CSCU9YW – Web Services

Assignment Report

Student Number: 3058058

Table of Contents

1. Problem Overview
1.1 Task Outline

The objective of this assignment is to Build and implement a Web-Based Polling Application. This Application is made by using JAVA as a Spring Boot Application, the design principle used is the REST Design Principle which will contain clients interacting with the application or web service.

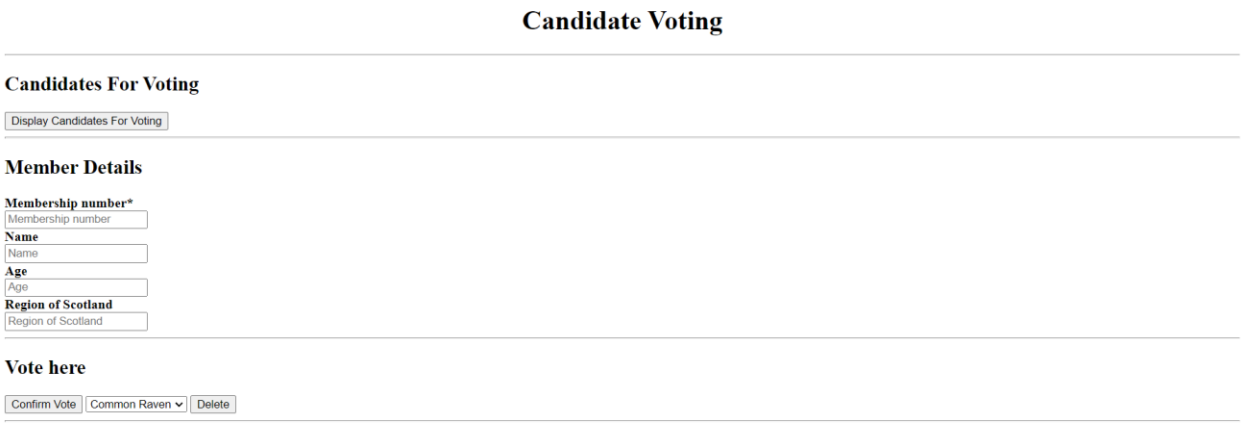
The Scottish Association for the Watching of Birds (SAWB) conducts an annual poll for members to vote on the bird of the year. The task is to implement a web service for the annual polling. The main function will allow the members of SAWB to vote for the bird they have selected Bird of the Year candidate. The Votes are not anonymous because member's data is recorded, but other members unable to find out anything about other individual votes.

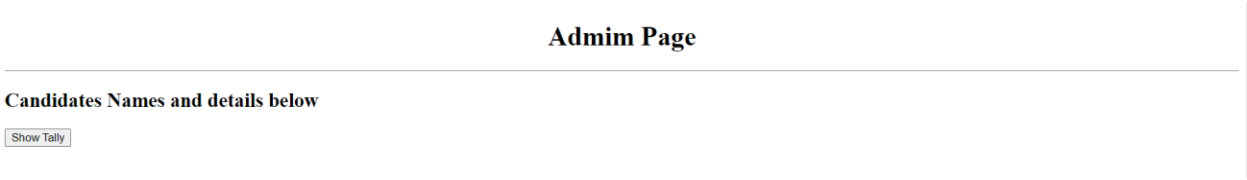
Administrators have extra functions in the polling system where they can open and close the polling service and tally the votes of each candidate. However, administrators can't check individual votes.

The Information and details about each candidate for the Bird of the Year have their common name, scientific name and the description of each bird. The details are recorded for SAWB members will only have their membership number, name, age and region in Scotland of where they lived. The voting system provide at least three choices of birds to choose from.

Each member can make only one vote, but this vote can be deleted if the member decides to change their choice. There can be multiple votes for each candidate but only the final count of the votes matter.

2. Implementation
2.1 Output

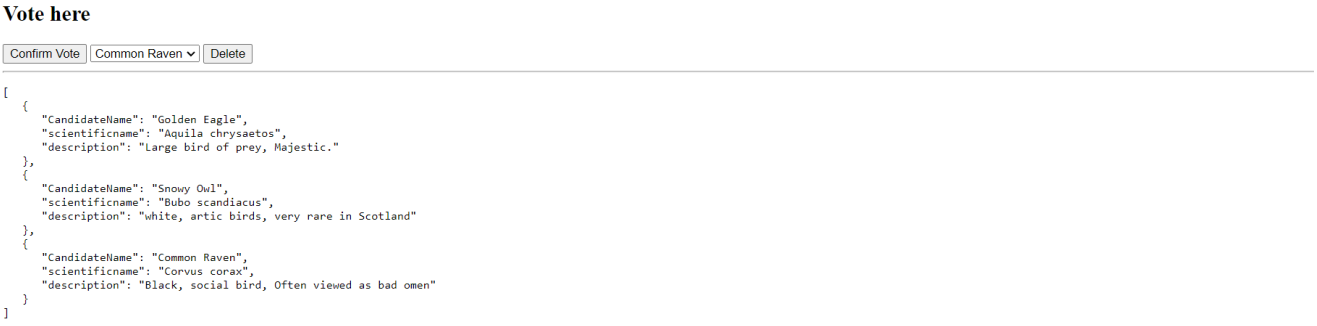




The above images show that the users (admin or client) can interact with the page. The Candidate Voting is for the members/clients and the Admin page is for the administrator.

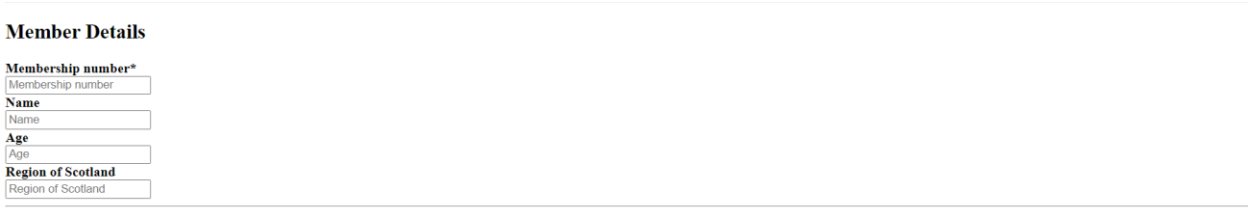
2.1.1 Display Candidate Name and Details

At the top of the Candidate Voting page, the client has the ability to display the candidates by clicking the “Display Candidates For Voting” Button, which will show all the candidates for the poll.

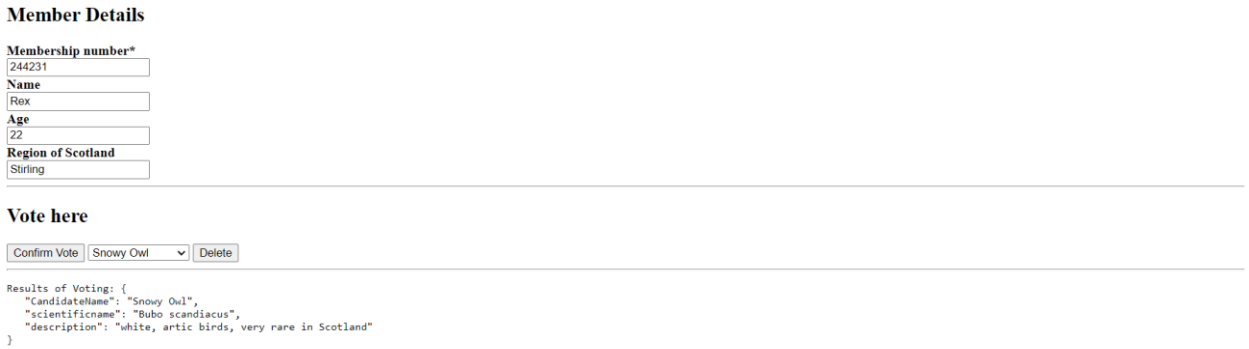


There are at least three candidates available for voting, the Golden Eagle, Snowy Owl, and the Common Raven. Along with the common names for each candidate are the scientific name and the description of each candidate as shown above.

2.1.2 Making a Vote



Every member must fill out basic details of themselves before submitting their final vote. The member must enter their Membership ID, name, age, and the region of Scotland where they live. After that they can choose their choice for bird of the year by first choosing their choice and the clicking the “Confirm Vote” button, as shown above.



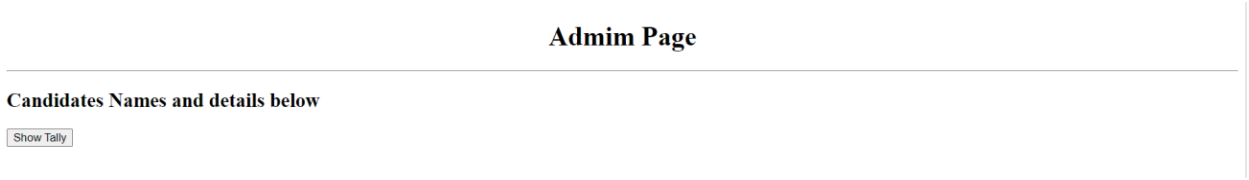
Once the vote is successful, the voting result will be shown at the bottom of the page showing the candidate they have chosen. The successful vote will be tallied in the Admin page. (Please note the Common Raven was tallied as this is a different vote than the one shown above.)

Admim Page

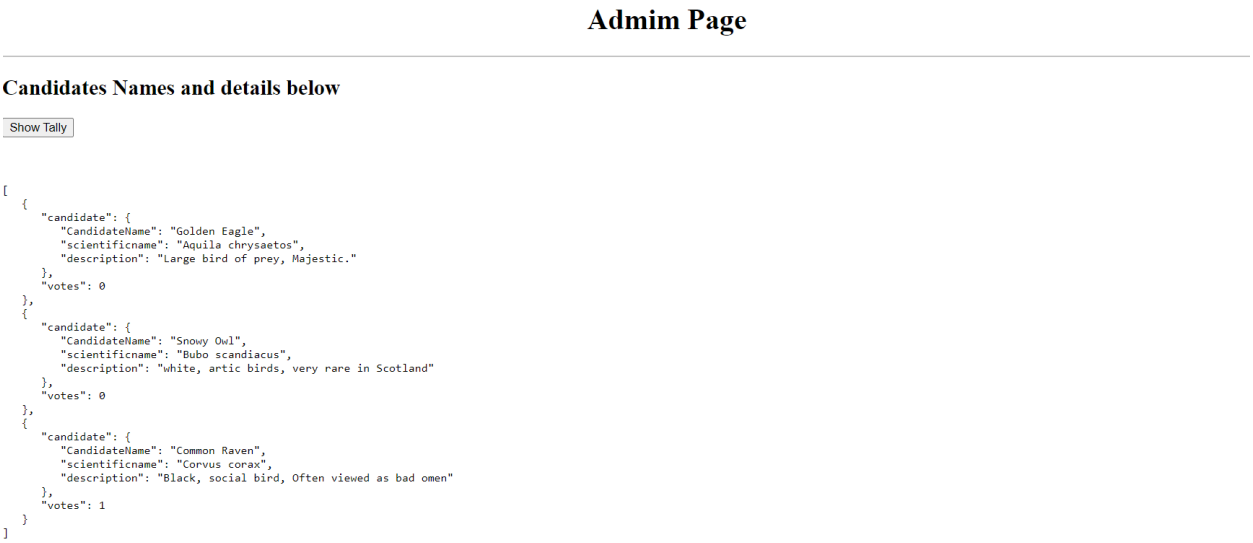
2.1.3 Deleting a Vote

If the Client wants to change their mind and choose a different candidate from the one, they have chosen they have to functionality to delete their vote and choose again. This can be done by clicking the “Delete” button which will erase their vote and redo the entire voting process again. The vote is also subtract from the Admin Page.

2.1.4 Display of Votes



The Administrator for SAWB can display and see the total tally for each candidate, an ability that clients/members do not have. The administrator can see the tally for each candidate after clicking the “Show Tally” button, which will display each candidate, details of each candidate and the total tally they have achieved.



The image above shows the results of the polls and the total votes for each candidate.

3. Appendix
3.1 WebServicesApplication.java

```
package WebServicesAssignment;


import java.util.HashMap;
import java.util.Map;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import WebServicesAssignment.Model.*;
import WebServicesAssignment.Services.*;
import org.springframework.context.annotation.Bean;
@SpringBootApplication
public class WebServicesAssignmentApplication {

	public static void main(String[] args) {
		SpringApplication.run(WebServicesAssignmentApplication.class, args);
	}
	@Bean
	public CommandLineRunner initDB(CandidateService candidateServiesO, VoteService
ballotServicesO)
	{
		return (args) -> {
			// Name and details of candidate birds are given here
			candidateServiesO.addCandidate(new Candidate("Snowy Owl", "Bubo
scandiacus","white, artic birds, very rare in Scotland"));
			candidateServiesO.addCandidate(new Candidate("Common Raven",
"Corvus corax","Black, social bird, Often viewed as bad omen"));
			candidateServiesO.addCandidate(new Candidate("Golden Eagle", "Aquila
chrysaetos","Large bird of prey, Majestic."));
	for (int i=0; i<candidateServiesO.getCandidate().size(); i++) {
		ballotServicesO.addVote(new Vote(candidateServiesO.getCandidate().get(i)));
		}
		};


	}
	}
```

3.2 VoteController.java

```
package WebServicesAssignment.Controller;



import static org.springframework.hateoas.server.mvc.WebMvcLinkBuilder.*;
import org.springframework.context.annotation.Bean;
import java.util.List;
import java.util.stream.Collectors;
import java.lang.reflect.Member;
import java.net.URI;
import java.net.URISyntaxException;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.server.ResponseStatusException;
import WebServicesAssignment.Model.*;
import WebServicesAssignment.Services.*;
import org.springframework.hateoas.CollectionModel;
import org.springframework.hateoas.EntityModel;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

@RestController
@CrossOrigin
public class VoteController {
	private VoteService ballotServicesO;
	private CandidateService candidateServiesO;

	public VoteController(CandidateService candidateServiesO, VoteService
ballotServicesO) {
		this.candidateServiesO = candidateServiesO;
		this.ballotServicesO = ballotServicesO;
	}

	@GetMapping("/candidateBN/{candidateNameBird}")
	//Here the program are linking to the Admin html page.
	public Candidate getCandidate(@PathVariable String candidateNameBird) {
		Candidate candidateBird = candidateServiesO.getCandidate(candidateNameBird);
```

```
		if (candidateBird == null) {
			throw new ResponseStatusException(HttpStatus.NOT_FOUND);
		}
		return candidateBird;
	}

	@GetMapping("/candidateBN")
	//List of candidate is sent for display
	public List<Candidate> getAllCandidates() {

		List<Candidate> list = candidateServiesO.getCandidate();
		return list;
	}

	@GetMapping("/Votes/{candidateNameBird}")
	//here the program is linking to the voter html page.
	public EntityModel<Vote> getVote(@PathVariable String candidateNameBird) {
		Vote ballot = ballotServicesO.getVote(candidateNameBird);
		if (ballot == null) {
			throw new ResponseStatusException(HttpStatus.NOT_FOUND);
		}
		return EntityModel.of(ballot,
linkTo(methodOn(VoteController.class).getVote(candidateNameBird)).withSelfRel(),

	linkTo(methodOn(VoteController.class).getAllVotes()).withRel("Ballots"));
	}

	@GetMapping("/votes")
	//Here gets the vote information
	public List<Vote> getAllVotes() {

		List<Vote> list = ballotServicesO.getAllVotes();
		return list;
	}

	@GetMapping("/votes/VoteAdd")
	//Vote is added to the database
	public ResponseEntity<List> countVote() {
		ballotServicesO.countVote();
		return new ResponseEntity<List>(ballotServicesO.getAllVotes(),
HttpStatus.OK);
	}

	@PutMapping("/votes/VotePost/{candidateNameBird}")
	//Vote is checked to be in the database
	public ResponseEntity<Candidate> postVote(@PathVariable String candidateNameBird,
```

```
                @RequestBody WebServicesAssignment.Model.Member members) {

        Candidate candidateBird =
(ballotServicesO.Post(members.getMembershipNumber(), candidateNameBird));
        if (candidateBird == null) {
            throw new ResponseStatusException(HttpStatus.BAD_REQUEST);
        }

        return new ResponseEntity<Candidate>(candidateBird,null,201);


        }

    @DeleteMapping("/votes/VoteRemove/{candidateNameBird}")
    //Vote is deleted from the database
    public ResponseEntity<String> removeVote(@PathVariable String candidateNameBird,
            @RequestBody WebServicesAssignment.Model.Member members) {
        ballotServicesO.Remove(members.getMembershipNumber(),
candidateNameBird);
        return new ResponseEntity<String>(null, null, 201);
    }
}
```

3.3 Vote.java

```
package WebServicesAssignment.Model;

import java.util.ArrayList;


public class Vote {

	private ArrayList<String> membershipNumber;
	private int vote;
	private Candidate candidateBird;


	//Constructor
	public Vote() {
		candidateBird = new Candidate();
		vote = 0;
		membershipNumber = new ArrayList<String>();
	}
	public Vote(Candidate birdCandidate) {
		this.vote= 0;
		this.candidateBird = birdCandidate;
		this.membershipNumber= new ArrayList<String>();
	}
  //copy constructor
	public Vote(Vote memberVote) {
		this.vote = memberVote.vote;
		this.candidateBird= memberVote.candidateBird;
		this.membershipNumber=memberVote.membershipNumber;

	}
	//Adds member ID number
	public void addMemberNum(String id)
	{
		this.membershipNumber.add(id);
	}
	//Gets the votes
	public int getVotes() {
		return this.vote;
	}
	//Sets the votes
	public void setVotes(int votes ) {
		this.vote=votes;
	}
	//Gets the Candidate
	public  Candidate getCandidate() {
```

```
        return this.candidateBird;
}
//Sets the candidate
public void setCandidate(Candidate birdCandidate) {
        this.candidateBird = birdCandidate;
}
//Array returns Member ID Number
public  ArrayList<String> ReturnMembershipNum() {
        return this.membershipNumber;
}
//Sets MembershipId number
public void setMembershipID(ArrayList<String> membershipid) {
this.membershipNumber = membershipid;
}
@Override
public String toString() {
        return  "Candidate{" +
          "Membership ID=\"" + membershipNumber + "\"," +
          "Count of vote=\"" + vote + "\"," +
          "Candidate=\"" + candidateBird + "\"" +
         "}";
}

}
```

3.4 Candidate.java

```
package WebServicesAssignment.Model;

import com.fasterxml.jackson.annotation.JsonProperty;

public class Candidate {
	private String candidateNameBird;
	private String scientificName;
	private String descriptionOfCandidates;

	public Candidate() {
		candidateNameBird = "_";
		scientificName = "_";
		descriptionOfCandidates = "_";
	}

	/**
	 * This constructor will create a candidate for the vote.
	 *
	 * @param commonName     common name of the bird candidate.
	 * @param scientificName scientific name of the candidate bird is given.
	 * @param description    description of the candidate bird is given here.
	 */
	public Candidate(String candidateNameBird, String scientificName, String description) {
		this.candidateNameBird = candidateNameBird;
		this.scientificName = scientificName;
		this.descriptionOfCandidates = description;
	}

	// Copy constructor
	public Candidate(Candidate candidateBird) {
		this.candidateNameBird = candidateBird.candidateNameBird;
		this.scientificName = candidateBird.scientificName;
		this.descriptionOfCandidates = candidateBird.descriptionOfCandidates;
	}

	// Gets the Candidate Name
	@JsonProperty("CandidateName")
	public String getCandidateName() {
		return this.candidateNameBird;
	}

	// Sets the Candidate name
	@JsonProperty("CandidateName")
	public void setCandidateName(String candidateNameBird) {
		this.candidateNameBird = candidateNameBird;
```

```
	}

	// Gets the Scientific Name of the candidate
	@JsonProperty("scientificname")
	public String getScientificName() {
		return this.scientificName;
	}

	// Sets the Scientific Name of the candidate
	@JsonProperty("scientificname")
	public void setScientificName(String scientificname) {
		this.scientificName = scientificname;
	}

	// Gets the description of the candidate
	@JsonProperty("description")
	public String getDescription() {
		return this.descriptionOfCandidates;
	}

	// Sets the description of the candidate
	@JsonProperty("description")
	public void setDescription(String description) {
		this.descriptionOfCandidates = description;
	}

	@Override
	// returns string statement of description of candidate ,candidate name and
	// Scientific name
	public String toString() {
		return "Candidate{" + "Candidate Name=\"" + candidateNameBird + "\"," +
"Scientific Name=\"" + scientificName
				+ "\"," + "Description=\"" + descriptionOfCandidates + "\"" + "}";
	}
}
```

3.5 Member.java

```
package WebServicesAssignment.Model;

public class Member {
	private String memberName;
	private String membershipNumber;
	private int memberAge;
	private String regionOfScot;

	public Member() {
		memberName = "_";
		membershipNumber = "-";
		memberAge = 0;
		regionOfScot = "_";
	}

	/**
	 * This constructor is being used to create a member and all the necessary
	 * attributes.
	 *
	 * @param membershipNumber The membership identification number of the member
is
	 *                         given here.
	 * @param memberName       The name of the member is given here.
	 * @param memberAge        The age of the member is given here.
	 * @param regionOfScot     In which region of the Scotland does the member live.
	 */
	public Member(String memberName, String membershipNumber, int memberAge,
String regionOfScot) {
		this.memberName = memberName;
		this.membershipNumber = membershipNumber;
		this.memberAge = memberAge;
		this.regionOfScot = regionOfScot;
	}

	/**
	 * This constructor is being used to create a member and all the necessary
	 * attributes.
	 *
	 * @param memberName The name of the member is given here.
	 */
	public Member(Member members) {
		this.memberName = members.memberName;
		this.membershipNumber = members.membershipNumber;
		this.memberAge = members.memberAge;
```

```
        this.regionOfScot = members.regionOfScot;
}

// Gets the member's membershipNumber
public String getMembershipNumber() {
        return membershipNumber;
}

// Sets the member's membershipNumber
public void setMembershipNumber(String membershipNumber) {
        this.membershipNumber = membershipNumber;
}

// Gets the member name
public String getMembername() {
        return memberName;
}

// Sets the member name
public void setMembername(String memberName) {
        this.memberName = memberName;
}

// Gets the age of member
public int getAge() {
        return memberAge;
}

// Sets the age of member
public void setAge(int age) {
        this.memberAge = age;
}

// Gets the region of Scotland the member lives in.
public String getRegion() {
        return regionOfScot;
}

// Sets the region of Scotland the member lives in.
public void setRegion(String region) {
        this.regionOfScot = region;
}

@Override

public String toString() {
```

```
                return "Participating Members{" + "Member ID='" + membershipNumber + '\'' +
",Person name='" + memberName + '\''
                        + ", Person age=" + memberAge + ", Scotland Regeion='" +
regionOfScot + '\'' + '}';
        }

}
```

3.6 VoteService.java

```
package WebServicesAssignment.Services;

import java.util.List;
import WebServicesAssignment.Model.*;

public interface VoteService {
        // Adds a vote to the database, or overwrites the existing one.
   void addVote(Vote memberVote);


   //True will be returned if there is a vote
   default boolean hasVote(String candidateNameBird) {
      return getVote(candidateNameBird) != null;
   }

   // Returns a vote; null if there is none.
   Vote getVote(String candidateNameBird);

   Candidate Post(String membershipNumber, String candidateNameBird);
   void Remove(String membershipNumber, String candidateNameBird);



   // Returns a list of all votes in the database.
   List<Vote> getAllVotes();


        void countVote();

   // Removes the welcome for language lang from the database.


}
```

3.7 VoteServiceImpl.java

```
package WebServicesAssignment.Services;

import java.util.ArrayList;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Component;
import org.springframework.web.server.ResponseStatusException;

import WebServicesAssignment.Model.*;

@Component
public class VoteServiceImpl implements VoteService {

	Map<String, Vote> database;

	public VoteServiceImpl() {
		database = new HashMap<>();
	}

	// Here the ballot is added to the database or overwrite an existing one
	public void addVote(Vote memberVote) {
		if (memberVote != null && memberVote.getCandidate() != null) {
			memberVote = new Vote(memberVote);
			database.put(memberVote.getCandidate().getCandidateName(),
memberVote);
		}
	}

	public Vote getVote(String candidateNameBird) {
		Vote vote = database.get(candidateNameBird);
		return vote;
	}

	// Here the list of all votes/ballots returned to the database.
	public List<Vote> getAllVotes() {
		List<Vote> list = new ArrayList<Vote>();

		database.values().forEach(vote -> {
			list.add(new Vote(vote));
		});
```

```
		return list;
	}

	public Candidate Post(String membershipNumber, String candidateNameBird) {

		for (Vote vote: database.values()) {

			if (vote.ReturnMembershipNum().contains(membershipNumber)) {
				System.out.print(membershipNumber);

				return null;
			}
		}
		database.get(candidateNameBird).addMemberNum(membershipNumber);

	database.get(candidateNameBird).setVotes(database.get(candidateNameBird).getVotes()
+ 1);

		return database.get(candidateNameBird).getCandidate();
	}

	@Override
	// Here the votes for each Candidate name is counted
	public void countVote() {
		int numVote;
		List<Vote> list = new ArrayList<Vote>();
		for (Map.Entry<String, Vote> e : database.entrySet()) {
			list.add(e.getValue());
		}
		for (Vote b : list) {
			numVote = 0;
			for (int i = 0; i < b.ReturnMembershipNum().size(); i++) {
				numVote++;
			}
			b.setVotes(numVote);
			database.replace(b.getCandidate().getCandidateName(), b);
		}
	}

	// Here the vote for the candidate is removed
	public void Remove(String membershipNumber, String candidateNameBird) {


	database.get(candidateNameBird).ReturnMembershipNum().remove(membershipNumbe
r);
```

```
        database.get(candidateNameBird).setVotes(database.get(candidateNameBird).getVotes()
- 1);

        }

}
```

3.8 CandidateService.java

```
package WebServicesAssignment.Services;

import java.util.List;

import WebServicesAssignment.Model.*;


public interface CandidateService {
        // Adds a candidate to our database
   void addCandidate(Candidate candidateBird);

   // Gets a candidate from our database
   Candidate getCandidate(String candidateNameBird);

   // Gets a list of candidates
   List<Candidate> getCandidate();


}
```

3.9 CandidateServiceImpl.java

```
package WebServicesAssignment.Services;

import java.util.ArrayList;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.stereotype.Component;

import WebServicesAssignment.Model.*;
import WebServicesAssignment.Services.*;

@Component
public class CandidateServiceImpl implements CandidateService {

        Map<String, Candidate> canDataBase;

        public CandidateServiceImpl() {
                canDataBase = new HashMap<>();
```

```
	}

	// Adds candidate to the database
	public void addCandidate(Candidate candidateBird) {
		if (candidateBird != null && candidateBird.getScientificName() != null) {

			candidateBird = new Candidate(candidateBird);
			canDataBase.put(candidateBird.getCandidateName(), candidateBird);
		}
	}

	// Gets a candidate's common name from our database
	public Candidate getCandidate(String candidateNameBird) {
		Candidate candidateBird = canDataBase.get(candidateNameBird);

		candidateBird = new Candidate(candidateBird);

		return candidateBird;
	}

	// Gets a list of all candidates
	public List<Candidate> getCandidate() {
		ArrayList<Candidate> list = new ArrayList<>();
		canDataBase.values().forEach(candidateBird -> {
			list.add(new Candidate(candidateBird));
		});
		return list;
	}

}
```