Muscat College

WEB SERVICES

Code: CSCU9YW

Assignment 2022

Submitted by:

2840080

Submitted on

29 Nov 2022

1. Table of Contents

2. Outline of problems……..3
2.1 Explain the Problem……3
2.2 Assumptions ……..3
2.3 Implemented Functionality……..4
3 Solutions……..4
3.1 Implementation overview……4
4 Server……..5
4.1 Data structure……..5
4.2 REST Principles……..5
4.3 Security……..6
5 Client……..6
5.1 Client User Interface……4
5.2 List The Candidates……..4
6 Admin……9
6.1 Admin User Interface……..9
7 Advanced features……..10
8 Evaluation……10
8.1 Completeness and areas for improvement……..10
8.2 Incomplete Implementation……11
9 Appendix ……12

2. Outline of problems

2.1 Explain the Problem

The task is to create a web service for polling and add vote for voters, to withdraw the information of members of the Bird Society through an external service, and to add or delete the number of votes and display them in the admin service by linking users. The task should follow the design using REST principles. On the other hand, the task should contain two different types of users: the voters who can vote on the birds and see the number of votes and cancel their vote, and the admin who can manage the polling service and view who voted.

1.2 Assumptions

1- Ballots are not anonymous, that is, the service records how individual members vote.

2- ballots are secret, that is, members can inspect their own ballots, but they cannot find out how anyone else voted.

3- administrators cannot inspect individual ballots because the polling service protects the secrecy of ballots.

4- The polling service should store information about Bird of the Year candidates, SAWB members and their ballots in suitable Java data structures, or in a data base.

5- Membership changes over time; some people cease to be members while others newly join the SAWB.

6- To cast a ballot, SAWB members must identify themselves to the polling service by providing the information stored by the membership register.

7- Every member has one vote, only the last ballot counts.

8- The polling service may verify a voter's eligibility to vote by checking their information against their record on the membership register. Ballots from ineligible voters should be rejected.

1.3 Implemented Functionality

For The Voters	1. View all the details about specific candidate (name, scientific name, description). 2. Change their opinion and cast a ballot for a candidate. 3. Inspect which bird they have voted for. 4. Retract their ballot.
For The Administrators	1. Tally the votes cast 2. List the tally for candidate 3. Open and close the poll
Others	1. The polling service verify a voter. 2. Login admin page by using pass and user

3. Solutions

3.1 Implementation overview

First, we run the client service which acts as the first basic infrastructure to start the polling service, the client user interface can display all the candidate birds, create a vote for a bird, view the vote, delete the vote.

A polling web service is designed with a client web service as well as for administrators. and the Spring web libraries are created.

On the other hand, the front-end for users is created using html, JavaScript languages. The web server for the client and the server linked by HTTP Requests.

All the http requests sent by the client via the controller.java, using these requests allows the user to:

1- View candidates

2- Create vote

3- Show the vote.

4- Delete vote.

5- Check the voter

Administrators web can open and close the polling server, as well as show all the members who have voted along with the number of votes for each member.

4. Server

4.1 Data structure

The following figure shows the data structure, consisting of limited tables, using a HashMap.

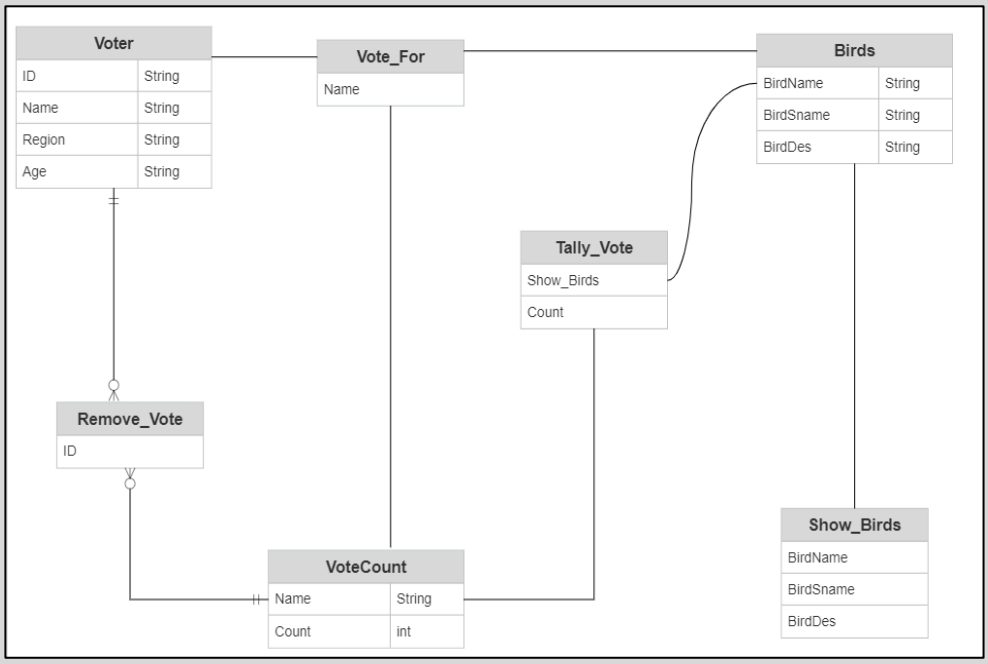


Figure 1 ERD For The service

The central structure of this assignment it is (Birds, Voter, VoteCount) which holds all the information's about the candidates and the voter

It is worth noting that the candidates and Voters are stored in a data structure (HashMap). Both voter and candidates are stored using a HashMap.

```
//Create HashMap For Two Models
private Map<String, Birds> Birddb;
private Map<String, Voters> VoteDb;
```

4.2 REST Principles.

As mentioned earlier, the task is designed by following the REST system because this system can store all resources. Initially, requests are made using JSON because both the Java language and JavaScript provide easy parsing between JSON and class objects.

When storing the results of candidate display requests, for example, it will lead to raising the performance of the server, and on the other hand, this will reduce the level of pressure on the server, and at the same time the server will be able to provide other functions.

Finally, the web service interface is uniformly designed to make it easier for the user. Quick access to candidates. To access the candidates (ding) is used, for the administrator (admin). When requesting resources from the server, the clint can use the following ways:

1- **GET Request**: This request is used to request all candidates the member has voted. In the current task the URL was used to request existing candidates http://localhost:8080/ding/ID-members.

2- This request is used to show all list in admin http://localhost:8080/Admin/.

3- **POST Requests:** POST is used to do a Nord send to the web server, e.g., create a new vote for candidates, which on receiving is then converted by Spring to an object in Java.

4- **PUT Requests:** This type of request is used to refresh; in the current task a PUT request is used to close and open polling via the admin page.

5- **DELETE Requests:** The vote will be removed from the server; it is used on the client page only.

4.3 Security

The application carries out the process of securing sensitive functions mainly by using the username and password, the password must be authenticated before entering the administrator page

```
//Set login user and pass
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
.withUser("Mai").roles("2840080").password("{noop}admin");
}
```

2. Client

5.1 Client User Interface

One of the requirements for creating this task is to design a main interface for the client. This requirement was implemented using HTML, where <Input> was used to enter the details of the members, in addition to a group of <Button> to create a group of buttons with different functions, and finally <Select> to display the names of the candidates.

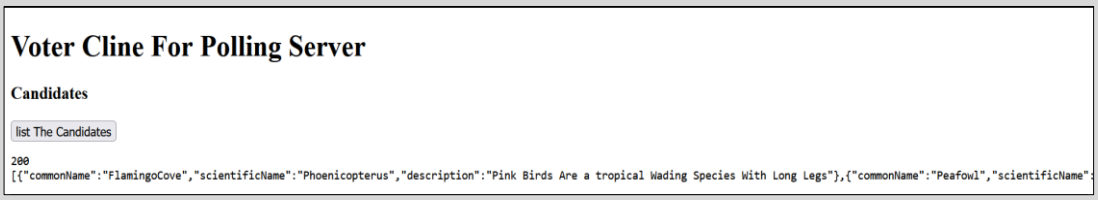
Figure 2 Client User Interface

- **List The Candidates:**

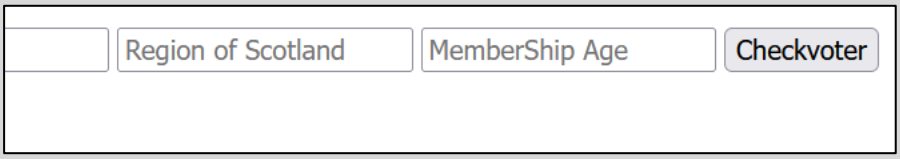
This button allows you to view all the list of candidate's birds for this year. This button will display the details of the candidates, the common name, the scientific name, and a short summary of that candidate. This feature has been implemented using @GetMapping as it will call all candidates.

```
@GetMapping("/ding")//Show The List Of Candidates in Clint Page
public ResponseEntity<List<Birds>> getAllCandidates()
{
    return new ResponseEntity(ws.getAllBirds(), HttpStatus.OK); //Ok = 200
}
```

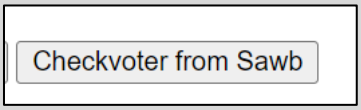
Result



- **Check Voter:** This button allows checking whether the member exists or not.



- **Check Voter:** This button will check member info from SAWB server.



- **Vote For:** This button allows to add vote for the one candidate for each member, The Add Vote feature allows creating a new vote for the candidate bird. To implement this feature, the user must fill in the required fields and then click on the Add Vote button to complete the process of creating a vote. If the required fields are empty, the server will return a 404 status, meaning that the user must add the required field to create a voting process.

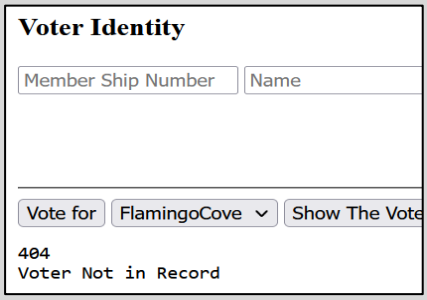


Figure 3 Post Field

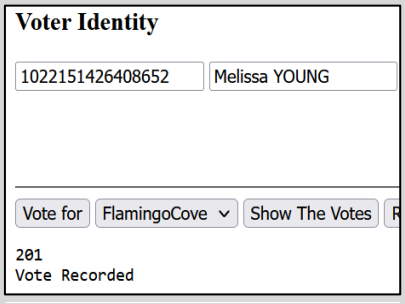


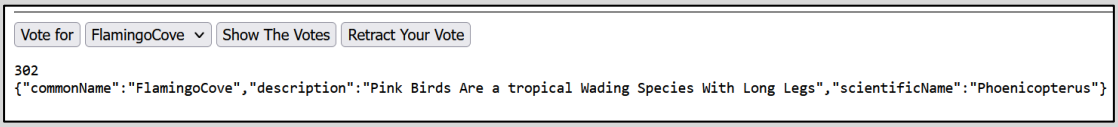
Figure 4 Post Created

An if statement is added to check two things, whether the polling service is open or not, and whether the user has added the required data or not, and then @POSTMapping is called, post is a means of creating a new poll.

It will then create a new vote and it will be added to the HashMap, where the entered data will be stored in an array and entered the HashMap.

```
VoteDb.put(vote.getId(),vote);
```

- **Show The Vote:** This button allows to view which candidate member voted



- **Retract The vote:** This button allows to delete candidates.

Through the delete button, the user is allowed to delete the vote he made. When clicking on the delete button, a signal will be sent to the service to remove the member's number from the database by using the remove method.

```
// React Vote method
public void removeVote(String ID) {
    VoteDb.remove(ID);
}
```

3. Admin

6.1 Admin User Interface

on the other hand of this task, creating design a main interface for the Admin. This requirement was implemented using HTML, where <Button> was only used to tally voter, as well as open and close the poll server.

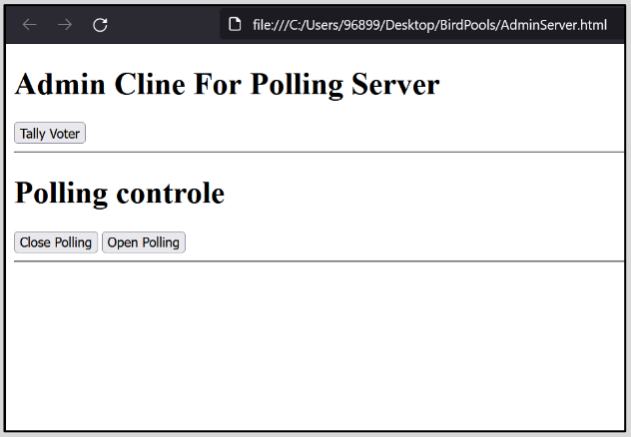
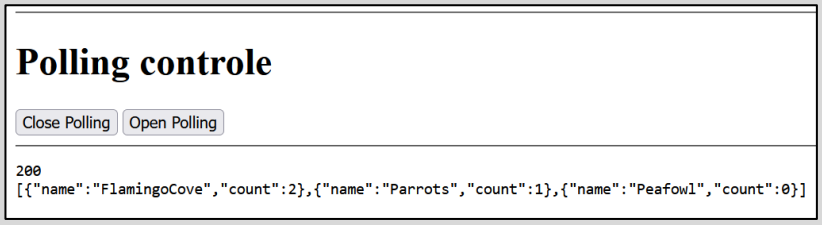


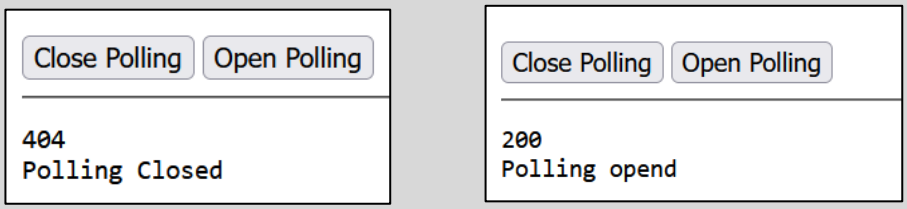
Figure 5 Admin User Interface

1- Tally Voter



This button allows to display all candidates in addition to the number of votes cast by members, when members vote on the client's page, these votes are saved in the VoteDb.Java, When the Tally Voter button is pressed on the Admin's page, the service reads the HashMap and then puts it in an Array List, using (Loop) this tick is read and adding 1 if this candidate is found, Finally, these results are saved in an array.

2- Close/Open Polling Server



These buttons work by using PUT Mapping

6 Advanced features

All these requests that were mentioned above will display the HTTP status code, which symbolizes the result of the operation. In the current assignment, there are some of status code have used in this task

Http Status Code	Meaning	Trigger
200	Ok	GET/DELET/PUT successfully
201	Created	POST
404	Member_ID does not exist	GET/PUT failed
302	List Founded	Found

7 Evaluation

7.1 Completeness and areas for improvement

I was able to implement all the basic features in the application in addition to adding some features that I mentioned earlier. These features were discussed with the addition of screenshots. The following is a list of the features that have been added to the application:

1- Spring RESTful web service.
2- An internal database to store the info for members and candidates. this was made with HashMap.
3- The ability to (Check voter, check voter from SWAB, show candidates, add vote, remove vote, show vote, open and close polling, tally voter.
4- When an action is performed, relevant messages are displayed on the screen if the action was successful or not.
5- View all candidates in the database.
6- The member can be researched using their ID.
7- CURD (GET, PUT, POST, DELETE) requests can be send and receive, and handling for the correct responses.

8- The user interface is created for the client, and for the administrator. The client interface includes delete, view, and add. While the administrator interface includes open, close polling and viewing.

9- The info of members will be showing also in text box.

7.2 Incomplete Implementation

I succeeded in processing and implementing all the requirements that were mentioned in this task, as it was explained in the previous sections, the project is working at full capacity, as all functions have been completed and added in the web service.

Command to decline the security

```
chrome.exe --user-data-dir="C://Chrome dev session" --disable-web-security
```

8 Appendix

8.1 HTML & JavaScript Code:

Polling Server Clint

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="utf-8"/>
        </head>
  <body>
    <title>Client Polling Service</title>
      <h1>Voter Cline For Polling Server</h1>

      <h3>Candidates</h3>
          <div>
      <button type="button" onclick="show_all_Candidates();">list The
Candidates</button>

          <pre
          id="Candidates_output">
          </pre >


           <hr/>
         <h3>Voter Identity</h3>
         <input type="text" id="Mebernumber" placeholder="Member Ship
Number"/>
         <input type="text" id="Name" placeholder="Name"/>
         <input type="text" id="Region" placeholder="Region of
Scotland"/>
         <input type="text" id="age" placeholder="MemberShip Age"/>

                  <button type="button"
onclick="check_request();">Checkvoter</button>
                   <button type="button"
onclick="check_sawb_request();">Checkvoter from Sawb</button>

                <pre id="check_request" >

               </div>

        <hr/>
    <button type="button" onclick="vote_request();">Vote for
</button>

          <select name="Birds common name" id="canditname">

</select>

      <button type="button" onclick="Show_vote_request();">Show The
Votes</button>
      <button type="button" onclick="retract_request();">Retract Your
Vote</button>
           <pre
           id="Vote_OUTPUT">
```

```
            <pre
           id="retract_OUTPUT">
            </div>

   <script>



          function dataBody(data) {
        return data ? JSON.parse(JSON.stringify(data)) : '';
    }

           function vote_request() {
           const Voteout = document.getElementById('Vote_OUTPUT');
//set output display
        const voterid = document.getElementById("Mebernumber").value;
//get the ID number
               const bird =
document.getElementById("canditname").value; //set the candidantes in
the option list
               const url = 'http://localhost:8080/polling/';
               let data = {
            id: voterid,
            name: bird,
        }
                  fetch(url, {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify(data),
        }).then(async (response) => {
          let data = await response.text();
          Voteout.innerText =
`${response.status}\n${dataBody(data)}`;
        }).catch(error => Voteout.innerHTML = error);


      }



          function populateOptions(options) {
        const select = document.getElementById('canditname');
        // Reset options
        select.innerHTML = "";
        JSON.parse(options).forEach(d=> select.add(new
Option(d.commonName)));
    }

        function show_all_Candidates() {
        const output = document.getElementById('Candidates_output');
        let url = 'http://localhost:8080/polling/'
        fetch(url).then(async (response) =>
               {let data = await response.text();
          output.innerText = `${
                 response.status
                 }\n${
                 dataBody(data)
                 }`;
          populateOptions(data);
```

```
        }
               ).catch(error => output.innerHTML = error);

               }
               function Show_vote_request() {
        const output = document.getElementById('Vote_OUTPUT'); //set
display the output
                const voterid =
document.getElementById("Mebernumber").value; //get ID members
        let url = 'http://localhost:8080/polling/'+voterid
        fetch(url).then(async (response) => {
          let data = await response.text();
          output.innerText = `${response.status}\n${dataBody(data)}`;

        }).catch(error => output.innerHTML = error);
    }

         function retract_request(){
         const output = document.getElementById('Vote_OUTPUT'); //set
display the output
          const retract =
document.getElementById("Mebernumber").value; //get ID members
           let url = 'http://localhost:8080/polling/'+retract
           fetch(url, {
            method: 'DELETE',
        }).then(async (response) => {
          let data = await response.text();
          output.innerText = `${response.status}\n${dataBody(data)}`;
        }).catch(error => output.innerHTML = error);
        }
          function check_request() {

        const voterid = document.getElementById("Mebernumber").value;
//get the ID number
        const output = document.getElementById('check_request');

               const Voterage = document.getElementById('age');
               const Votergin = document.getElementById('Region');
               const Votename = document.getElementById('Name');
        const url = 'http://localhost:8080/info/'+voterid;
        // send request and display and process response
        fetch(url).then(async (response) =>
               {let data = await response.text();
          output.innerText = `${
                  response.status
                  }\n${
                  dataBody(data)

                  }`;
                  const inf = JSON.parse(data);
                  Voterage.value = inf.age;
                  Votergin.value = inf.region;
                  Votename.value = inf.name;

        }
               ).catch(error => output.innerHTML = error);


      }
```

```
          function check_sawb_request() {

        const voterid = document.getElementById("Mebernumber").value;
//get the ID number
        const output = document.getElementById('check_request');

               const Voterage = document.getElementById('age');
               const Votergin = document.getElementById('Region');
               const Votename = document.getElementById('Name');
        const url =
'https://pmaier.eu.pythonanywhere.com/sawb/member/'+voterid;
        // send request and display and process response
        fetch(url).then(async (response) =>
               {let data = await response.text();
          output.innerText = `${
                 response.status
                 }\n${
                 dataBody(data)

                 }`;
                 const inf = JSON.parse(data);
                 Voterage.value = inf.member.age;
                 Votergin.value = inf.member.region;
                 Votename.value = inf.member.name;

        }
               ).catch(error => output.innerHTML = error);


      }


    </script>
  </body>
</html>
```

Admin Server

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="utf-8"/>
    <title>Admin Polling Server </title>
  </head>
  <body>
      <h1>Admin Cline For Polling Server</h1>

         <div>
      <button type="button" onclick="show_all_Candidates();">Tally
Voter</button>
         </div>
         <div>
         <hr/>

         <h1>Pool controle</h1>
```

```
          <button type="button" onclick="close_pool();">Close
Polling</button>
          <button type="button" onclick="open_pool();">Open
Polling</button>
          <hr/>
   <pre
          id="Candidates_output">
          </pre >
          </div>
          <div>

     </div>

   <script>
          function dataBody(data) {
        return data ? JSON.parse(JSON.stringify(data)) : '';
    }



           function show_all_Candidates() {
        const output = document.getElementById('Candidates_output');
        let url = 'http://localhost:8080/Admin'
        fetch(url).then(async (response) =>
               {let data = await response.text();
          output.innerText = `${
                  response.status
                  }\n${
                  dataBody(data)
                  }`;

        }
               ).catch(error => output.innerHTML = error);

               }

               function open_pool() {
                const output =
document.getElementById('Candidates_output');
                const o = 'open';
                  let url = 'http://localhost:8080/Admin/'+o;
              fetch(url, {
            method: 'PUT',
        }).then(async (response) => {
          let data = await response.text();
          output.innerText = `${response.status}\n${dataBody(data)}`;
        }).catch(error => output.innerHTML = error);




               }



               function close_pool(){

               const c = 'close';
```

```
                const output =
document.getElementById('Candidates_output');
                let url = 'http://localhost:8080/Admin/'+c;
               fetch(url, {
             method: 'PUT',
         }).then(async (response) => {
           let data = await response.text();
           output.innerText = `${response.status}\n${dataBody(data)}`;
         }).catch(error => output.innerHTML = error);
                }

    </script>
  </body>
</html>
```

8.2 Java Code:

BirdDemoApplication.java

```
package pool; //Package Name

import pool.model.Birds;
import pool.service.BirdPoService;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
//----------------------Import Classes

@SpringBootApplication //launch a Spring application from main method
public class BirdDemoApplication
{//Start WelcomeDemoApplication Class

   public static void main(String[] args)
   {//Start Main Method Tp Lunch The Class
      SpringApplication.run(BirdDemoApplication.class, args);
   }//End Main Method

   @Bean // Use Bean TO Add a Returned Object Of The Method.
   public CommandLineRunner initDB(BirdPoService ps)
   {
      return (args) ->
      {
         ps.addBird(new Birds("FlamingoCove", "Phoenicopterus", "Pink Birds Are a
tropical Wading Species With Long Legs"));
         ps.addBird(new Birds("Parrots", "Psittaciformes", "It Is Raucous Birds Of The
Family Psittacidae."));
         ps.addBird(new Birds("Peafowl", "PavoCristatus", "Is Native To The Indian
Subcontinent."));
      };
   }

}//End Class
```

AuthenticationEntryPoint.java

```
package pool.config;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import
org.springframework.security.core.AuthenticationException;
import
org.springframework.security.web.authentication.www.BasicAuthe
nticationEntryPoint;
import org.springframework.stereotype.Component;

@Component
public class AuthenticationEntryPoint extends
BasicAuthenticationEntryPoint
{//Start AuthenticationEntryPoint class

    /**
     *
     * @param request
     * @param response
     * @param authEx
     * @throws IOException
     */

    //Handling The HTML File Code
    @Override
    public void commence(HttpServletRequest request,
HttpServletResponse response, AuthenticationException authEx)
      throws IOException {
        response.addHeader("WWW-Authenticate", "Basic realm="
+getRealmName());

response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        PrintWriter writer = response.getWriter();
        writer.println("HTTP Status 401 - " +
authEx.getMessage());
    }

    @Override
    public void afterPropertiesSet()  {
        setRealmName("DeveloperStack");
        super.afterPropertiesSet();
    }

}//End AuthenticationEntryPoint class
```

SpringSecurityConfig.java

```
package pool.config;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.
builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.Ht
tpSecurity;
import
org.springframework.security.config.annotation.web.configurati
on.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurati
on.WebSecurityConfigurerAdapter;


@Configuration
@EnableWebSecurity
public class SpringSecurityConfig extends
WebSecurityConfigurerAdapter {

      @Autowired
      private AuthenticationEntryPoint authEntryPoint;

      @Override
        // disable CSRF so PUT POST Methods  work  without
proplem
      protected void configure(HttpSecurity http) throws
Exception {
            http.csrf().disable().authorizeRequests()
                        .anyRequest().authenticated()
                        .and().httpBasic()

      .authenticationEntryPoint(authEntryPoint);
      }

        //Set login user and pass
      @Autowired
      public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
            auth.inMemoryAuthentication()

.withUser("Mai").roles("2840080").password("{noop}admin");
      }

}
```

BirdController.java

```
package pool.controller;
//----------------------Package Name

import pool.model.Birds;
import pool.model.Voters;
import pool.service.BirdsServiceImpl;
import pool.service.BirdPoService;
import java.io.IOException;
import java.net.MalformedURLException;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException; // check
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
//----------------------Import Classes

@RestController //Allows To Handle All REST APIs


public class BirdController
{//Start Class
    //keep Reference For Service Class.

    private final BirdPoService ps;
    String ctr ="close"; // value to control the pool

    public BirdController(BirdPoService ps)
    {//Start Constructure
        this.ps = ps; //Invokkes Parameter
    }//End Constructure

    //---------------------------[REST Principles]------------
----------------\\
    @GetMapping("/polling")//Show The List Of Candidates in
Clint Page

    public ResponseEntity<List<Birds>> getAllCandidates()
    {
        return new ResponseEntity(ps.getAllBirds(),
HttpStatus.OK); //Ok = 200
    }
```

```
      @GetMapping("/Admin") //Show All List Of Voter in Admin
Page
    public ResponseEntity<List<Voters>> Tally()
    {
        return new ResponseEntity(ps.getAllVote(),
HttpStatus.OK);
    }
    @PutMapping("/Admin/{msg}")//close and open the polling in
web clint from adim page
      public @ResponseBody ResponseEntity<String>
VoteCrl(@PathVariable String msg)

    {
        System.out.println(msg);
       if(msg.equals("open")){
           ctr ="open";
        return new ResponseEntity<>("Polling opend",
HttpStatus.OK); //returen ok =200
       }else
           ctr ="close";
           return new ResponseEntity<>("Polling Closed",
HttpStatus.NOT_FOUND);
    }

    //-------------------------------------------------------
---@GetMapping
    @PostMapping("/polling") //set mapping for vote - Create -
    public @ResponseBody ResponseEntity<String>
AddVote(@RequestBody Voters vo)
    { //add ip and bird name
         //add vote recored to database (Hashmap)
        if(ctr.equals("open"))
        {
         String result = ps.addVote(vo);
         if(result.equals("Vote Recorded"))
         {
         return new ResponseEntity<>(
                 ps.addVote(vo), HttpStatus.CREATED
         );
         }//If User Add Member Http Respond 201 = Created Vote
For Member
         else
         return new ResponseEntity<>(
                 ps.addVote(vo), HttpStatus.NOT_FOUND
         );
        }
        else return new ResponseEntity<>("Pool Closed",
HttpStatus.NOT_FOUND);
    }//If There Is No User, Or Wrong Member Http Respond 404 =
Not Found
    //-------------------------------------------------------
-----@PostMapping
       @GetMapping("polling/{id}") //set vote mapping, Show
       public @ResponseBody  ResponseEntity<Birds>
getVote(@PathVariable String id)
       { //Add ID numbers
        Voters vote;
           try{
```

```
                vote = ps.getVote(id); //call id from the get
method from welcome server
                Birds birdss ; //get bird name from vote
                String cc =vote.getName();
          if(cc != null)
          {
            birdss= ps.getbirds(vote.getName());
            return new ResponseEntity<>(birdss,
HttpStatus.FOUND);}
           }
           catch(Exception e)
           {}
         return new ResponseEntity<>(null,
HttpStatus.NOT_FOUND);
       }//End
    //-----------------------------------------------------
---------@DeleteMapping
    @DeleteMapping("polling/{id}") //Mapping to remove the
vote
    public @ResponseBody ResponseEntity<String>
RemoveVote(@PathVariable String id)
    { //add ID number in remove vote method
        ps.removeVote(id); // remove vote from voterdb
        return new ResponseEntity<>("vote Removed",
HttpStatus.OK); //returen ok =200
    }

    //-----------------------------------------------------
----------------------

    @GetMapping("info/{id}") //set vote mapping, Show
       public @ResponseBody  ResponseEntity<String>
getVoterinfo(@PathVariable String id) throws IOException
       {
           BirdsServiceImpl bsi =new BirdsServiceImpl();

        try {
            return new
ResponseEntity<>(bsi.ckeckvoterinfo(id).toString(),
HttpStatus.OK);
        } catch (MalformedURLException ex)
        {

Logger.getLogger(BirdController.class.getName()).log(Level.SEV
ERE, null, ex);
        } catch (JSONException ex)
        {

Logger.getLogger(BirdController.class.getName()).log(Level.SEV
ERE, null, ex);
        }
        return new ResponseEntity<>("Voter not found",
HttpStatus.OK);
       }
}//End Class
```

Birds.java

```
package pool.model; //Package Name

public class Birds
{ //Start Birds Class Name

    private String BirdName;
    private String BirdSname;
    private String BirdDes;
    //--------------------Inisilizing The Veribles

    public Birds()
    {  //Start Default Birds Constructure
        BirdSname = "_";
        BirdName = "_";
        BirdDes = "_";
        //------------Setting Attributes
    }//End Default Birds Constructure

    // Standard constructor
    public Birds(String birdname, String birdsname ,String
birddes)
    {//Start Standard constructor
        this.BirdName = birdname;
        this.BirdSname = birdsname;
        this.BirdDes = birddes;
        //--------------------Invokes Paramerter
    }//End Standard constructor

    // Copy constructor
    public Birds(Birds that)
    {//Start Copy constructor
        this.BirdName = that.BirdName;
        this.BirdSname = that.BirdSname;
        this.BirdDes = that.BirdDes;
    }//End Copy constructor


    //-----------------------{Set Getter And Setter}---------
----------------\\
     public String getscientificName()//Stert Get Scientific
Name
     {
         return BirdSname;
     }//End Get Scientific Name
    public void setscientificName(String birdsname) //Start
Set Scientific Name
    {
        this.BirdSname = birdsname;
    }//End Set Scientific Name
    //----------------------------Gettter Seeter Scientific
Name
```

```
    public String getcommonName() //Stert Get Common Name
    {
        return BirdName;
    }//End Get Common Name
    public void setcommonName(String birdname) //Stert Set
Common Name
    {
        this.BirdName = birdname;
    }//End Set Common Name
    //-----------------------------Gettter Seeter Common Name

     public String getdescription()
     {//Stert Get Description
         return BirdDes;
     }//End Get Description
    public void setdescription(String birddes)
    {//Stert Set Description
        this.BirdDes = birddes;
    }//End Set Description
    //-----------------------------Gettter Seeter Description

    @Override
    public String toString()//Method To returen Bird Scine
Name,Discription with Comman Name.
    {//Start ToString Method
        return "Welcome{" +
                "scientificName=\"" + BirdSname + "\"," +
                "commonName=\"" + BirdName + "\"," +
                "description=\"" + BirdDes + "\"}";
    }//End ToString Method

}//End Birds Class
```

VoteCount.java

```
package pool.model;
public class VoteCount {

    private String Name;
    private int Count;
    //----------------Declare veribles

    //construcuter
    public VoteCount(String name,int count)
    {

    this.Name=name;
    this.Count=count;

    }

    //------------------------(Getter and Setter)
    public String getName() {
        return Name;
    }

    public void setName(String Name) {
        this.Name = Name;
    }

    public int getCount() {
        return Count;
    }

    public void setCount(int Count) {
        this.Count = Count;
    }

}
```

Voters.java

package pool.model;

```
public class Voters {
    private String ID;
    private String NAME;
    //----------------Create veribles
    public Voters() {  NAME = "_"; ID = "_" ;};

    // Standard constructor
    public Voters(String id,String name) {
        this.ID = id;
        this.NAME = name;
    }

    // Copy constructor
    public Voters(Voters that) {
        this.ID = that.ID;
        this.NAME = that.NAME;
    }
     public String getName() {

         if(NAME==null){
         return null; }
         return NAME;
     }
    public void setName(String name) { this.NAME = name; }
    public String getId() { return ID; }
    public void setId(String id) { this.ID = id; }

    @Override
    public String toString() {
        return "{" +
                "\"Name\":\"" + "\""+NAME+ "\"," + "\"ID\":\""
+ "\""+ID + "\"}";
    }

}
```

BirdPoService.java

```
// An interface to the business logic, living in the service sub-package.

package pool.service;

import pool.model.VoteCount;
import pool.model.Birds;
import pool.model.Voters;
import java.util.List;


public interface BirdPoService
{
   // Adds a Candidates to the database.
   void addBird(Birds welcome);

   // Adds a candidates birds to the database
   String addVote(Voters vote);


   // Returns true if there is a birds or not
   default boolean hasBirds(String brd)
   {
      return getbirds(brd) != null;
   }

   // Returns a birds
   default Birds getbirds(String brd)
   {
      return getBirds(brd, null);
   }

   // Returns a birds
   default Voters getVote(String brd)
   {
      return getVote(brd);
   }

   Birds getBirds(String brd, String name);

   // Returns a list of all candidates in the database.
   List<Birds> getAllBirds();
    List<VoteCount> getAllVote();

   // Removes the birds
   void removeVote(String lang);

   //show candidates
   void Tally();
```

}

BirdsServiceImpl.java

```
package pool.service;
//---------------------Package Name

import pool.model.VoteCount;
import pool.model.Birds;
import pool.model.Voters;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Map;
import java.util.HashMap;
import java.util.List;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONException;
import org.json.JSONObject;
import org.springframework.stereotype.Component;
//----------------------Import Classes

@Component
public class BirdsServiceImpl implements BirdPoService
{
    //Create HashMap For Two Models
    private Map<String, Birds> Birddb;
    private Map<String, Voters> VoteDb; //----

    public BirdsServiceImpl()
    {//Start Constructures
        Birddb = new HashMap<>();
        VoteDb = new HashMap<>();//HashMap to Store Vote
Record
    }//End Constructres

    // Adds a birds to the database.
    public void addBird(Birds birds)
    {//Start AddBird Method
        if (birds != null ) {

            birds = new Birds(birds);
            Birddb.put(birds.getcommonName(), birds);
        }
    }//End AddBird Method

    public Birds getBirds(String lang, String name) {
        Birds birds = Birddb.get(lang);
        if (birds == null) {
            return null;
```

```
        }
        birds = new Birds(birds);
        if (name != null) {

        }
        return birds;
    }

    // Returns a list of all candidates in the database.
    public List<Birds> getAllBirds() {
        ArrayList<Birds> list = new ArrayList<>();
        // copying each welcome to pwelcomerotect objects in
the database from changes
        Birddb.values().forEach(cand -> {
            list.add(new Birds(cand));
        });
        return list;
    }

    // React Vote method
    public void removeVote(String ID) {
        VoteDb.remove(ID);
    }

   public String addVote(Voters vote) {
      String msg ="Voter Not in Record" ;
        try {
            String id = vote.getId();
            String res = ckeckvoter(id);


            if (vote != null &&  res.equals("found")) {
                // copying welcome to isolate objects in the
database from changes
                vote = new Voters(vote);
                VoteDb.put(vote.getId(),vote);
               msg ="Vote Recorded";
            }
        } catch (IOException ex) {

Logger.getLogger(BirdsServiceImpl.class.getName()).log(Level.S
EVERE, null, ex);
        } catch (JSONException ex) {

Logger.getLogger(BirdsServiceImpl.class.getName()).log(Level.S
EVERE, null, ex);
        }
        return msg;

   }
     public Voters getVote(String id) {
        Voters vote = VoteDb.get(id);
        if (vote == null) {
            return null;
        }

        if (id != null) {
           vote = new Voters(vote);
```

```
        }
        return vote;
    }

    public String  ckeckvoter(String id) throws
MalformedURLException, IOException, JSONException{
        // conecct to Voter web site and get information
        String url
="https://pmaier.eu.pythonanywhere.com/sawb/member/"+id;
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)
obj.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", "Mozilla/5.0");
        int responseCode = con.getResponseCode();
        BufferedReader in = new BufferedReader(
                new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();
        // ------------------------------------------------
        JSONObject obj2 = new JSONObject(response.toString());
// Read JSON file Recvied from website
        String oname =
obj2.getJSONObject("member").getString("name"); // Read Name
from JSON file
        String number =
obj2.getJSONObject("member").getString("number"); // Read ID
Number From JSON

        JSONObject json = new JSONObject();  // Creat JSON
container
        json.put(number, oname);   // ADD number and Name to
the json
        String message = json.toString(); //conver json to
String
         System.out.print(oname+number+id);
        if( id.equals(number)){

        return "found";
        }else

        {return "not found";}}

//=============================================================
=========

        public JSONObject  ckeckvoterinfo(String id) throws
MalformedURLException, IOException, JSONException{
        // conecct to Voter web site and get information
        String url
="https://pmaier.eu.pythonanywhere.com/sawb/member/"+id;
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)
obj.openConnection();
```

```
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", "Mozilla/5.0");
        int responseCode = con.getResponseCode();
        BufferedReader in = new BufferedReader(
                new InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();
        // ------------------------------------------------
        JSONObject obj2 = new JSONObject(response.toString());
// Read JSON file Recvied from website
        String oname =
obj2.getJSONObject("member").getString("name"); // Read Name
from JSON file
        String number =
obj2.getJSONObject("member").getString("number"); // Read ID
Number From JSON
        String age =
obj2.getJSONObject("member").getString("age"); // Read ID
Number From JSON
        String region =
obj2.getJSONObject("member").getString("region"); // Read ID
Number From JSON

        JSONObject json = new JSONObject();  // Creat JSON
container
        json.put("name", oname);   // ADD number and Name to
the json
        json.put("number", number);   // ADD number and Name
to the json
        json.put("age", age);   // ADD number and Name to the
json
        json.put("region", region);   // ADD number and Name
to the json
        String message = json.toString(); //conver json to
String
        return json;
}

    @Override
    public void Tally() {
    }


    @SuppressWarnings("empty-statement")
    public ArrayList<VoteCount> getAllVote() {
        ArrayList<Voters> list = new ArrayList<>();

        VoteDb.values().forEach(vote -> { list.add(new
Voters(vote)); }); // Read Hashmap value and put it in #list
              Integer count1 =0;
              Integer count2 =0;
              Integer count3 =0;
              //Declare varibles for count in admin
```

```
              // for loop to count vote from list
              for (int i = 0; i < list.size(); i++) {

if("FlamingoCove".equals(list.get(i).getName())){  // read
list and add 1 to int count1 if b1 is found
                count1++;
                };
                 if("Parrots".equals(list.get(i).getName())){
// read list and add 1 to int count2 if b1 is found
                count2++;
                };
                if("Peafowl".equals(list.get(i).getName())){
// read list and add 1 to int count3 if b1 is found
                count3++;
                };
    }
          //-----------------------------------------
              //VoteCount: is a data model
               ArrayList<VoteCount> vc = new ArrayList<>();
// array list to save counting result
               vc.add(new VoteCount("FlamingoCove",count1));
// add result for b1 with numbers
               vc.add(new VoteCount("Parrots",count2)); // add
result for b1 with numbers
               vc.add(new VoteCount("Peafowl",count3)); // add
result for b1 with numbers

        return vc ;  // Send the vc arraylist to be displayed
when tally button in clicked in admin
    }}
```