



# IWP Assignment 2

## Pollster

11/11/2021

---

Gowtham Gorrepati - BT19CSE033  
Koya Sree Prabhav Sai - BT19CSE055

## 1. About the project - [Github](#)

This is a simple polling web application that helps people to create polls and take surveys, vote for polls and view statistics. This project is created by 2 CSE '23 undergraduates of [VNIT](#), [Gowtham](#) and [Prabhav](#).

## 2. Technologies used

- **ReactJS**
  - It is an open-source JavaScript framework for UI Components.
  - We have used useState, useEffect, useParams hooks and rendered multiple components using map.
  - useState is used to store and update the state of DOM.
  - useEffect is used to retrieve data from API, and update the DOM through useState variables.
  - We rendered multiple components using the map function.
- **React Router**
  - It is the standard routing library for React.
  - We have used useParams, Route, Link, Redirect, BrowserRouter, Switch.
  - useParams is used to read parameters from the API endpoint.
  - Route is used to render a specific UI based on the current URL.
  - Link is used to change the route on click.
  - Redirect is used to change route conditionally.
  - BrowserRouter along with Switch is used.
- **JWT**
  - JSON Web Token (JWT) is an open standard that is used to securely transmit information between parties as a JSON object.
  - We have used it for authentication of the user.
- **ChartJS**
  - ChartJS is a community maintained open-source library that helps you easily visualize data using JavaScript.
  - We have used Bar and Pie charts for visualizing the results.
- **NodeJS & ExpressJS**
  - NodeJS is an open-source, JavaScript runtime environment. Express is a minimal and flexible NodeJS framework.
  - We have used it for running our backend server.
- **MongoDB & Mongoose**
  - MongoDB is a NoSQL database. Mongoose is an ODM(Object Document Mapper) for MongoDB.

- We have used the cloud database service, MongoDB Atlas to store and manipulate data.
- Axios
  - It is a promise-based HTTP Client for NodeJS and the browser.
  - We have used it to make requests to the backend to transfer data.

### 3. Features

- One can create and conduct polls at ease.
- One can vote one's opinions on polls.
- One cannot vote the same poll once again.
- One can delete his/her created polls at any time.
- One cannot access the website after one's session (1hr) expires, and is redirected to the login page.
- One can view the statistics of the votes on one's polls.

### 4. Code

#### 4.1 Environment Variables File (.env)

The .env file contains sensitive and important information like :

- JWT\_SECRET\_KEY - A random string used to encode the payload into JWT
- MONGO\_URI - URI of the cluster, present in MongoDB Atlas
- SALT\_ROUNDS - Number of salt rounds required to hash password using "bcrypt.js"
- TOKEN\_LIFE - Lifetime of a JSON Web Token

#### 4.2 Database

The database schema and connection procedure is present in database.js file.

We use MongoDB Atlas to store all the data.

Mongoose framework is used for database management in the backend.

Connection to the MongoDB cluster is made using the URI of the cluster, along with the authentication credentials (stored in .env file).

##### **Database Structure:**

We made 2 collections, Users and Polls. They are accessed through the mongoose models made using the respective schemas.

##### **Poll:**

A poll contains

- a question (String),

- an array of options, each containing a key-value pair of (option name, frequency of selection).
- The author of the Poll (String)
- An array of voters (String)

#### User:

A user contains

- a username (String)
- the hashed password (String)

### 4.3 Backend abstracted as an API

#### Middleware : authenticateToken

Parses the JWT, passes through the “authorization” header, and verifies it. If successful, passes the user, extracted from the JWT, through the req object to the next middleware. If failed, it returns the appropriate error.

Type of request and route	Request parameters	Response (if request is successful)	Description
GET /api/verify	none	Current user's username	Verifies the JWT which is passed through the “authorization” header
POST /api/register	Username and password	none	Saves the username and the hashed password into the database as a new User.
POST /api/login	Username and password	none	Verifies the password for that username. If true, a JWT is created and stored in localStorage
POST /api/create	Poll question and options	none	Creates a new poll with the passed parameters, along with the author as current user
GET /api/polls	none	All polls and current user's username	Sends all the polls and their authors
GET /api/poll/:id	ObjectId of Poll	Poll of that ObjectId, and current user's username	Sends the poll of that ObjectId and current user's username

DELETE /api/poll/:id	ObjectId of Poll	none	Deletes the poll of that ObjectId in the database
POST /api/vote/:id	ObjectId of Poll	none	Increments the count of the option vote frequency. Also adds the current user's username into the voters array. Updates the Poll of that ObjectId, with this updated information in the database

## 4.4 Authentication

### Register:

When the user enters the credentials for registration, the backend checks if a user exists with the same username. If it exists, then an error is raised. Otherwise the password is hashed with a random salt, and stored along with the username as a new user.

### Login:

When the user enters the credentials for login, the backend checks if this exists by using the username. If it doesn't exist, then an error is raised. Otherwise the password is hashed and compared with the stored password. If it matches, the user is redirected to the home page, else an error is raised.

### JWT verification:

We use the authenticateToken middleware for every API request to the backend, before retrieving or sending data. If the JWT is expired, an error is received, the user is redirected to the login page.

## 4.5 React Components

### Create:

This page helps the user to create polls, which contains questions and options (which can be added or removed at the time of creation). After the user submits them, an API call is made to the backend to send the data via the authentication middleware as discussed above.

### Polls:

This page contains all the polls by default where each poll is truncated to atmost first 100 characters of the question. The user can click on 'My Polls' to see only his/her polls and also delete them. Upon deletion, an API call is made to the backend to delete the data via the authentication middleware as discussed above. When the user clicks on any poll, the page gets redirected to 'Poll'.

**Poll:**

This page is used to view the question and options of each poll clearly and also vote for that poll. Upon voting, an API call is made to the backend to vote via the authentication middleware as discussed above. If a user tries to vote again, an error message is raised. There is another button 'Stats' to redirect to the current results of the poll.

**Stats:**

This page is used to view the current results of the poll. On reload, an API call is made to the backend to fetch data via the authentication middleware as discussed above. Bar and Pie charts from ChartJS are used to display the data.

## 4.6 Navbar, Nav, Footer

**Navbar:**

This is the most important component of the web application which is rendered in almost every other component. It contains the links to various other pages. It displays respective links in the nav-list, based on the authentication of the user, i.e. User is logged in or not.

On each render, it makes a call to the backend to verify if the user is logged in or not. If the user is not logged in, it redirects to the Login page, else the username is displayed at the top and the user gets to taste the features of the website.

**Nav:**

This is an auxiliary component similar to Navbar, but doesn't display menu items. It is used to display when the nav-list is loading. This is done because it takes some time to choose and display the appropriate nav-list based on whether the user is authenticated or not.

**Footer:**

This component is present at the bottom of every page, mentioning the copyright statement.

## 4.7 Miscellaneous Pages - About, Dashboard, Home

**App:**

This component is used to render the appropriate page based on the switch condition of react routes.

**About:**

This is a static page used to display information about the web application and its creators.

**Dashboard:**

This is the landing page to be displayed if no user has logged in. It contains a gist of the web application and options to login and register.

**Home:**

This is the landing page to be displayed if a user has logged in. It contains a gist of the web application and relevant options like Create, Polls, About, Logout.

**NotFound:**

This page is rendered if the user tries to access a page which is not present in the application. It shows a 404 Error and gives an option to redirect to Home.

## 4.8 Miscellaneous Components - Error, Features, NotFound, NoPolls, Loader, Success

**Error:**

This component is used to print any error messages of a page present in the error buffer which are accumulated whenever there is a requirement.

**Success:**

This component is used to print any success messages of a page present in the success buffer which are accumulated whenever there is a requirement.

**Features:**

This component contains the gist of the web application and is used in Dashboard and Home pages.

**NoPolls:**

This component is used to convey that no polls are present if there are none. It also helps the user to create one, if he/she wants to.

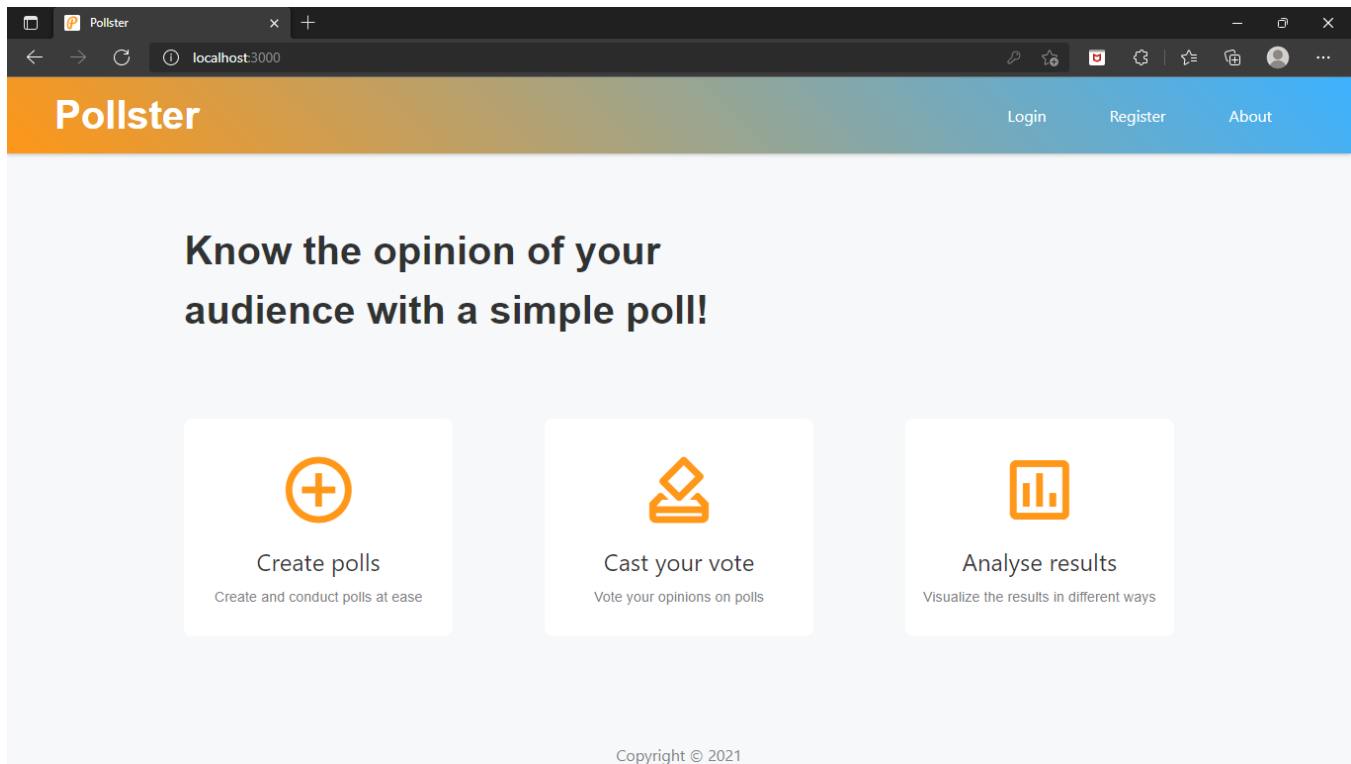
**Loader:**

This is an important component which is rendered in pages with API calls. It is a simple loading animated component. It helps to pause the display of unnecessary content until the data is fetched from the backend.

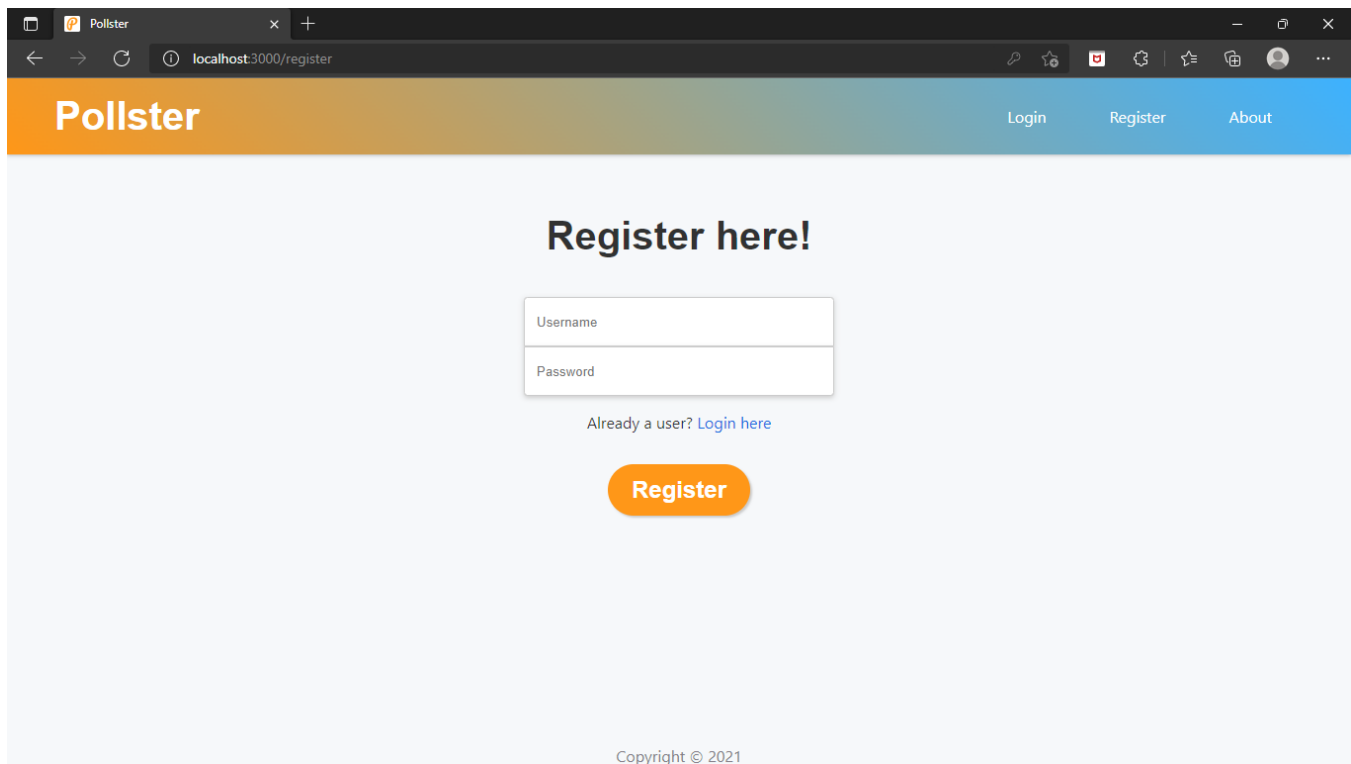
## 5. A glimpse of the web application

### Desktop

#### Dashboard:

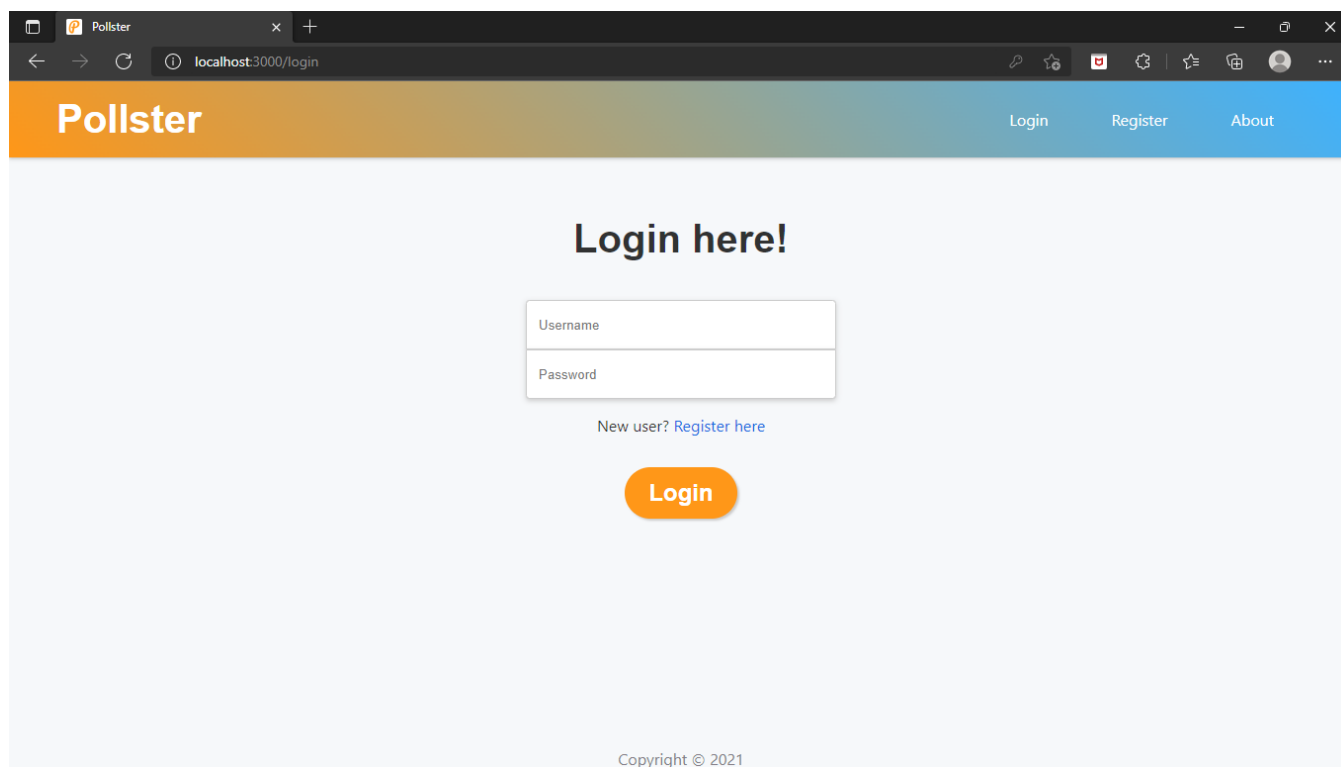


#### Register:





## Login:



A screenshot of a web browser displaying the Pollster login page. The browser's address bar shows 'localhost:3000/login'. The page has a header with the 'Pollster' logo on the left and 'Login', 'Register', and 'About' links on the right. The main content area features the heading 'Login here!' followed by a form with 'Username' and 'Password' input fields. Below the form is a link 'New user? Register here' and an orange 'Login' button. The footer contains the text 'Copyright © 2021'.

Username

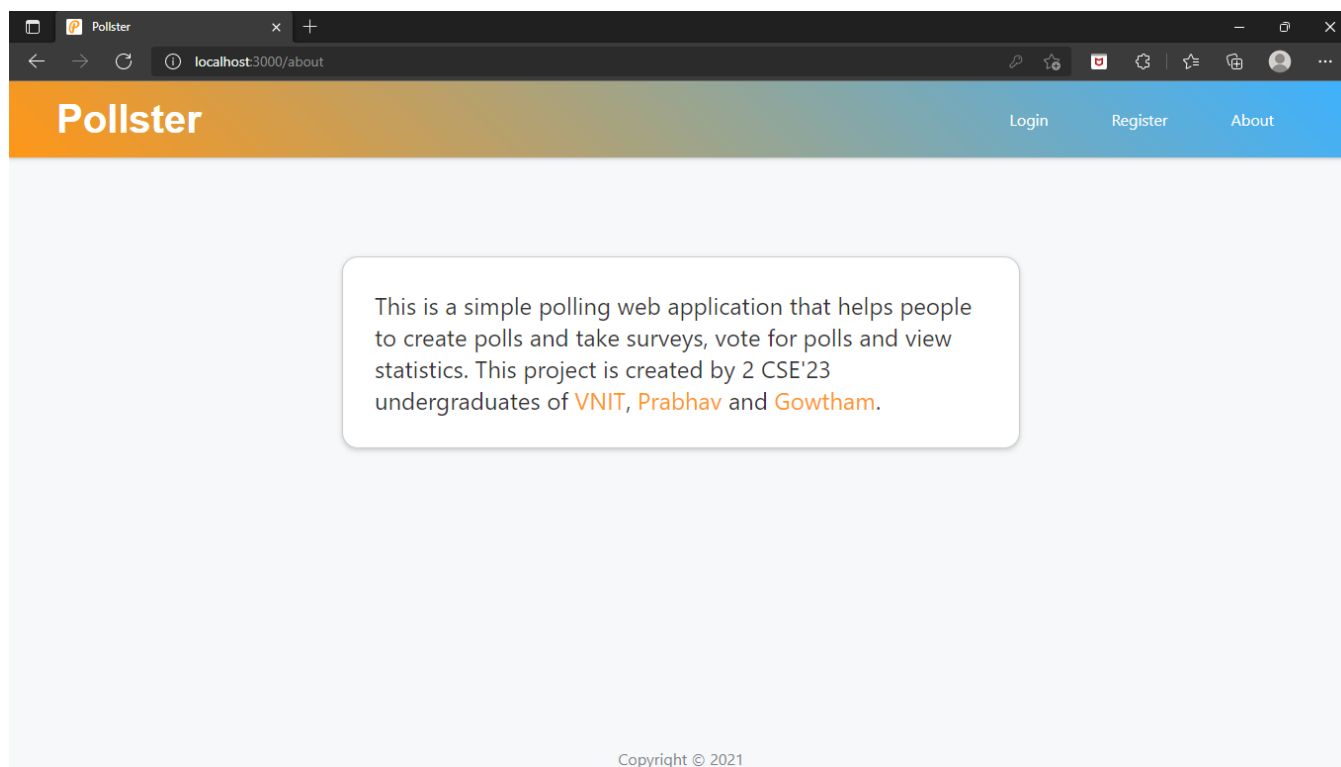
Password

New user? [Register here](#)

Login

Copyright © 2021

## About:

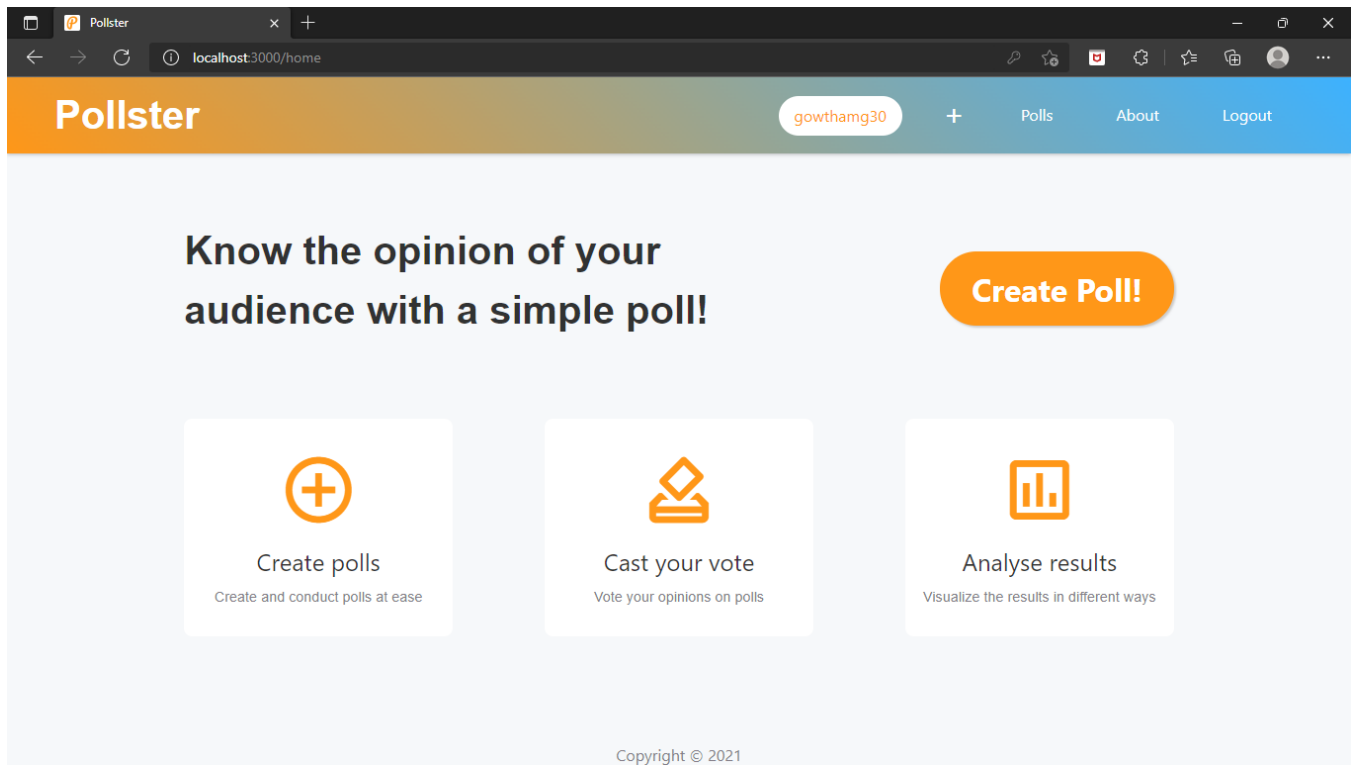


A screenshot of a web browser displaying the Pollster about page. The browser's address bar shows 'localhost:3000/about'. The page has a header with the 'Pollster' logo on the left and 'Login', 'Register', and 'About' links on the right. The main content area features a text box with the following text: 'This is a simple polling web application that helps people to create polls and take surveys, vote for polls and view statistics. This project is created by 2 CSE'23 undergraduates of VNIT, Prabhav and Gowtham.' The footer contains the text 'Copyright © 2021'.

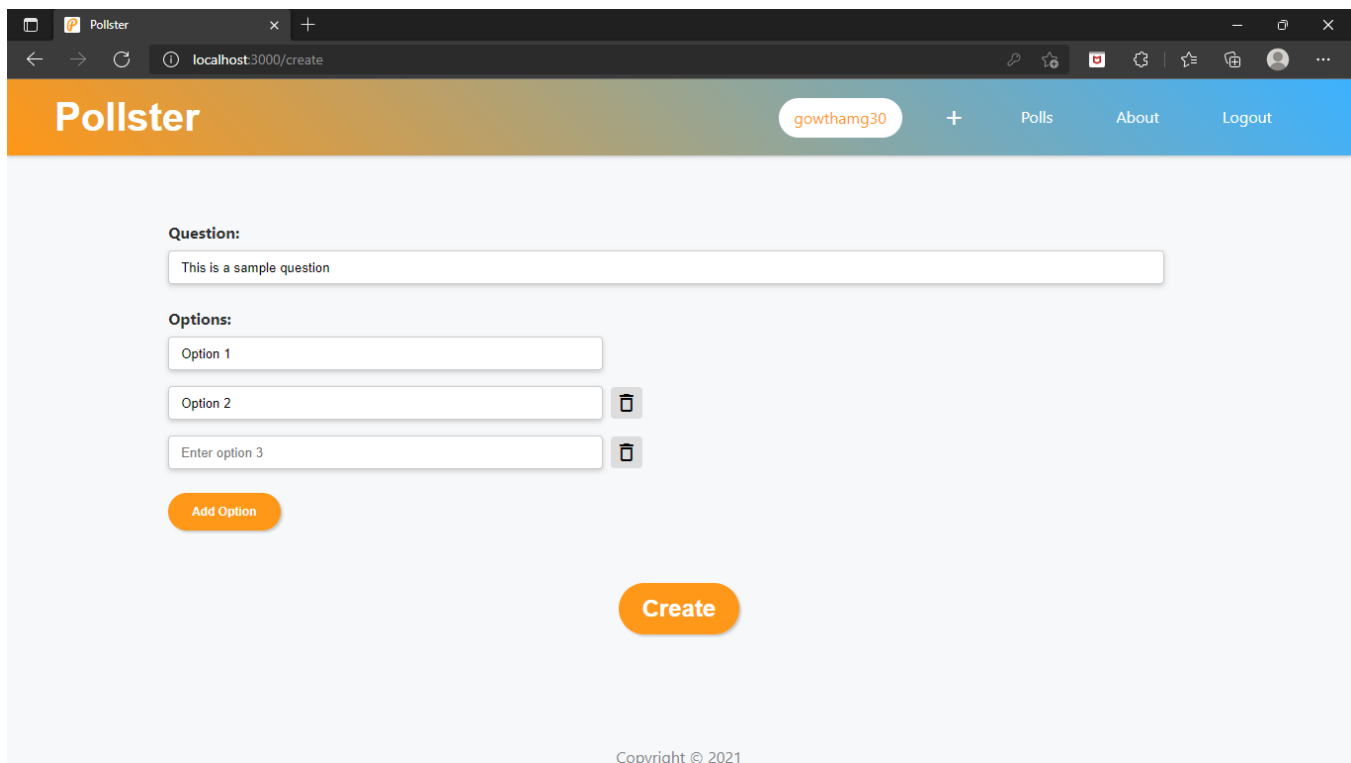
This is a simple polling web application that helps people to create polls and take surveys, vote for polls and view statistics. This project is created by 2 CSE'23 undergraduates of VNIT, Prabhav and Gowtham.

Copyright © 2021

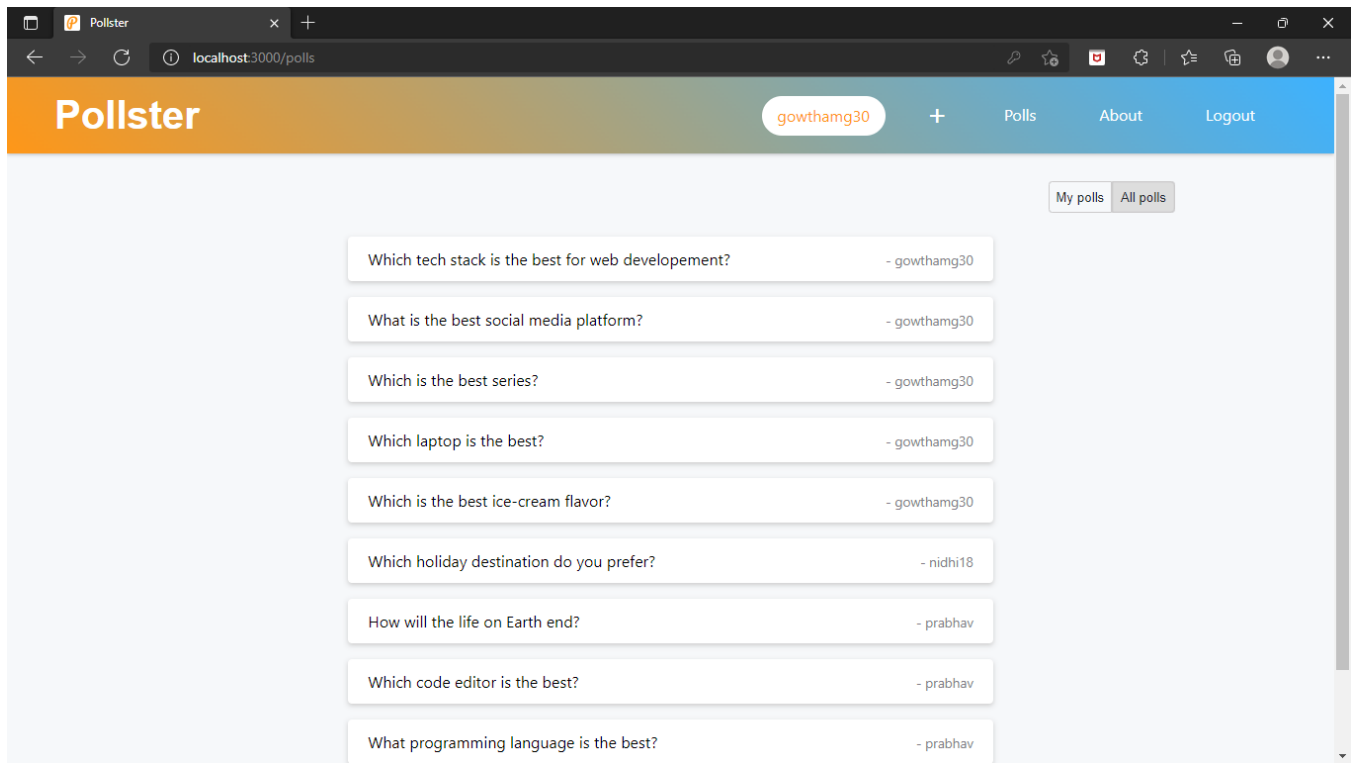
## Home:



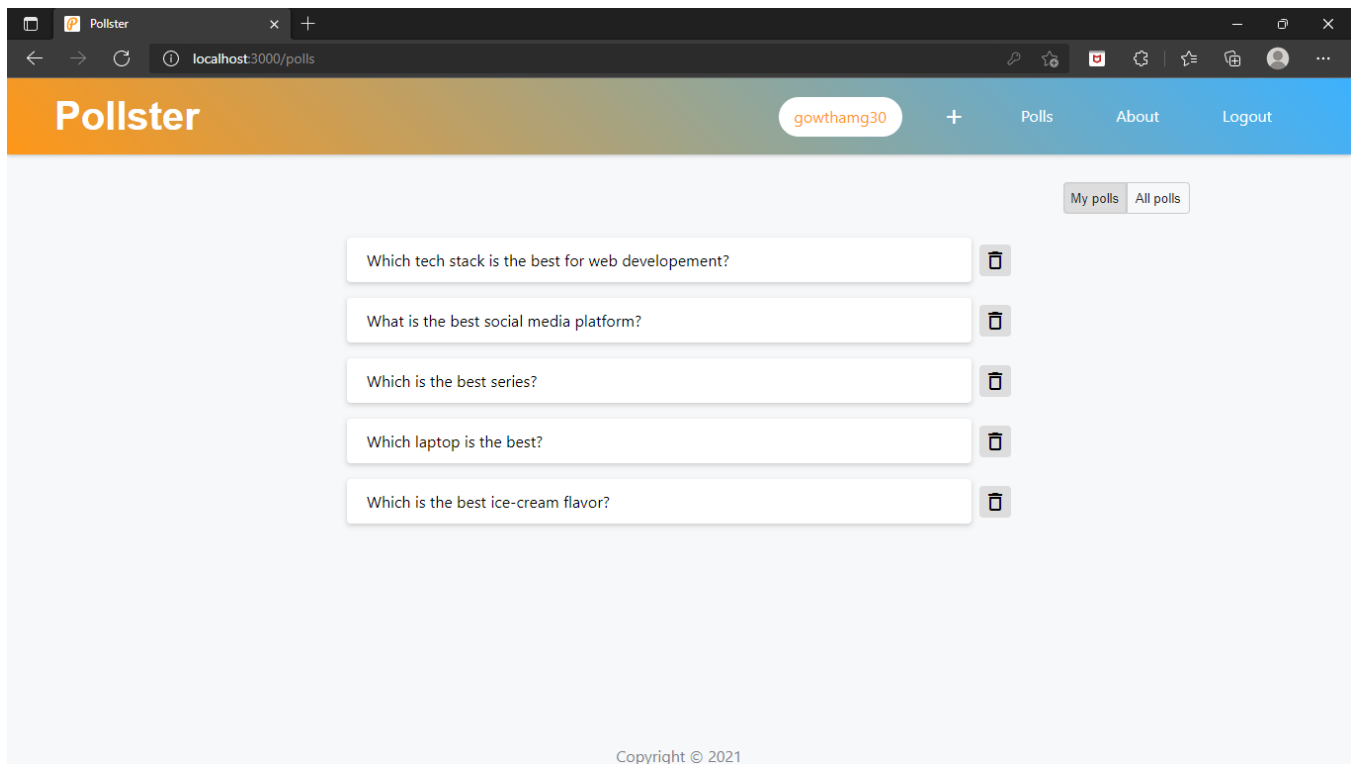
## Create:



## Polls => All polls:



## Polls => My polls:



**Poll:**

Pollster

gowthamg30 + Polls About Logout

Stats

What is the best social media platform?

☐ Instagram

☒ Whatsapp

☐ Facebook

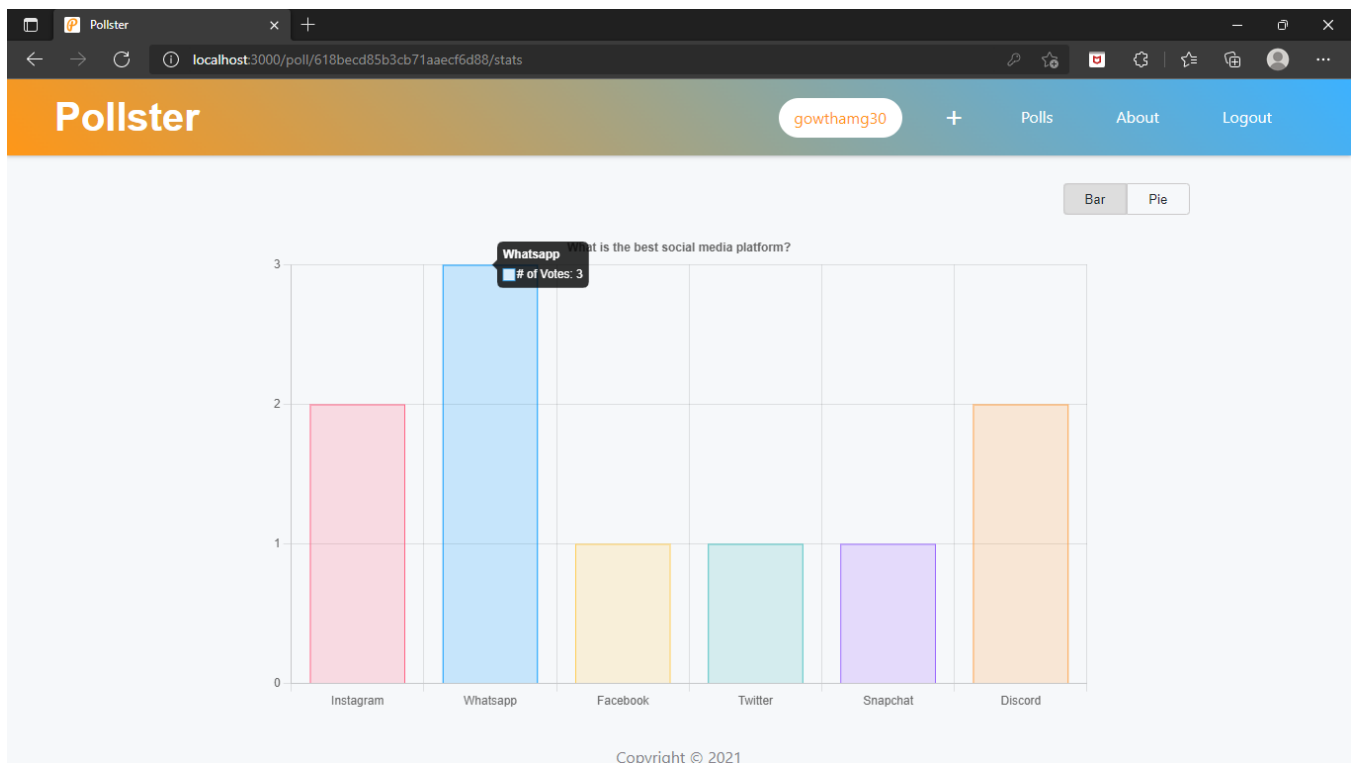
☐ Twitter

☐ Snapchat

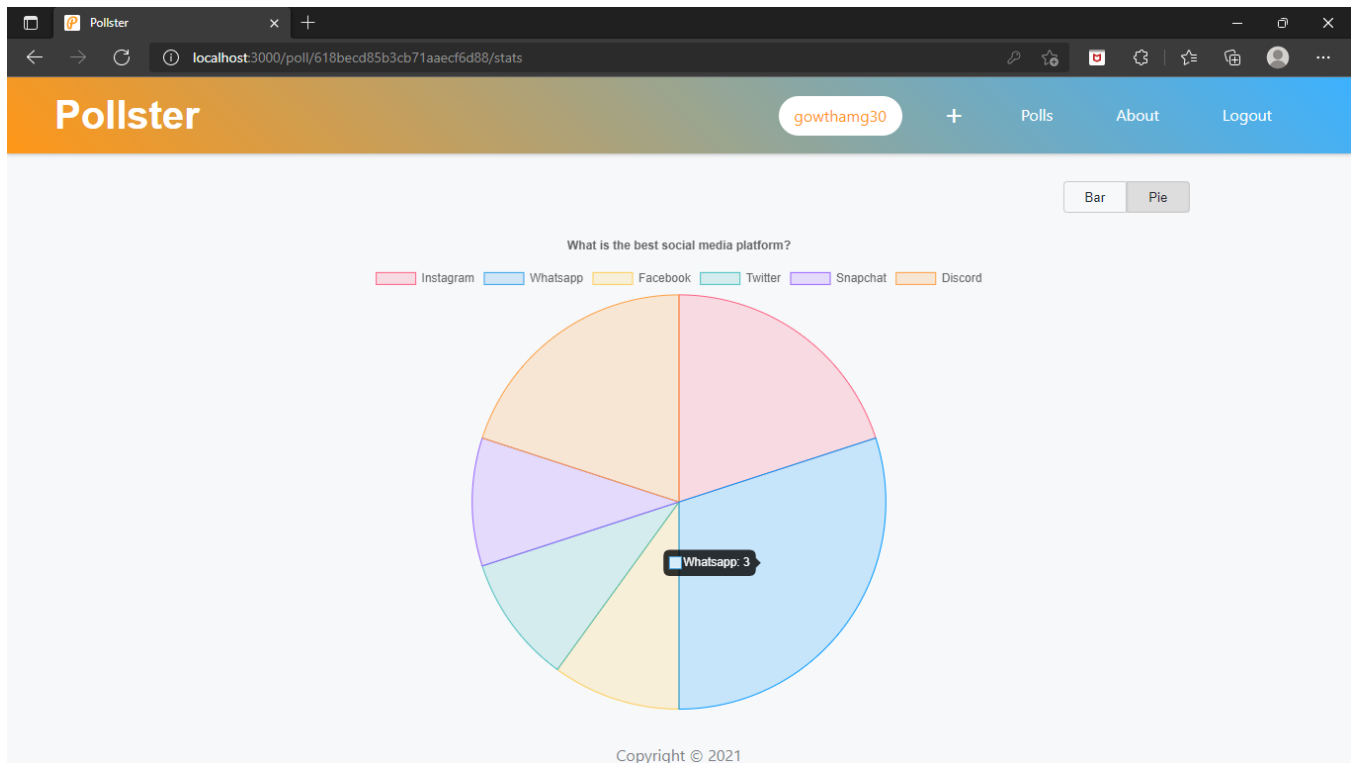
☐ Discord

Vote

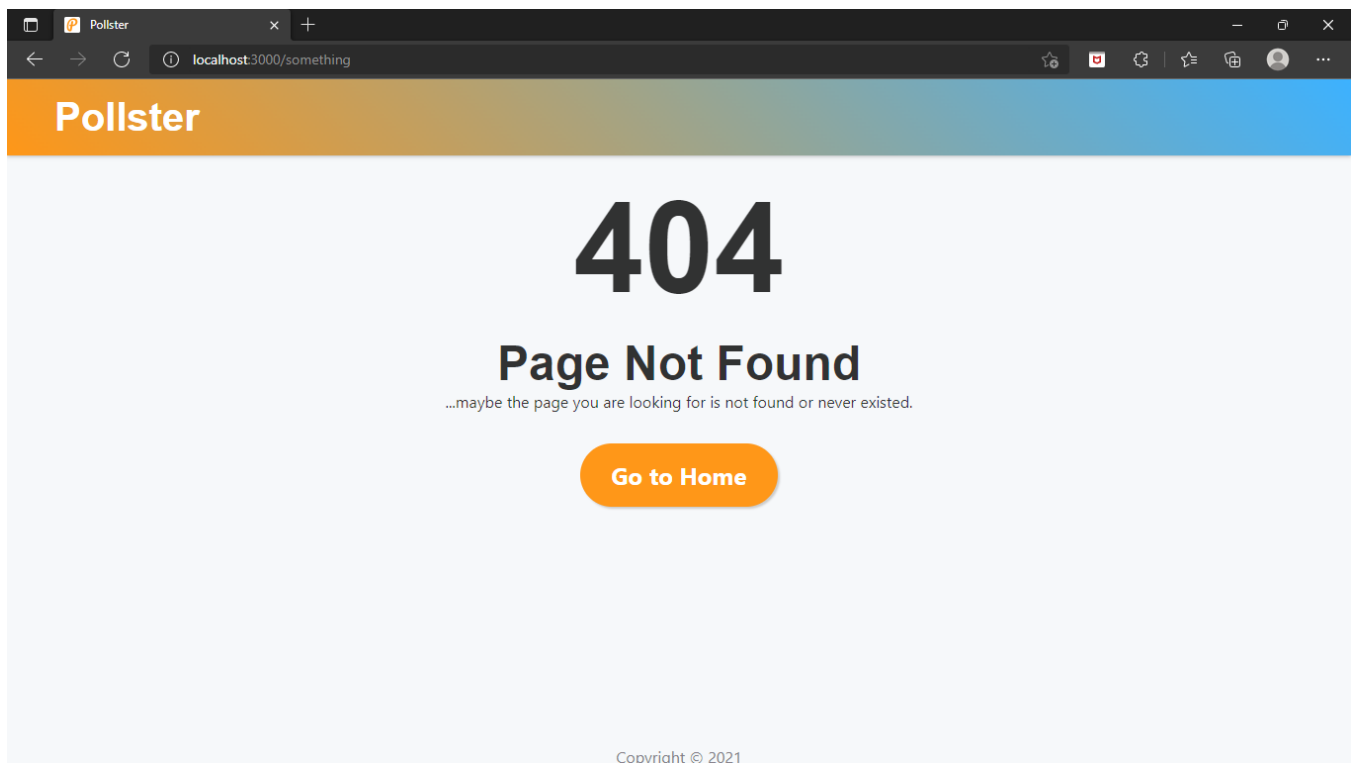
Copyright © 2021

**Stats => Bar:**

## Stats => Pie:



## Page Not Found:



**Error:**

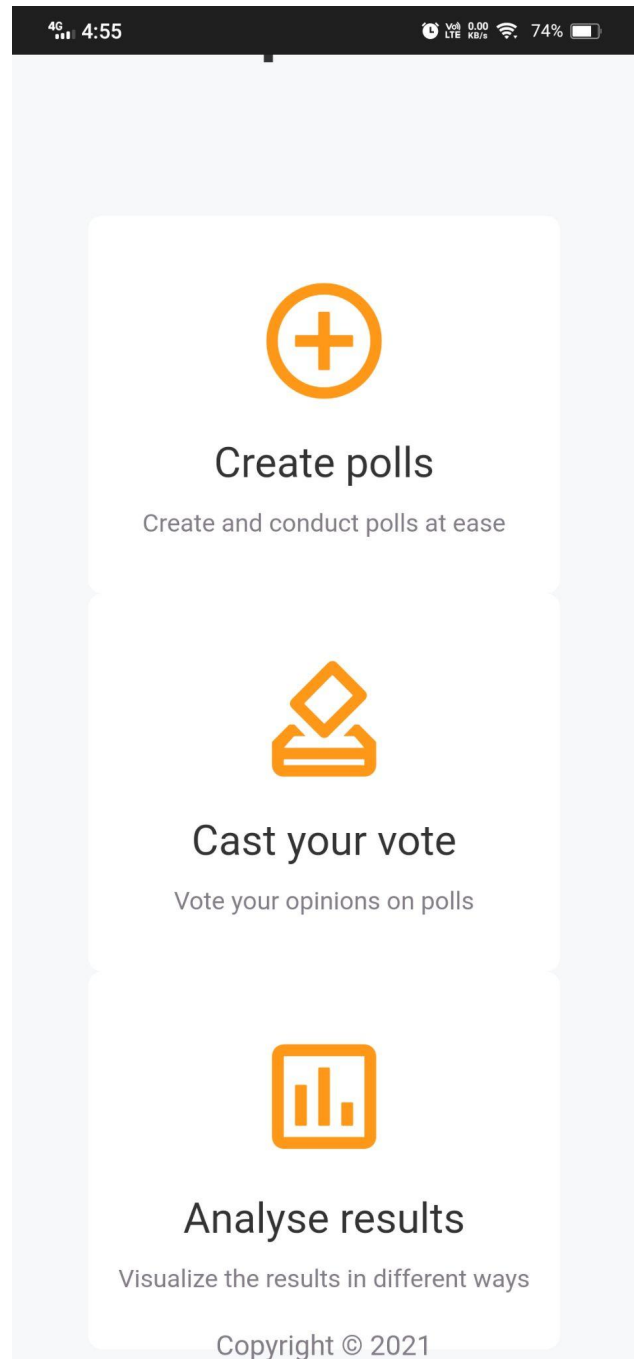
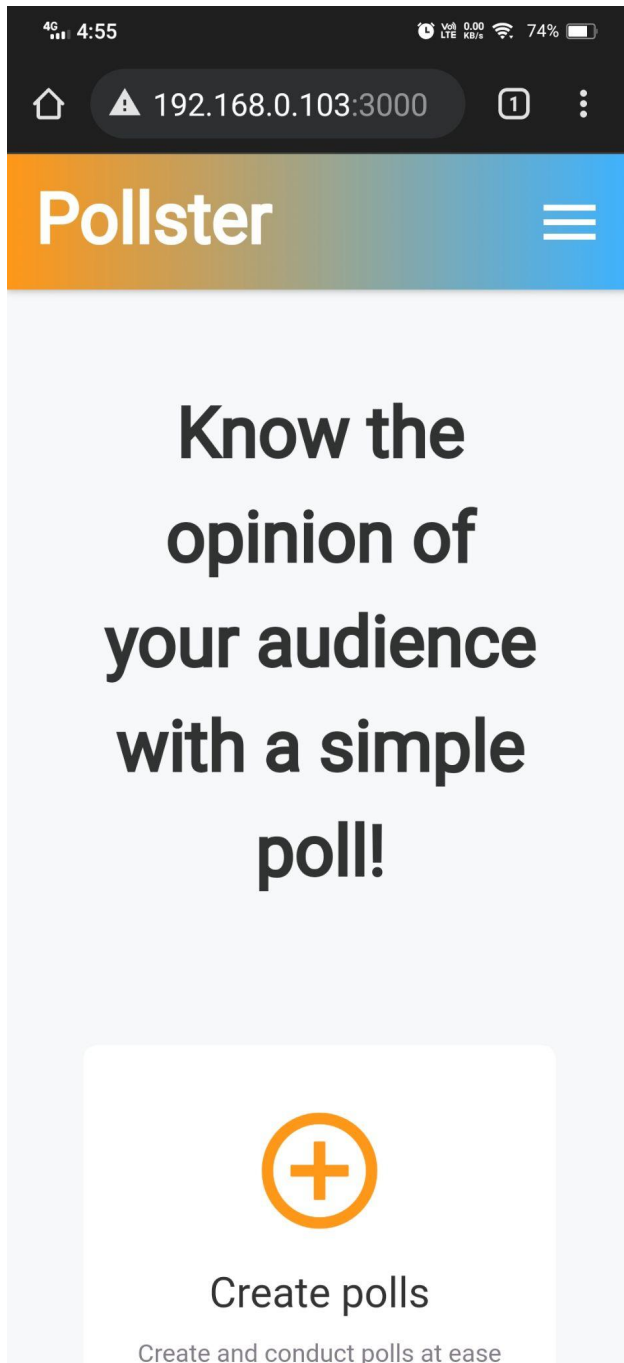
A screenshot of a web browser displaying the Pollster login page. The browser's address bar shows 'localhost:3000/login'. The page has a header with the 'Pollster' logo and links for 'Login', 'Register', and 'About'. The main content area is titled 'Login here!' and contains a form with 'Username' and 'Password' input fields. Below the form, there is a link 'New user? Register here'. A red error box displays two messages: 'Username should not be left empty' and 'Password should not be left empty'. At the bottom of the form is an orange 'Login' button. The footer of the page reads 'Copyright © 2021'.

**Success:**

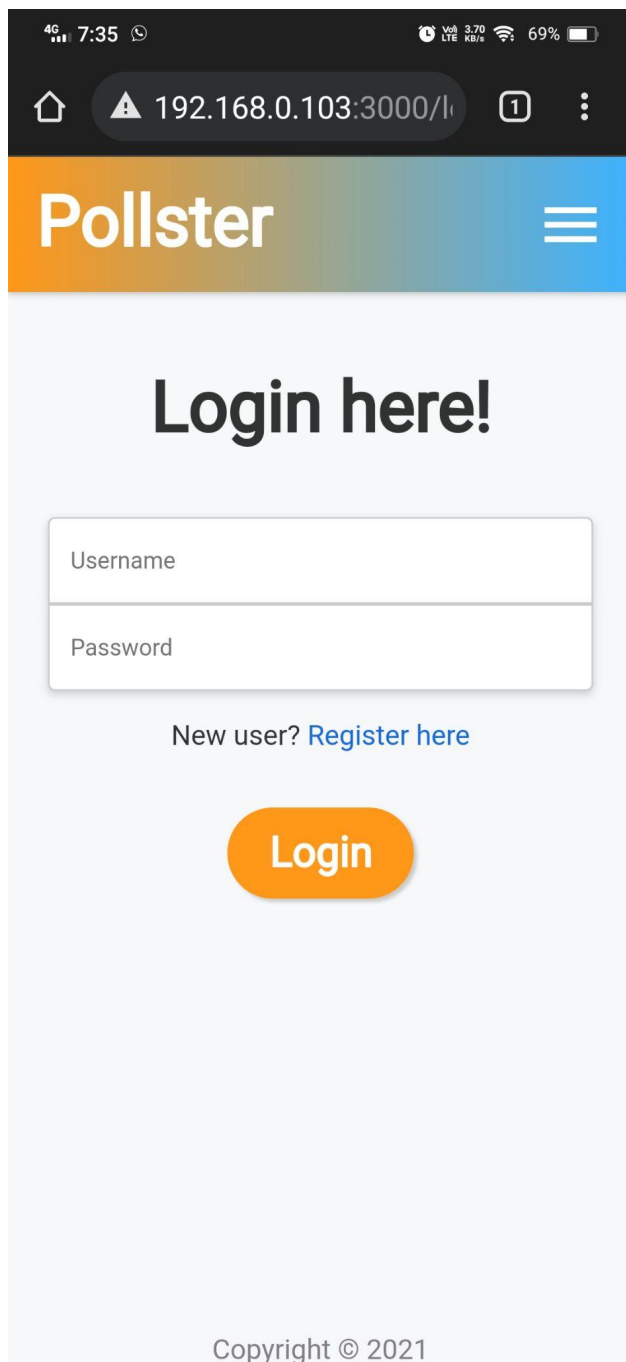
A screenshot of a web browser displaying the Pollster 'create' page. The browser's address bar shows 'localhost:3000/create'. The header includes the 'Pollster' logo, a user profile 'gowthamg30', and links for '+', 'Polls', 'About', and 'Logout'. The main form is titled 'Question:' and has a text input field containing 'Sample question'. Below this, the 'Options:' section has two input fields labeled 'Option 1' and 'Option 2', with a trash icon next to the second one. An orange 'Add Option' button is positioned below the options. A large orange 'Create' button is centered below the form. A light green success message box at the bottom states 'Poll created!'. The footer of the page reads 'Copyright © 2021'.

## Mobile

### Dashboard:



## Login:



The screenshot shows the login interface of the Pollster app. At the top, there's a status bar with the time 7:35 and 69% battery. Below it, a browser address bar shows the URL 192.168.0.103:3000. The app header features the 'Pollster' logo on the left and a hamburger menu icon on the right. The main content area has a large heading 'Login here!' followed by two input fields for 'Username' and 'Password'. Below these fields is a link 'New user? Register here' and a prominent orange 'Login' button. At the bottom, a copyright notice 'Copyright © 2021' is visible.

4G 7:35 69%

192.168.0.103:3000/

Pollster

# Login here!

Username

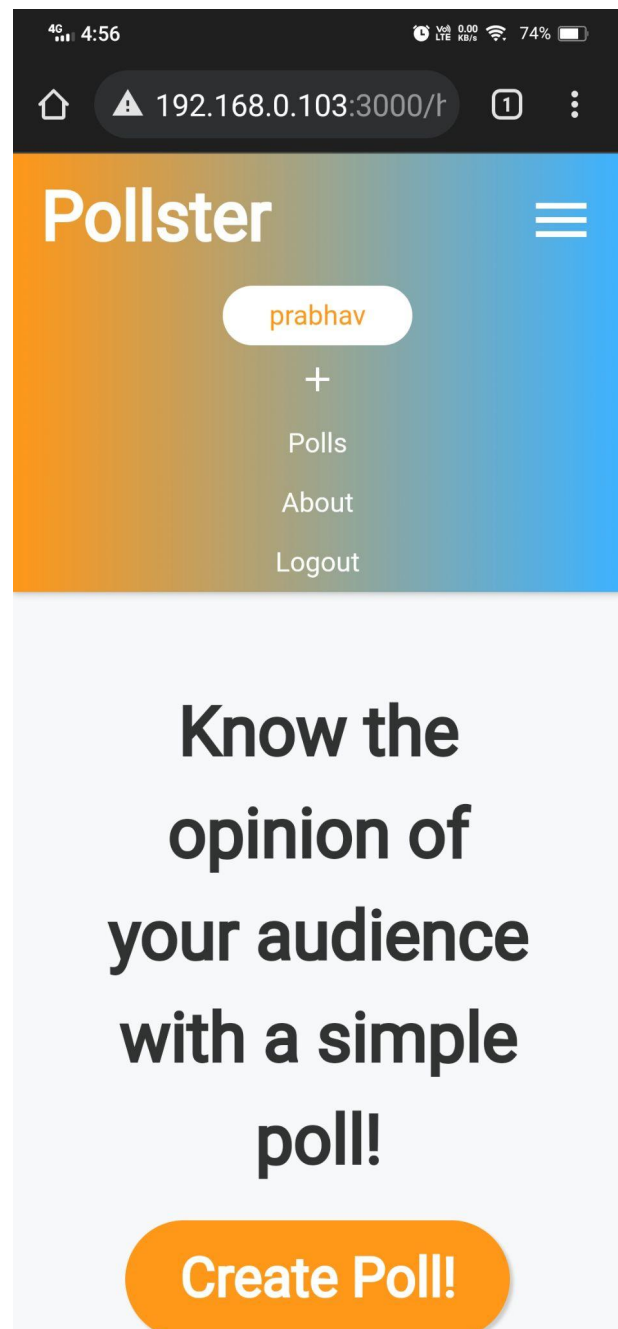
Password

New user? [Register here](#)

Login

Copyright © 2021

## Home:



The screenshot shows the home interface of the Pollster app. The status bar at the top shows the time 4:56 and 74% battery. The browser address bar shows the URL 192.168.0.103:3000. The app header is identical to the login screen. Below the header, the user's name 'prabhav' is displayed in a white pill-shaped button, followed by a '+' icon and a list of menu items: 'Polls', 'About', and 'Logout'. The main content area features a large heading 'Know the opinion of your audience with a simple poll!' and a prominent orange 'Create Poll!' button at the bottom.

4G 4:56 74%

192.168.0.103:3000/

Pollster

prabhav

+

Polls

About

Logout

# Know the opinion of your audience with a simple poll!

Create Poll!



## Create:

4G 4:56 74%

192.168.0.103:3000/c

# Pollster

**Question:**

**Options:**

Add Option

Create

Copyright © 2021

## Poll:

4G 7:41 68%

192.168.0.103:3000/p

# Pollster

Stats

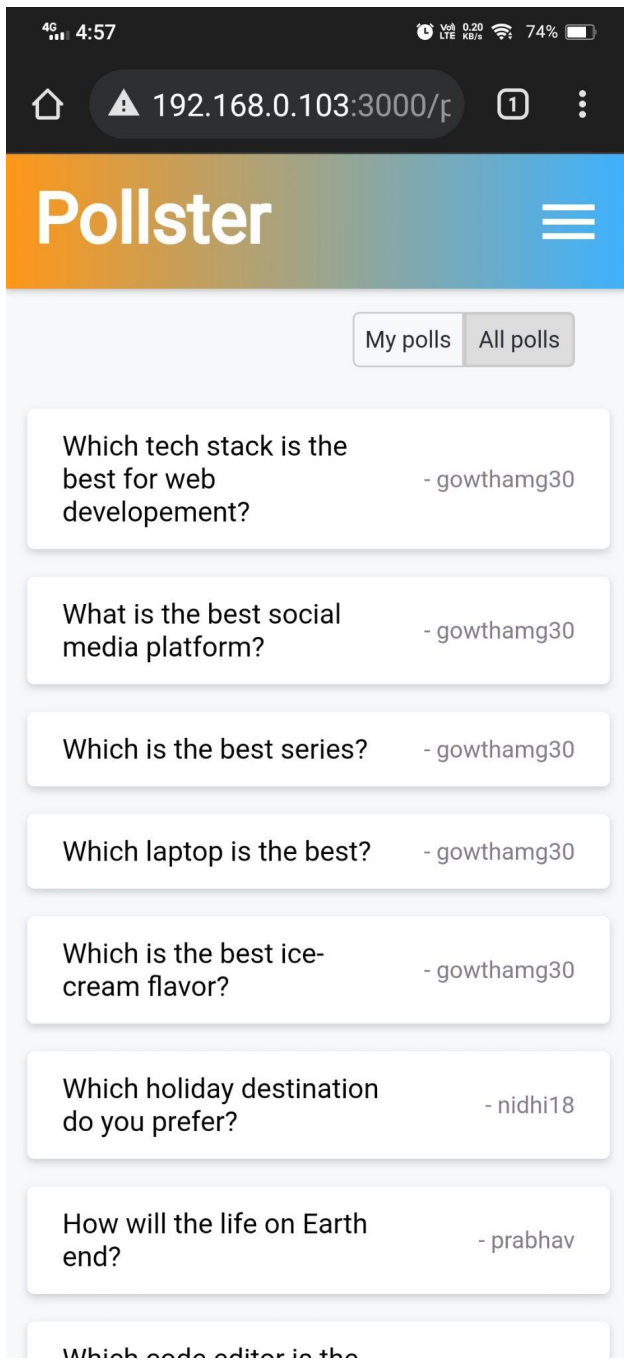
What is the best social media platform?

- ☐ Instagram
- ☐ Whatsapp
- ☐ Facebook
- ☐ Twitter
- ☐ Snapchat
- ☐ Discord

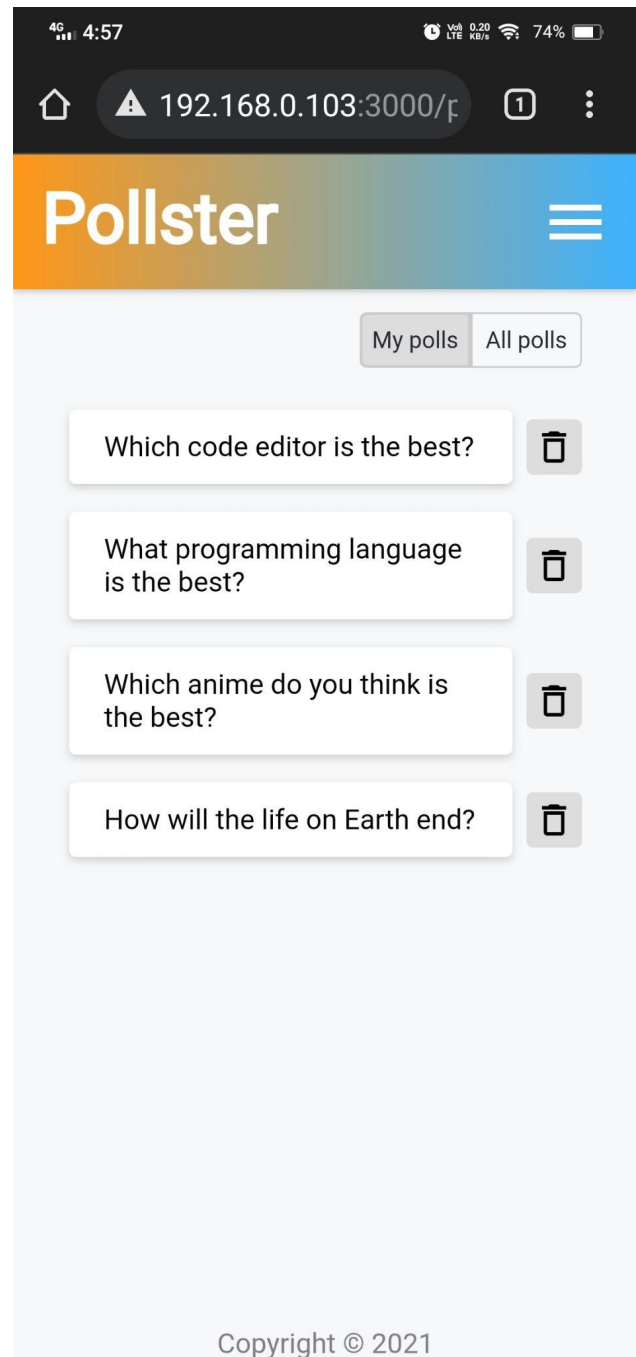
Vote

Copyright © 2021

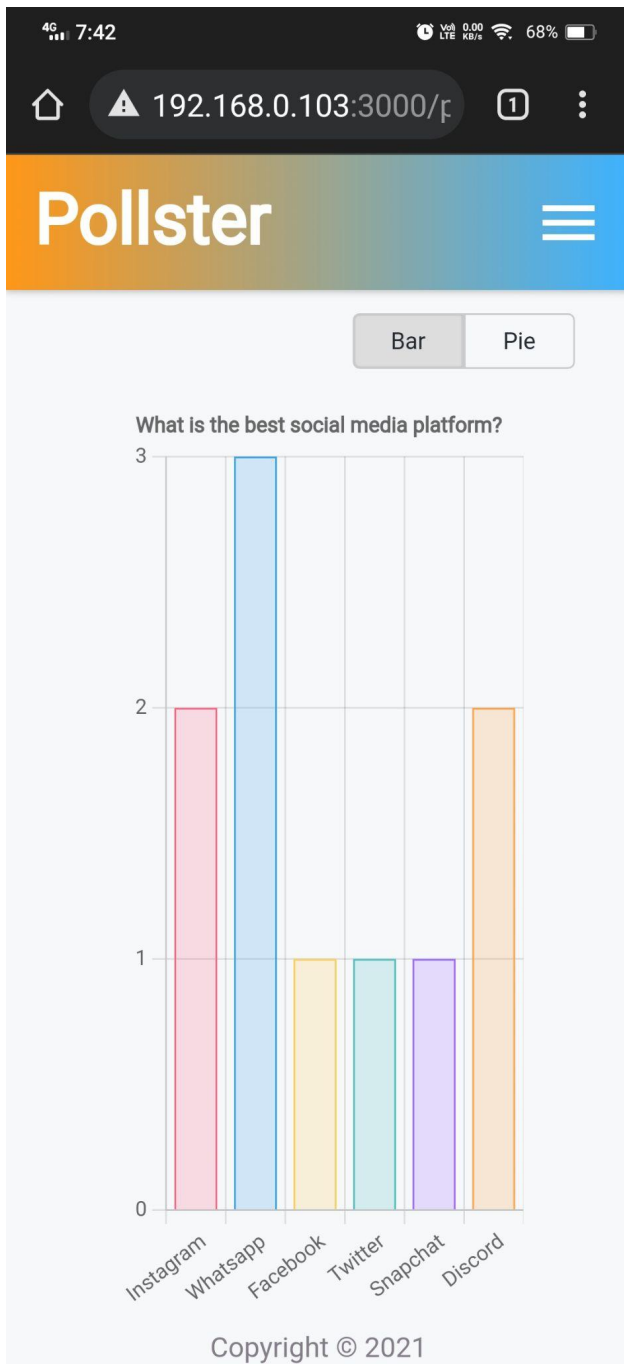
## Polls =&gt; All polls:



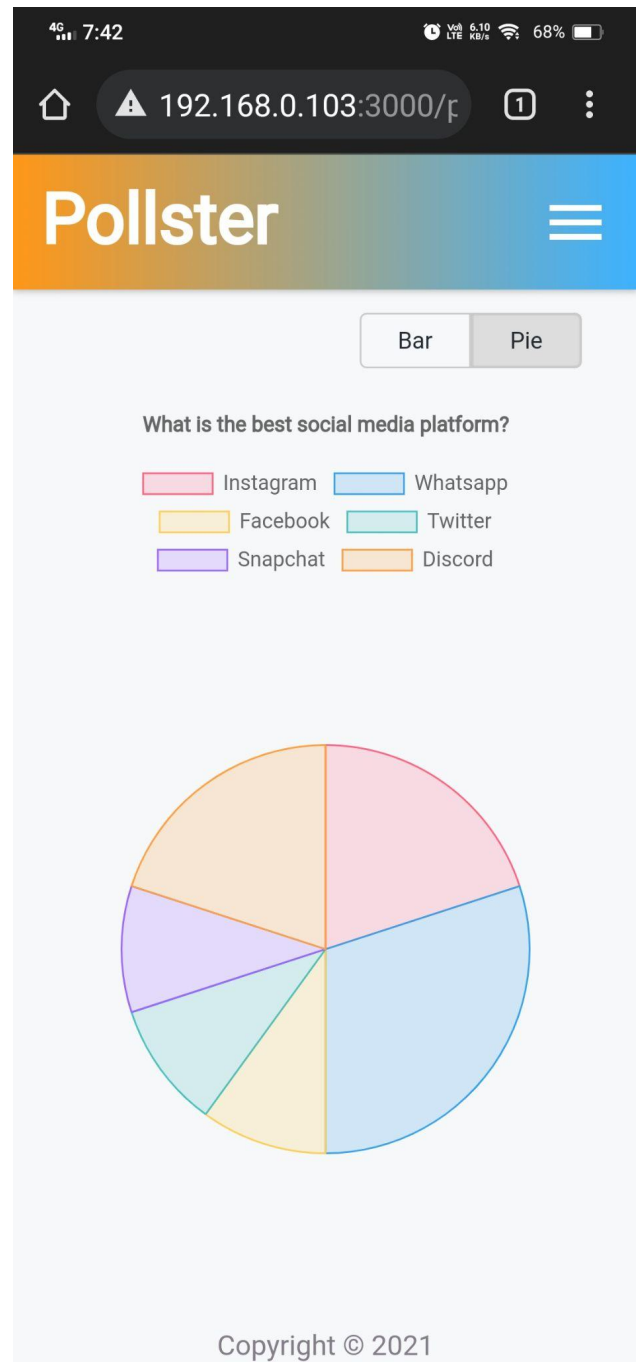
## Polls =&gt; My polls:



## Stats =&gt; Bar:



## Stats =&gt; Pie:



## 6. Challenges faced

- **JWT verification**

- We required the user to not access the API or go to any page other than About, Dashboard, Login and Register when the JWT is expired
- To resolve this issue, we have placed the Navbar component in every page. In the Navbar, we have used the authenticateToken middleware by calling the "api/verify" route. If the JWT is expired, an error is thrown and the user is redirected to the login page.

- **Conditional Redirection**

- Sometimes, we wanted to redirect to some other page conditionally. Some examples are:
  - When a new poll is created, user has to be redirected to the Polls page
  - When a poll is submitted, the user has to be redirected to the Stats page.
- To solve this issue, a "redirect" useState variable is made and is set in useEffect if the appropriate API call in the component is successful.
- <Redirect to="/newPath" /> is returned conditionally w.r.t. "redirect".

- **Uniqueness of voting**

- When the polls were first made, any user would be able to vote a particular poll any number of times.
- This loophole is solved by maintaining an array of voters for every poll in the database. Whenever a user votes, his name is searched in the voters list. If found he is not allowed to vote, otherwise the vote is accepted and his/her name is added into the voters list.

- **Footer**

- The footer is supposed to stay at the bottom of the page irrespective of the above content. But if the content wasn't enough, the footer used to come up
- To solve this we have used position: absolute property and set the bottom offset to 0, to make it stay at the bottom of the page.

## 7. Further Improvements

- Search bar in All polls and My polls.
- Private polls - they are polls that are not visible in All polls, but only accessible using a code.
- Forgot password feature in login page.
- Deadline for a poll.

## 8. Resources

- [Getting Started – React](#)
- [React Router: Declarative Routing for React.js](#)
- [React Router Tutorial | React For Beginners - YouTube](#)
- [JSON Web Token Introduction - jwt.io](#)
- [JWT Authentication Tutorial - Node.js - YouTube](#)
- [axios/axios: Promise based HTTP client for the browser and node.js](#)
- [React components for Chart.js, the most popular charting library](#)
- [Mongoose v6.0.12: API docs](#)