred timing

	Preference Points *
Monday 10:30AM	3
Monday 11:00AM	4
Monday 11:30AM	3
Monday 12:00PM	
Friday 10:30AM	
Friday 11:00AM	
Friday 11:30AM	
Friday 12:00PM	

Total Points: 10

Save

When negative points is allocated to a preference

Each option should have points between 0 and 8(inclusive).

Scrum Meeting
Please give your preferred timing for scrum meetings among the below options.
Posted on April 24, 2022

Please tell us your preferred timing

	Preference Points *
Monday 10:30AM	3
Monday 11:00AM	-1
Monday 11:30AM	3
Monday 12:00PM	
Friday 10:30AM	
Friday 11:00AM	
Friday 11:30AM	
Friday 12:00PM	

Total Points: 5

Save

When no points are assigned to any option

Assign points to atleast 1 option.

Scrum Meeting
Please give your preferred timing for scrum meetings among the below options.
Posted on April 24, 2022

Please tell us your preferred timing

	Preference Points *
Monday 10:30AM	0
Monday 11:00AM	0
Monday 11:30AM	0
Monday 12:00PM	
Friday 10:30AM	
Friday 11:00AM	
Friday 11:30AM	
Friday 12:00PM	

Total Points: 0

Save

When points allocated to an option exceeds the total points allocated to the user

Each option should have points between 0 and 8(inclusive).
Maximum points allowed: 8 ; Current points: 9

Scrum Meeting
Please give your preferred timing for scrum meetings among the below options.
Posted on April 24, 2022

Please tell us your preferred timing

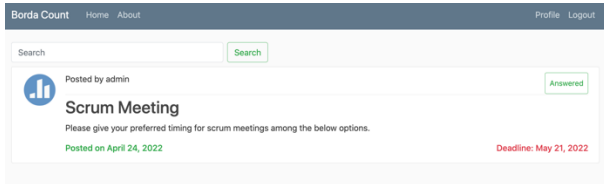
	Preference Points *
Monday 10:30AM	0
Monday 11:00AM	0
Monday 11:30AM	0
Monday 12:00PM	0
Friday 10:30AM	0
Friday 11:00AM	0
Friday 11:30AM	0
Friday 12:00PM	9

Total Points: 9

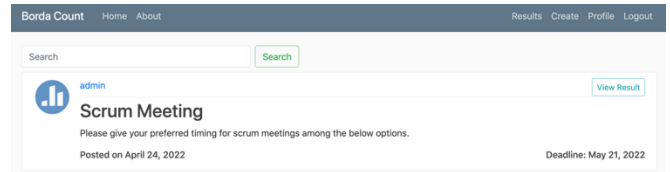
Save

Results:

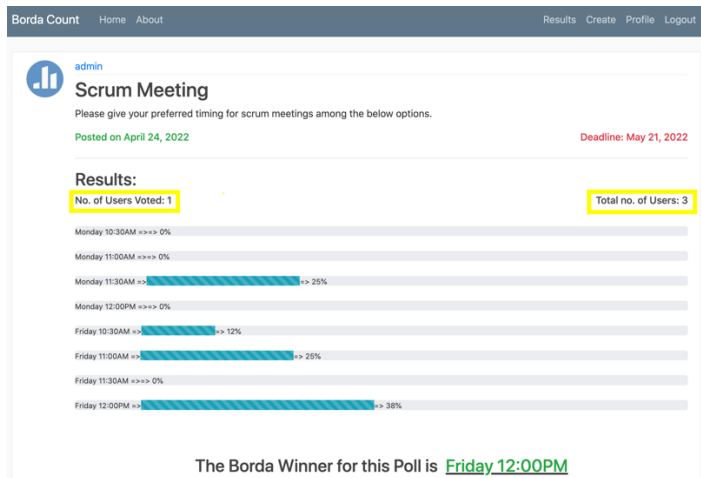
After the user has completed the poll



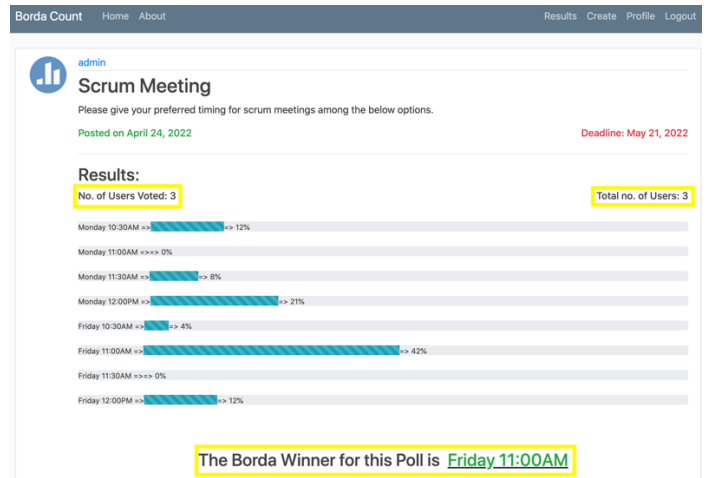
The admin can view the results



Results when only a percent of users have taken the poll



User needs to allocate points to the preferences



Borda Winner for this poll gives an insight to 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

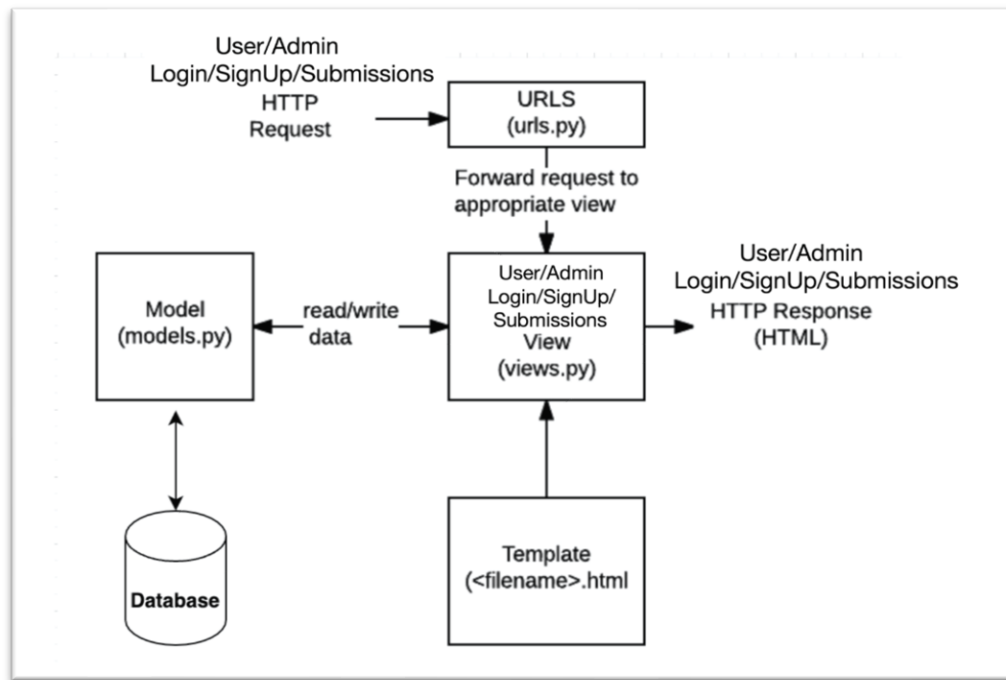
ID	USER_ID	AUTH_USER	IMAGE

Post design:

ID	AUTHOR	TITLE	CONTENT	DATE_POSTED	DEADLINE	ALLOWED_USERS	ANSWERED_USERS	OPTIONS

Submission design:

ID	OPTIONS	PREFERENCES	POST_ID	SUBMITTED_BY	SUBMITTED_DATE

Design diagram:

Django uses the Model View Template (MVT) pattern. MVT has views for receiving HTTP requests and returning HTTP responses. When an admin/user creates/submits/views a poll, the respective HTTP requests will be sent. Using `urls.py`, the application eventually calls the respective view. These views use the models (Post & Submission) that interact with the database. In this way, the admin can create and view polls and users can submit their preferences.

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:

- Borda count algorithm was new to everyone across the team, so we went through Wikipedia and other learning sources to understand the algorithm.
- The team members were not very familiar with the Django framework, so they had to learn and adapt to it quickly.
- Challenge with AWS Cloud9 implementation – We faced an issue with the version of SQLite3 while setting up the project in AWS Cloud9. It was solved by following a procedure to upgrade the version of Django.
Following is the link to the solution: [Solution to version upgrade of Sllite3](#)
- Deployment Issue: On altering the existing Submission model to add a new attribute, the existing data started throwing exceptions, as the field was not present for old rows. We tried adding the default argument to solve the issue, but it still did not help. So after checking the production data and confirming it was raw and meaningless, we unmigrated the existing migration files, deleted all existing migration files under 'bordaapp/migrations', recreated migrations and pushed them using the below command:

```
heroku run python manage.py migrate bordaapp zero
delete all the migration files under 'bordaapp/migrations'
heroku run python manage.py makemigrations
git add .
git commit -m <msg>
git push heroku main
heroku run python manage.py migrate
```

Customer Interaction

- **02/23/2022:** We met in-person with our customer and received the overview of the project.
- **02/27/2022:** We scheduled weekly scrum calls on zoom with our customer for every Wednesday from 11:30am – 12:00pm.
- **03/02/2022 MoMs:**
 - Supervisor to clarify requirements about multiple admins
 - Team to check the legacy code.
 - Team to create wireframe for adding new Admin.
 - Team to explore the DB changes that needs to be done.
- **03/23/2022 MoMs:**
 - Team demoed working UI prototype as a plan for multi-admin functionality.
 - Discussed queries about the future work for the project, Cumulative Borda Count.
- **03/30/2022 MoMs:**
 - Team demoed multi-admin functionality using two approaches on Heroku.
 - Supervisor to forward it to Prof. Ray and gather further feedback.
- **04/06/2022 MoMs:**
 - Discussed the queries about the new Cumulative Borda-Count Algorithm.
- **04/20/2022 MoMs:**
 - Team demoed the implemented version of Cumulative Borda-Count.
 - Discussed the validations we implemented for the new algorithm and asked for further inputs.
- **05/10/2022 MoMs:**
 - Team to give final walk-through of Multi-admin functionality and Cumulative Borda Count to Prof. Ray and his team.

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 every iteration for implementing the user requirements and once implemented we merged the changes to the parent repository, deployed the code to a private Heroku app and tested the application by creating 4 users and 2 admins. Demo was presented to the client post every iteration.

The credentials for the initial user and admin is held by Prof. Korok Ray and will be provided to the future teams. No other credentials are needed.

Repo Content

Installation: (also present in Readme.txt)

- To create Virtual Environment (First Only, skip this step and activate Virtual Environment if you have the setup already.)

```
python3 -m venv myvenv
```

- To activate Virtual Environment:

```
source myvenv/bin/activate
```

- To update pip (First time only, skip this step if you're not setting up the application)

```
python -m pip install --upgrade pip
```

- To install necessary packages for the applications, use:

```
pip install -r requirements.txt
```

Note: requirements.txt has the required packages for this application. Do `pip freeze > requirements.txt` to add newly installed packages.

- To run migrations:

```
python manage.py migrate
```

Note: Migrations are to be run for the first time and only when there's a change in models.py

- To run the server:

```
python manage.py runserver
```

Visit <http://127.0.0.1:8000/> to see the Application running.

Deployment

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

```
heroku login
heroku git:remote -a bordacount
git push heroku main
heroku logout
```

Future Scope

- Add email validations and forward them emails once a new poll is created for them.
- Change the UI for accepting input for allowed users and options to make poll creation easy and convenient.
- Add features to allow poll creation for distributed lists/user groups, rather than individual users.
- Although we have covered the basic input handling at all the places, we can add a feature to do comprehensive input handling.
- We can try using other variants of Borda Count algorithm with this application.

Project Resources

Heroku App

The application till today is deployed on Heroku at this link: <https://bordacount.herokuapp.com/>

Heroku Project Link: <https://dashboard.heroku.com/apps/bordacount>

Pivotal Tracker: <https://www.pivotaltracker.com/n/projects/2556139>

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

Final source code till today can be found under the main branch.

Slack: <https://softwareengin-nat9577.slack.com/archives/C033ZEWBT9B>

Project Demo and Poster Video: <https://youtu.be/W21kQuD2QHs>

- 00:00 – Introduction (Poster Presentation)
- 04:10 – Application Demo
- 14:47 - Feedback

THANK YOU