**PROJECT INCREMENT 1**

**PROJECT TITLE**: REAL TIME POLL SYSTEM

**PROJECT TEAM MEMBERS**:

|  |  |
| --- | --- |
| **Name** | **Class ID** |
| RAJASHEKAR | 25 |
| KOUSHIK REDDY | 21 |
| SAI KRISHNA | 14 |

**What is the use of polling system how is it benefited to the public?**

An electronic polling system allows users cast their votes with ease without the hassle and stress of visiting a polling booth. This makes it easily accessible as it can be used by users anywhere in the world. Adding real time functionality to the application improves the user experience as votes are seen in real time.

**How polling system is different from voting system?**

A poll vote can only be used on a motion that can be decided by ordinary resolution. A poll vote cannot be used on a motion that is a secret ballot. When voting on a motion to be decided by ordinary resolution each lot has 1 vote. A poll is a different way of counting votes for the motion.

**GOALS AND OBJECTIVES OF THE PROJECT:**

The main idea of this project is to build a poll system, where the public can vote to their respective teammates, or for their favorites once from which ever the place they are present and the main thing is public can watch this voting as it goes online.

The goal of our project is we’ll build a real time polling application using MongoDB, Angular and charts.js for data visualization.

Using our application users will get to vote for their favorite cricket player in the Indian Premier League.

We’ll send our votes to the server and with the help of MongoDB and update our polls in real time.

We’ll be using these tools to build our application:

* Express
* Node
* MongoDB
* Angular
* Charts

**Express**: Express.js, or simply Express, is a web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

**Charts**: This tool is used for graphical representation of information and data which is called as Data Visualization.

**MongoDB**: MongoDB is a cross-platform document-oriented database program.

Presently during this phase we are developing the code for the poll system and below is the code that we have written.

The work is being shared equally between all the teammates and each one has been provided with a specific task with a timeline.

Component.html file

<div>

<h2>Vote for your player of the season</h2>

<ul>

<li \*ngFor="let player of playerData">

<img [src]="player.image" [alt]="player.name" (click)="castVote(player.shortName)" [ngClass]="getVoteClasses(player.shortName)">

<h4>{{player.name}}</h4>

<p>{{player.runs}} Runs</p>

<p>{{player.average}} Average</p>

</li>

</ul>

</div>

app.component.ts

import { Component, OnInit } from '@angular/core';

import { HttpClient } from '@angular/common/http';

**@Component**({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.scss'],

})

export class AppComponent implements OnInit {

constructor(private http: HttpClient) {}

event = 'vote';

vote = '';

voted = false;

playerData = [

{

name: 'Virat Kohli',

runs: 790,

average: 60,

shortName: 'vkohli',

},

{

name: 'MS Dhoni',

runs: 560,

average: 58,

shortName: 'msd',

},

{

name: 'Rohit Sharma',

runs: 652,

average: 57,

shortName: 'rohit',

},

{

name: "David Warner",

runs: 650,

average: 59,

shortName: 'warner',

},

];

voteCount = {

vkohli: 0,

msd: 0,

rohit: 0,

warner: 0,

};

castVote(player) {

this.http

.post(`http://localhost:4000/vote`, { player })

.subscribe((res: any) => {

this.vote = res.player;

this.voted = true;

});

}

getVoteClasses(player) {

return {

elect: this.voted && this.vote === player,

lost: this.voted && this.vote !== player,

};

}

ngOnInit() {

}

}

App.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

import {HttpClientModule} from '@angular/common/http';

....

**@NgModule**({

declarations: [AppComponent],

imports: [BrowserModule, HttpClientModule],

providers: [],

bootstrap: [AppComponent],

})

....