

ONLINE VOTING SYSTEM

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

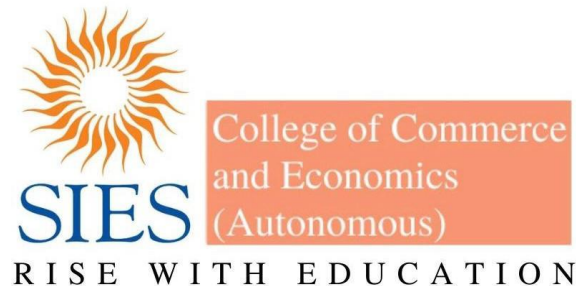
By

MR. JAMES NADAR

Seat no. 20

**Under the esteemed guidance of
Ms. RHUGWEDA JAGE**

**Coordinator Of BSC.IT
Mrs. BHAVINI SHAH**



**DEPARTMENT OF INFORMATION TECHNOLOGY
S.I.E.S COLLEGE OF COMMERCE AND ECONOMICS**

(AUTONOMOUS)

MUMBAI, 400022

MAHARASHTRA

2022-23

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No:

Roll No: _____

1. Name of the Student

2. Title of the Project

3. Name of the Guide

4. Teaching experience of the Guide _____

5. Is this your first submission?

Yes ☐

No ☐

Signature of the Student

Signature of the Guide

Date:

Date:

Signature of the Coordinator

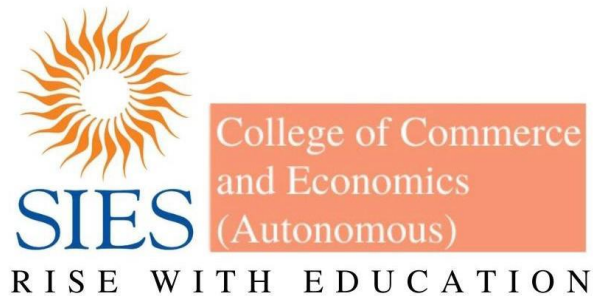
Date:

S.I.E.S COLLEGE OF COMMERCE AND ECONOMICS

(AUTONOMOUS)

MUMBAI-MAHARASHTRA-400022

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project entitled, "**ONLINE VOTING SYSTEM** ", is bonafied work of **MR. JAMES NADAR** bearing Seat. No: **20** submitted in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE** in **INFORMATIONTECHNOLOGY** from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

ABSTRACT

The Online Voting System is a website. The system has a centralized database to keep records of all the voters and candidates and final results. This website is time saving, workload reduced information available at time and it provide security for the data. The admin will maintain this website. This is a simple, safe and secure method that takes minimum of time.

The word 'vote' means to choose from a list, to elect or to determine. The main goal of voting is to come up with leaders of the people's choice. During elections, some of the problem faced by commission involved include ridging votes during election, insecure or inaccessible polling stations, inadequate polling materials and also inexperienced personnel. This system will provide more transparency to the voters and as well as the candidate also.

ACKNOWLEDGEMENT

We thank the people who were a part of this project in numerous ways, people who gave their unending support right from the stage the project idea was conceived.

The four things that go on to make a successful endeavor are dedication, hard work, patience and correct guidance.

We would like to thank our principal **DR. NINA ROY CHOUDHURY** who has always been the source of inspiration.

We are also thankful to **Mrs. BHAVINI SHAH** our coordinator for all the help she has rendered to ensure the successful completion of the project.

We take this opportunity to offer sincere thanks to **Ms. RHUGWEDA JAGE** who was very much kind enough to give us an idea and guide us throughout our project work.

We thank all teaching staff (I.T) who shared their experience and gave their suggestion for developing our project in a better way.

DECLARATION

I hereby declare that the project entitled, “**Online Voting System**” done at **SIES COLLEGE OF COMMERCE AND ECONOMICS (AUTONOMOUS)**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENTS

Chapter 1: Introduction	11
1.1. Background.....	11
1.2. Objectives	13
1.3. Purpose, Scope, and Applicability	14
1.3.1 Purpose	14
1.3.2 Scope	14
1.3.3 Applicability.....	15
1.4 Achievements.....	15
Chapter 2: Survey of Technologies.....	16
Chapter 3: Requirement and Analysis	18
3.1 Problem Definition	18
3.2 Requirements Specification	18
3.3 Planning and Scheduling.....	19
3.3.1 Planning.....	19
3.3.2 Scheduling	19
3.4 Proposed System.....	21
3.5 Existing System.....	22
3.5.1 Ballot Voting	22
3.5.2 EVM Voting	22
3.6 Software and Hardware Requirements	23
3.6.1 Software Requirements.....	23
3.6.2 Hardware Requirements	23
3.7 Conceptual Models	24
3.7.1 ER Diagram	24
3.7.2 Data Flow Diagram	25
3.7.3 UML Diagram.....	28
3.7.3.1 Class Diagram.....	28
3.7.3.2 Activity Diagram	29
3.7.3.3 Use case Diagram	30
3.7.3.4 Sequence Diagram	31

Chapter 4: System Design.....	32
4.1 Basic Modules.....	32
4.2 Data Dictionary.....	33
4.3 User Interface Design	35
4.4 Security Issues.....	39
Chapter 5: Implementation and Testing	40
5.1 Implementation Approaches.....	40
5.2 Testing Approach	101
5.2.1 Unit Testing	101
5.2.2 Integrated Testing	101
5.2.3 Beta Testing.....	102
5.2.4 System Testing.....	102
Chapter 6: Result and Discussion.....	109
6.1 Test Reports.....	109
6.2 User Documentation	110
Chapter 7: Conclusion	120
7.1 Significance of the System	120
7.2 Limitation of the System	121
7.3 Future Scope of the Project	121
References	122

List of Tables

COCOMO.....	100
Login Test Case.....	103
Registration Test Case.....	104
Admin Test Case.....	106
Add Candidate, Add Election Test cases.....	107

List of Figures

3.7.1	ER Diagram	24
3.7.2	Data Flow Diagram	25
3.7.3.1	Class Diagram	28
3.7.3.2	Activity Diagram	29
3.7.3.3	Use case Diagram	30
3.7.3.4	Sequence Diagram	31
4.3.1	Home Page	35
4.3.2	User Login Page.....	35
4.3.3	Registration Page	36
4.3.4	Admin Login Page	37
4.3.5	Admin UI Page	37
4.3.6	Add Candidate Page	38
4.3.7	Voting Page & View Result	38

Chapter 1

Introduction

1.1 Background

The Online voting system (OVS), also known as e-voting is a term encompassing several different types of voting embracing both electronic means of counting votes. Electronic voting technology can include punched cards, optical scan voting systems, and specialized voting kiosks (including self-contained direct recording electronic voting systems or DRE). It can also involve the transmission of ballots and votes via telephones, private computer networks, or the internet.

Online voting is an electronic way of choosing leaders via a web-driven application. The advantage of online voting over the common "queue method" is that the voters have the choice of voting in their own free time and there is reduced congestion. It also minimizes errors in vote counting. The individual votes are submitted in a database which can be queried to find out who of the aspirants for a given post has the highest number of votes.

With the "ONLINE VOTING SYSTEM", a voter can use her voting right online without any difficulty. She has to register as a voter first before being authorized to vote. The registration should be done before the voting date to enable data updates in the database.

However, not just anybody can vote. For one to participate in the elections, he/she must have the requirements. For instance, he/she must be a registered citizen i.e., must be 18 and above years old. As already stated, the project 'Online Voting' provides means for fast and convenient voting, and access to this system is limited only to registered voters.

Internet voting systems are appealing for several reasons which include; People are getting more used to working with computers to do all sorts of things, namely sensitive operations such as shopping and home banking and they allow people to vote far from where they usually live, helping to reduce absenteeism rate.

Election plays an important role in such a huge democratic country like India where the leader is elected by residents. Elections preserve a truthful state functioning, as they provide people the choice to select their personal government. So, the election ought to be an unfastened and truthful process. Every citizen of a democratic country has a right to vote with his/her own choice. One of the fundamental issues in the conventional democratic framework is that it expends bunches of labor and resources. Also, some humans can be worried about illegal publications of movement at some point of this manner of election or its preparation.

There are some disadvantages of the conventional election voting process which is being used in our country such as machine stops working, chances of brutality, time consuming, resource consuming, spot arranged etc. Many people couldn't vote because the voter has to reach the poll booths to vote or some people like those who are living far away from their original birthplace where they are allowed to vote. So, to get rid of their drawbacks, a new System is introduced i.e., Online Voting System, which provides accuracy, security, flexibility, mobility etc.

An online voting System in a web-based application to use in the election process. Initially ballot paper technique was used in the election process. Then the Electronic Voting Machine comes, these are easy to store the data and easily manageable. These are more secure than the ballot paper and less time consuming. Now, we proposed a system which is more authentication to make the voting process more secure and reduce the time taken in the voting process. By the use of this, the electorate can solidify their vote for his or her preferred candidate through the use of their system. We use more security system for authentication of citizens to see that he/she is the proper user or not.

We provide many modules in which admin can login with inside the tool and show the numerous operations. Also, users can login in the system and use their right to vote. When the Voter uses the system, the system will ask login Id, password and try to match with the original password, which was given by them in registration process, which is stored in the database. If both Id and passwords are the same, then the voter can cast his/her vote. Most higher learning establishments in some countries conduct elections routinely to elect an understudy leadership to choose them. They proposed, the process of an online system, which includes systems like enlistment of voters, vote casting, vote checking, and pronouncing results which would establish a decent answer for a substitute of framework that is in the institutes in many countries.

1.2 Objectives

The specific objectives of the project include Reviewing the existing/current voting process and implementing an online voting system. Validating the system to ensure that only eligible voters are allowed to vote.

The aim is to develop an application that seeks to use various stages of security authentication to enhance the election process for political party elections using
The real case study, i.e. The University of Mumbai, in the end imparting an internet platform which permits all eligible electorate to work out their franchise from any region for the duration of the election period.

The targets are:

- To create a secure online voting platform where authenticity of votes and voters are ensured with the use of login Id and password.
- To enhance Voter's password to identity authentication process of users.
- To ease the trouble of queuing while balloting duration in elections.
- Validating the system to ensure that only legible voters are allowed to vote.
- Implementing an online protected voting system.

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose:

The main purposes of the Online Voting System include:

- Provision of improved voting services to the voters through fast, timely and convenient voting.
- Reduction of the costs incurred by the Kenyan Electoral Commission during the voting time in paying the very many clerks employed for the sake of the success of the manual system.
- Check to ensure that the members who are registered are the only ones to vote.
- Cases of "DeadPeople" voting are also minimized.
- An online voting system (OVS) will require being very precise or cost-cutting to produce an effective election management system.

Therefore, crucial points that this (OVS) emphasizes are listed below.

- i. Require a smaller number of staff during the election.
- ii. This system is a lot easier to independently moderate the elections and subsequently reinforce its transparency and fairness.
- iii. Less capital, less effort, and less labour intensive, as the primary cost and effort will focus primarily on creating, managing, and running a secure online portal.
- iv. Increased number of voters as individuals will find it easier and more convenient to vote, especially those abroad.

1.3.2 Scope:

It makes sure that the people's votes are counted. This system is a lot easier to independently moderate the elections and subsequently reinforce its transparency and fairness.

This is also will produce less effort and less labor intensive, as the primary cost focuses primarily on creating, managing, and running a secure web voting portal. An increasing number of voters as individuals will find it easier and more convenient to vote, especially those abroad. It is limited to registered voters only.

1.3.3 Applicability:

Online Voting System can be used in the internal elections such as

- o Used in National Elections.
- o Used in Television shows.
- o Used in taking mass opinions.
- o Society Elections
- o College Elections

This type of online elections gives feasibility to the user and the candidate and the main important thing is transparency. So, by these types of advantages Online Voting System is very useful and easy to use.

1.4 Achievements

The ONLINE VOTING SYSTEM shall reduce the time spend making long queues at the polling stations during voting. It shall also enable the voters to vote from any part of the globe as explained since this is an online application available on the internet. Cases of vote miscounts shall also be solved since at the backend of this system resides a well-developed database using SQL Server that can provide the correct data once its correctly queried. Since the voting process shall be open as early as possible, the voters shall have ample t time to decide when and whom to vote.

Chapter 2

Survey of Technologies

For Online Voting System, we have used Asp.Net razor page for creating web pages. CSHTML describes the structure of a web page. It is very easy to learn and understand. A CSHTML file is a C# HTML webpage file used by Razor, an ASP.NET view engine that generates webpages. It is like a standard ASP.NET webpage (.ASP or .ASPX file) but uses slightly different syntax. CSHTML files run on a web server, which generates HTML for a client web browser. CSHTML is supported by all browsers. CSHTML is the friendliest search engine, compared to other front-end languages it is simple to edit and can integrate easily with other languages.

One easy way to recognize a CSHTML file is to look for @ symbols that precede Razor reserved directives and code blocks. CSHTML files use the @ symbol and Razor reserved directives to transition from HTML markup to C# code. Most CSHTML files are created in **Microsoft Visual Studio**. Visual Studio provides syntax highlighting and Intelligence code suggestions that help developers create CSHTML files.

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colors, layout, and fonts, thus making our web pages presentable to the users. It is independent of HTML and can be used with any XML-based markup language. There are many advantages of CSS, it saves time, and you can write CSS once and then reuse the same sheet in multiple html pages. Reduced file size means reduced bandwidth, which means faster loading time. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, etc.

Files that contain the .cs file extension are files that contain source code written in the C# ("C-Sharp") programming language. C# is a programming language used to create a variety of .NET applications. It is commonly created with the Microsoft Visual C# toolset. The CS files are usually compiled by a csc.exe file. The types of source code that a CS file can contain may include source code for console applications, library files and GUI applications.

For Storing data records or files the database used is SQL. SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system. Large amount of data is retrieved quickly and efficiently. Due to documentation and long establishment over years, it provides a uniform platform worldwide to all its users. It can be used in programs in PCs, server, laptops independent of any platform

Chapter 3

Requirement and Analysis

3.1 Problem Definition

The existing manual Voting system consumes more time for Vote Casting. The voter has to wait for a vote polling station to vote for the right candidate. The election officers have to be checked the voter; this voter can vote in this booth and then check the voter ID present in the voter's list of booths are that information will be present then the voter can vote in that booth. The voter had to stand in the queue to cast his vote. All the work is done on paper ballots so it is very hard to locate a particular candidate, some voters cast their votes for all candidates. To overcome all these problems, we have to implement a web application, which helps vote from anywhere.

3.2 Requirements specification

It is focused on studying the existing system of voting and making sure that the people's vote is counted for fairness in the elective positions. The existing machine is not able to recognize the eligibility of a candidate, so the corrupted officers may misguide the people. The corrupted officers may increase the count of the voting. After voting if any technical problems or damage occurs with the machine, it may lead to the re-election. Online voting will produce less effort and be less labor intensive, as the primary cost and focus primarily on creating managing, and running a secure web voting portal.

An increasing number of voters as individuals will find it easier and more convenient to vote, especially those abroad. In order to provide the user with a feeling of community, the following requirement should be taken care:

- Each user will have to create their own profile that they can log into each time they visit the site.
- If the user does not create or log in to an account, they will only be able to browse, they will not be able to use any of the site's other functionalities.
- Once they create an account the user will be able to Log in and out of the system.

The admin page is also made where the admin can organize the election.

3.3 Planning and scheduling

3.3.1 Planning

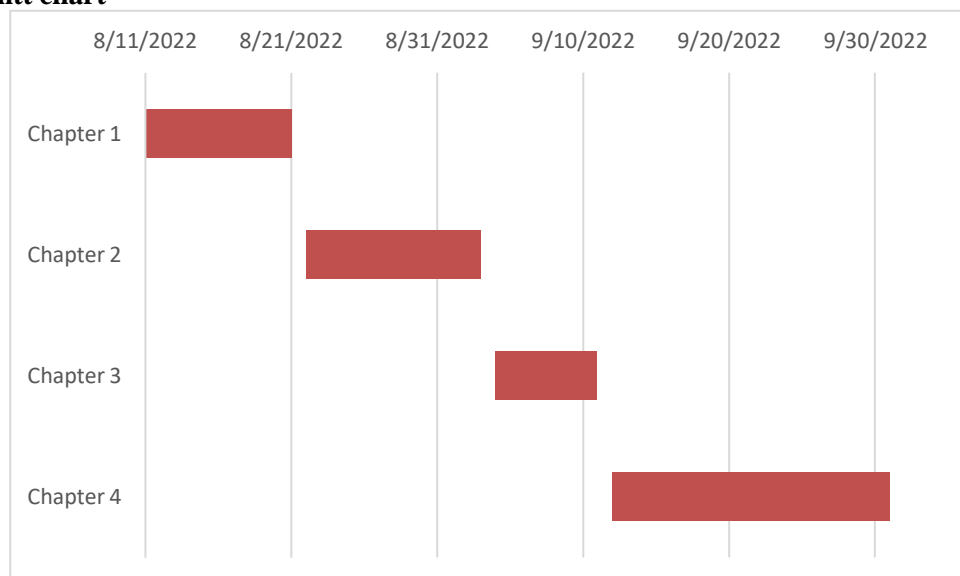
We are planning a web application for the voting process that is voting online. The online voting system will manage the voter's details, Candidate details. The main feature of the project includes voter information and candidate information, voters can log in and use his/her voting rights. The system can manage the information data very efficiently. The proposed system is more reliable, faster, accurate, and easy to handle compared to an existing manual system. It helps to computerize everything and reduce the errors as compared to the manual voting system

3.3.2 Scheduling Gantt Chart

Gantt charts are commonly used for tracking project schedules, **and they are especially useful in project management**. To put it simply, they illustrate and allow you to know what needs to be done, and when it needs to be done. Gantt charts are also able to show you additional information regarding the different tasks or sections of a project, such as how far have tasks progressed, how a group of tasks might depend on other groups of tasks, how important several tasks are, and resources are being used within the project.

One of the great things about Gantt charts is that they are **extremely visual**. On the left side of a chart, you will find a list of tasks necessary for a project, and on the top, you will find a time scale. Here comes the important part: each task is represented by a bar; its length represents the duration of a task. At the same time, each bar is represented by the title of the activity that needs to be completed, along with any relevant information that will help carry it to completion.

Gantt chart



CPM

Critical Path Method(CPM) is a method used in project planning.

It helps in the determination of the earliest time by which the whole project can be completed.

There are two main concepts in this method namely critical task and critical path. Critical task is a task/activity which can't be delayed otherwise the completion of the whole project will be delayed. It must be completed on time before starting the other dependent tasks.

Critical path is a sequence of critical tasks/activities and is the largest path in the project network. It gives us the minimum time which is required to complete the whole project. The activities in the critical path are known as critical activities and if these activities are delayed then the completion of the whole project is also delayed.

Major steps of the Critical Path Method:

1. Identifying the activities
2. Construct the project network
3. Perform time estimation using forward and backward pass
4. Identify the critical path

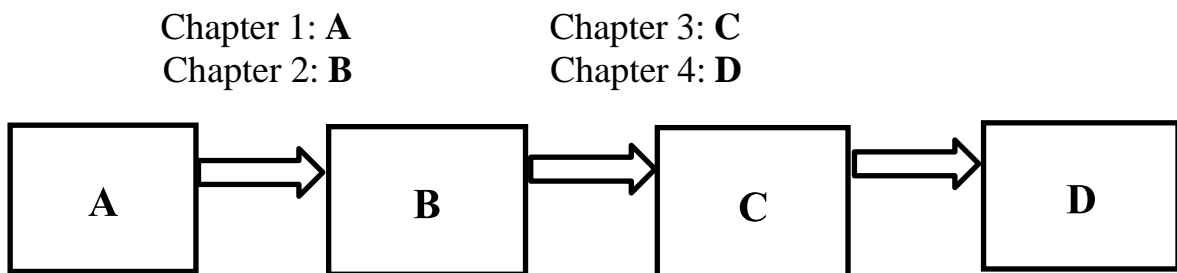


Figure 3.1 Activity

Activity	Precedence	Duration
A	----	10days
B	A	12days
C	B	7days
D	C	13days

Figure 3.2 Critical Activity

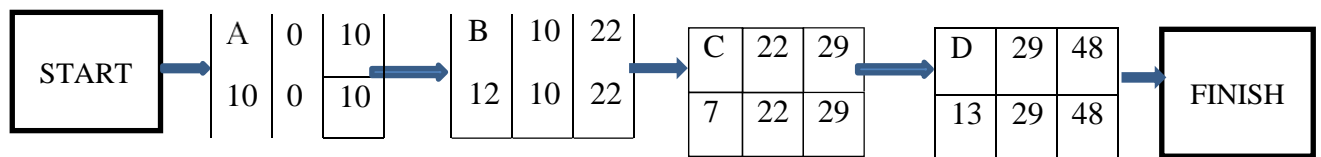


Figure 3.3 Critical Activity

TOTAL= 48 DAYS

Stack value: A=0, B=0, C=0, D=0

Critical path: A-B-C-D

3.4 Proposed System

Here we are proposing a web application for voting process that is voting through online. The online voting system will manage the voter's details, Candidate details. The main feature of the project includes voter's information and candidate information, voter can login and use his/her voting rights. The system can manage the information data very efficiently. The proposed system is more reliable, faster, accurate and easy to handle compared to existing manual system. It helps to computerize everything and reducing the errors as compared to manual voting system. It helps the users to reduce the password forgotten problem, because of face recognition it is easy to login the voting page, instead remembering passwords every time while login.

3.5 Existing System

The existing system is not too effective. At present there are two types of voting methods, they are:

- Ballot Voting
- EVM Voting

3.5.1 Ballot Voting

A ballot is a device used to cast votes in an election and may be a piece of paper used in secret voting. In this the voter is given a paper which consists of all the party symbols along with representative names in it. Here, people come to the polling booth, take the ballot paper and vote by putting a stamp on the desired party symbol. Finally, the ballot paper is folded and dropped into the ballot box. At last, the votes are counted by the Election commission officers.

3.5.2 EVM (Electronic Voting Machine) Voting

An EVM is a device which is used for voting. This machine consists of party symbols along with the representative's name and a button at the end for each and every party name. The voters come near the EVM machine after completion of their verification at the early level before voting. After verification the voter goes near the EVM and casts their vote by pressing the button. The above procedures are not so accurate as there may be possibility for the false/fake voting. The ballot papers may be lost at the time of counting which may affect results of the particular area or people may miscount the number of votes which leads authority into wrong hands. EVM machines sometimes get corrupted, and polling gets stopped temporarily and a lot of time is wasted or EVM may be tampered and the casted votes may be polled to a particular party only, even the vote is casted to different candidates or parties. This may lead authority into the wrong hands. They also lack security as one's vote can be casted by another voter or even a miscellaneous person. This factor is known as fake voting. Without proper authentication there is a possibility of fake voting. So, the existing system is not efficient for voting. Even though there is very little false/fake voting, this minor setback can turn the results in the opposite direction.

It is focused on studying the existing system of voting and to make sure that the people vote is counted for fairness in the elective positions. The existing machine is not able to recognize the eligibility of a candidate, so the corrupted officers may misguide the people. The corrupted officers may increase the count of the voting. After voting if any technical problems or damage occurs with the machine, it may lead to the re-election. Online voting will produce less effort and less labour intensive, as the primary cost and focus primarily on creating managing and running a secure web voting portal. Increasing number of voters as individuals will find it easier and more convenient to vote, especially those abroad. No any heavy budgets to be invested for voting like polling booths, etc.

3.6 Hardware and Software Requirements

3.6.1 Hardware Requirements

Windows:

1. Processor – i3 & above
2. Hard Disk – 500 GB
3. Memory – 4GB RAM

Android:

1. 550 MHz processor
2. 256 GB RAM
3. 3 MB of available disk

3.6.2 Software Requirements

1. Windows 7 Professional or Windows 10.
2. Development Language: Asp.net Razor Page
3. Database: Microsoft SQL Server Management

3.7 Conceptual Models

3.7.1 Entity Relation Diagram

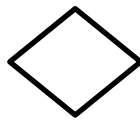
An entity relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure. It is used as a high-level logical data model, which is useful in developing a conceptual design for databases.

Elements used in ER diagram:

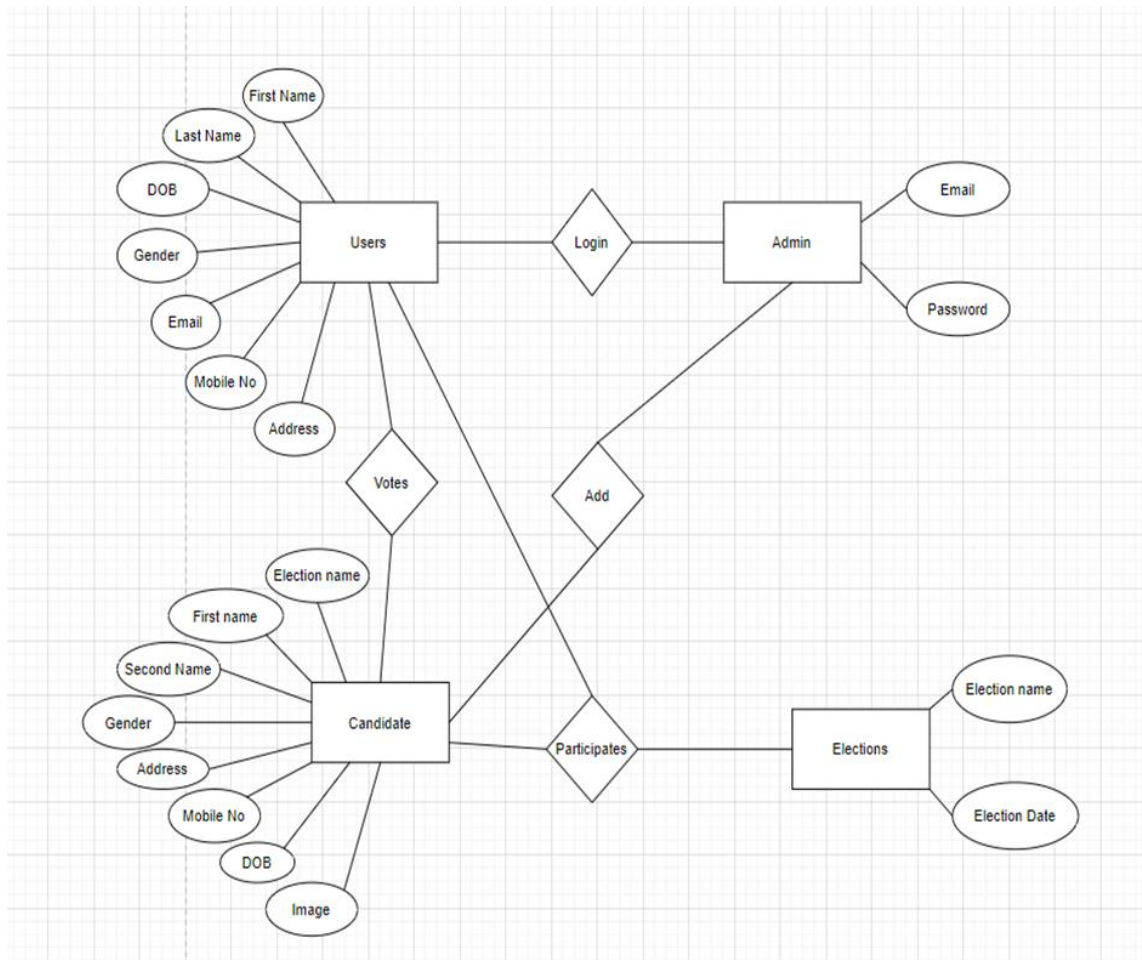
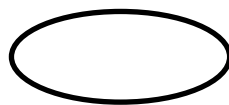
Entity



Relationship



Attribute

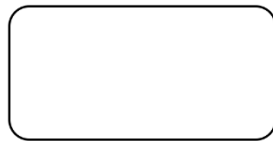


3.7.2 DFD Diagram

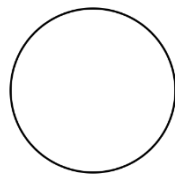
A data flow model is diagrammatic representation of the flow and exchange of information within a system. A DFD is often used as a preliminary step to create an overview of the system which can later be elaborated. DFD can also be used for the visualization of data processing.

Elements of Data Flow diagram:

External Entity



Process



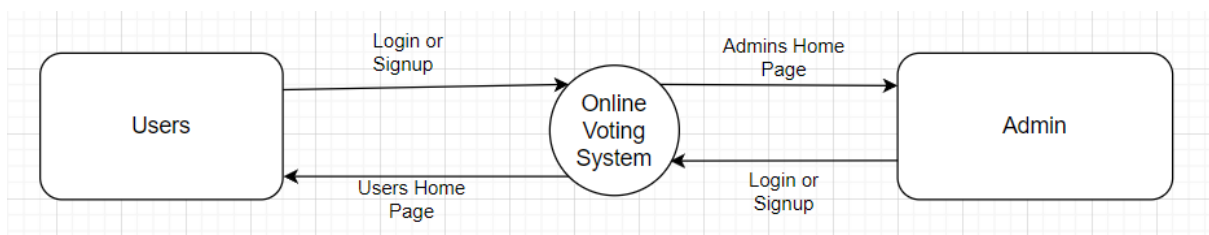
Data store



Data flow

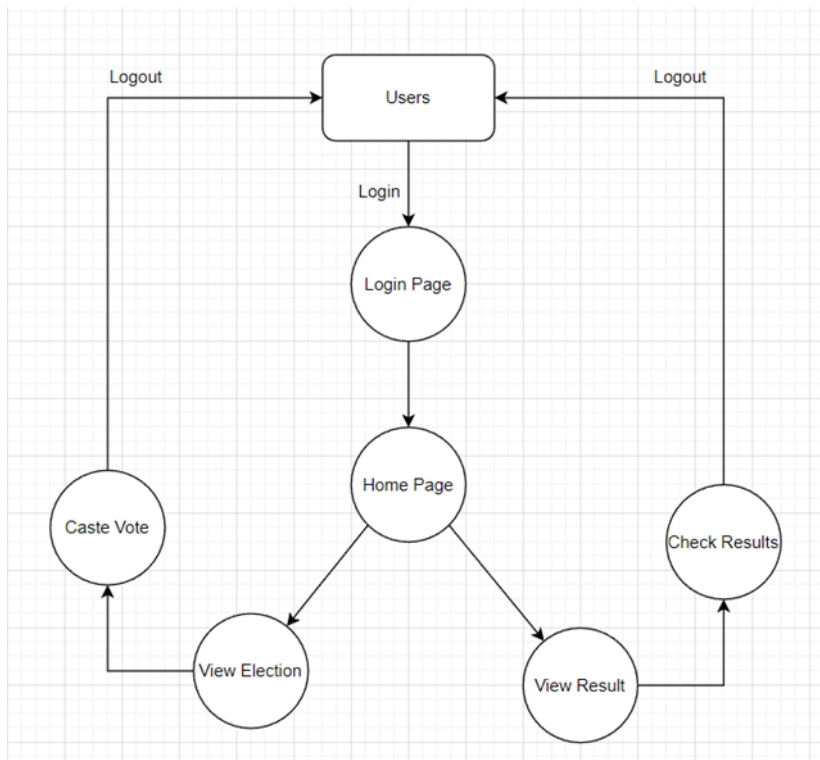


0 level DFD

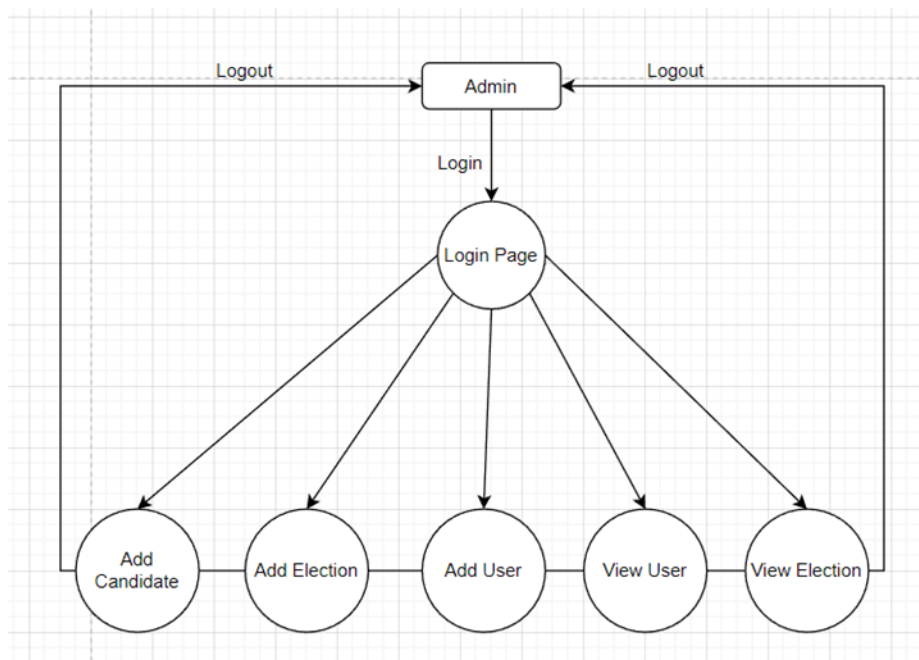


1 level DFD

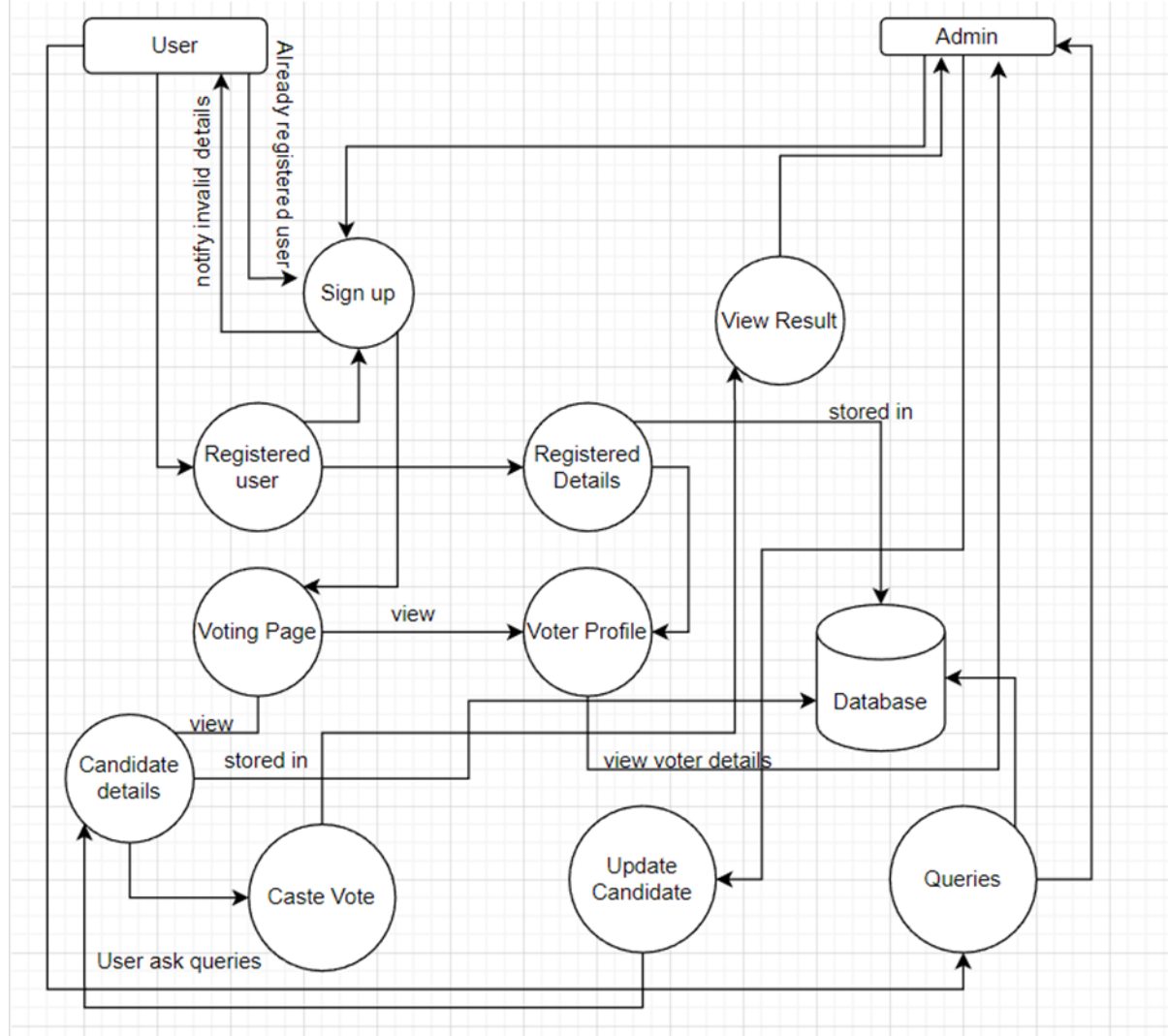
For Voter:



For Admin:



2 level DFD

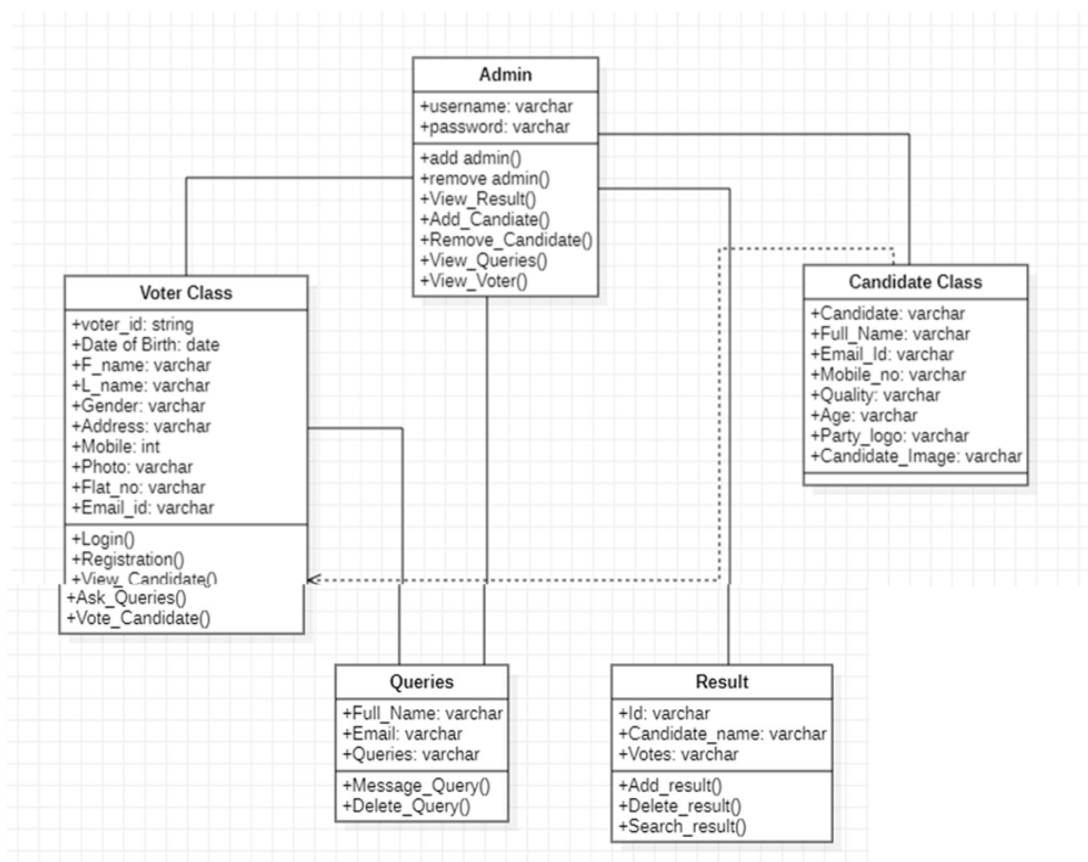


3.7.3 UML Diagram

3.7.3.1 Class Diagram

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods) and the relationships among objects.

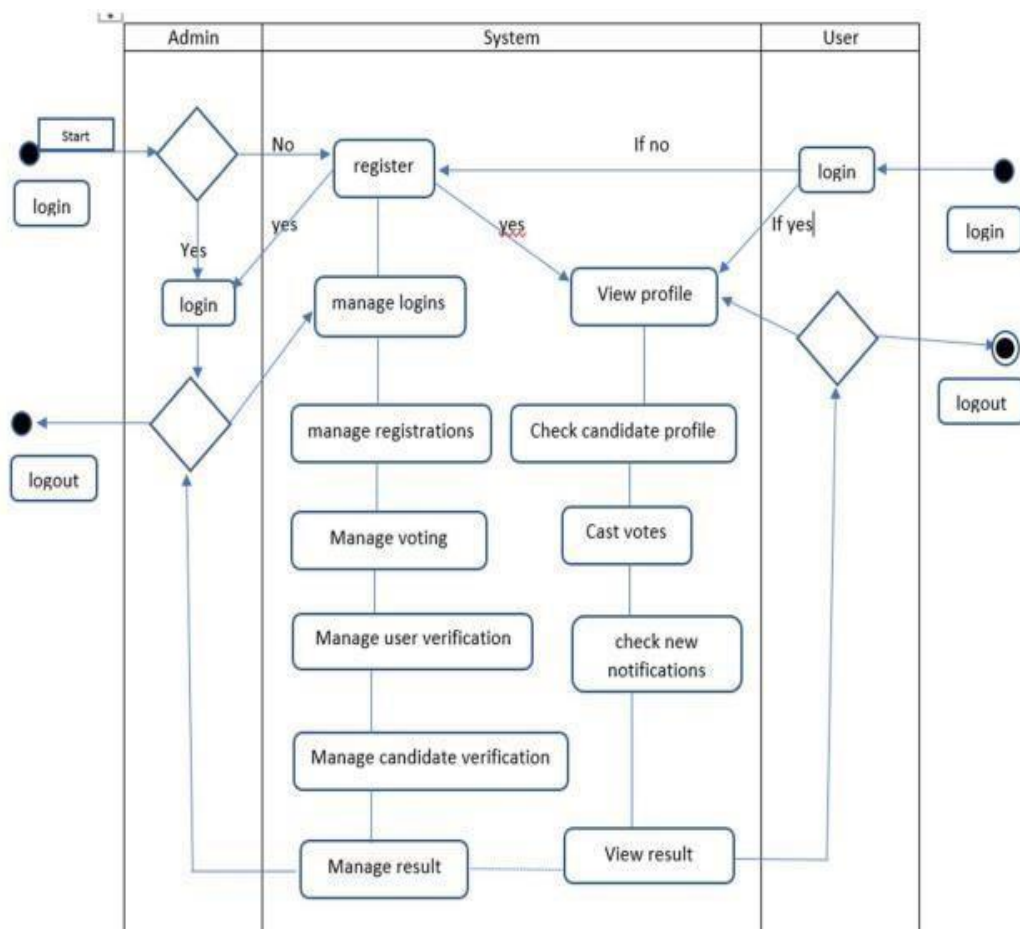
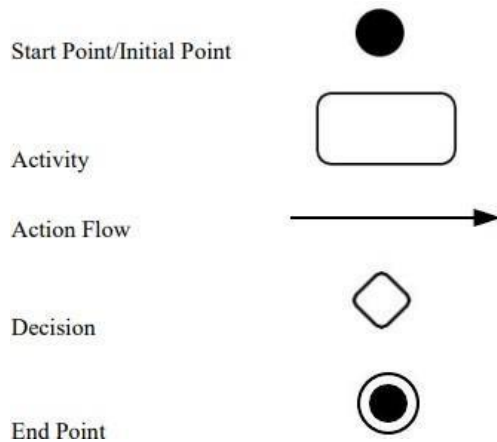
Elements of the class diagram:



3.7.3.2 Activity Diagram

Activity diagram in UML is to describe the dynamic aspects of the system. It is a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

Elements used in the Activity diagram:



3.7.3.3 Use case Diagram:

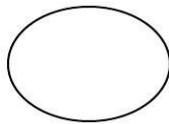
A use case diagram is a graphic depiction of the interactions among the elements of a system. It is used to describe a set of actions (use cases) that some system can perform in collaboration with one or more external users of the system (actors). Each use case provides some observable and valuable result to the actors or other stakeholders of the system. Use case diagrams are twofold: (i) Behavior diagrams because they describe the behavior of the system (ii) Structure diagrams - as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations.

Elements used in Use Case Diagram:

Actor



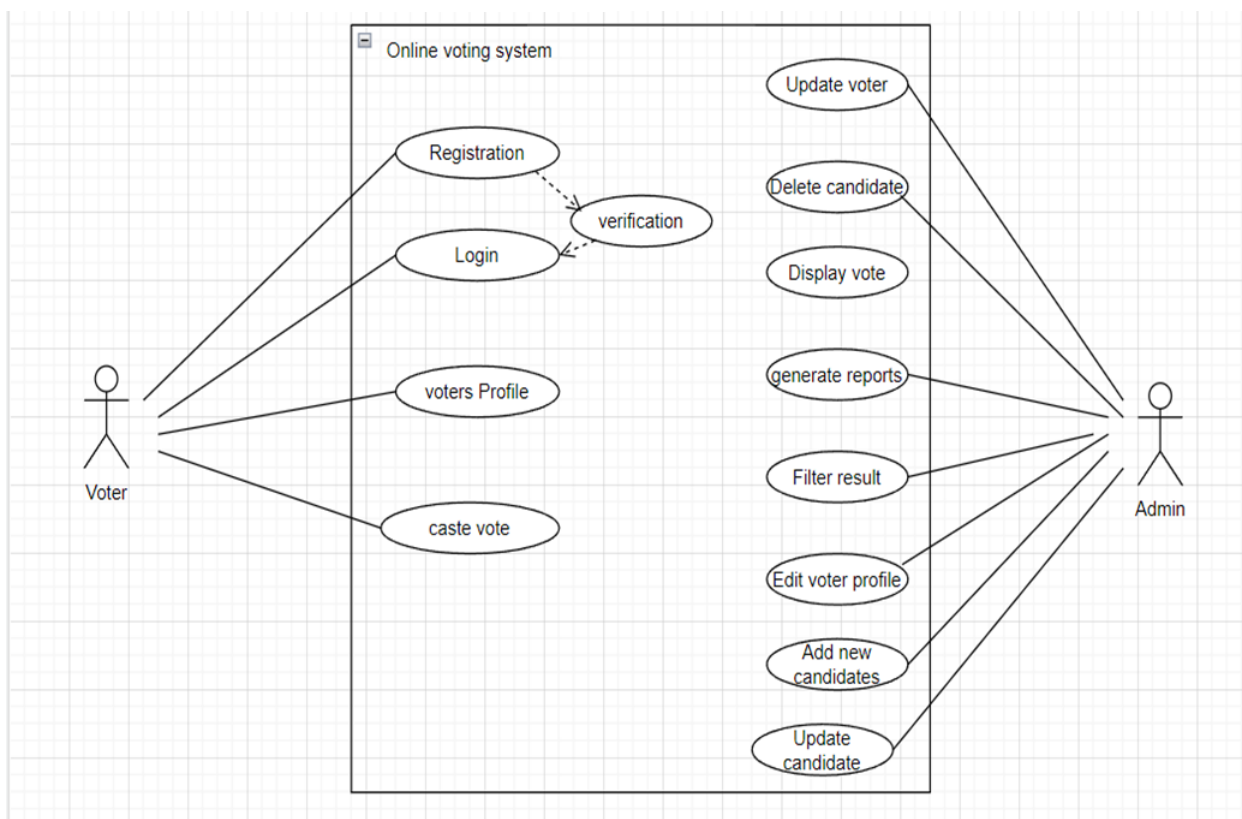
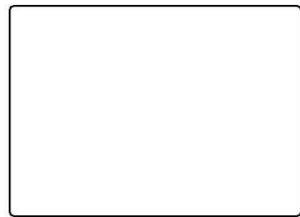
Use Case



Association



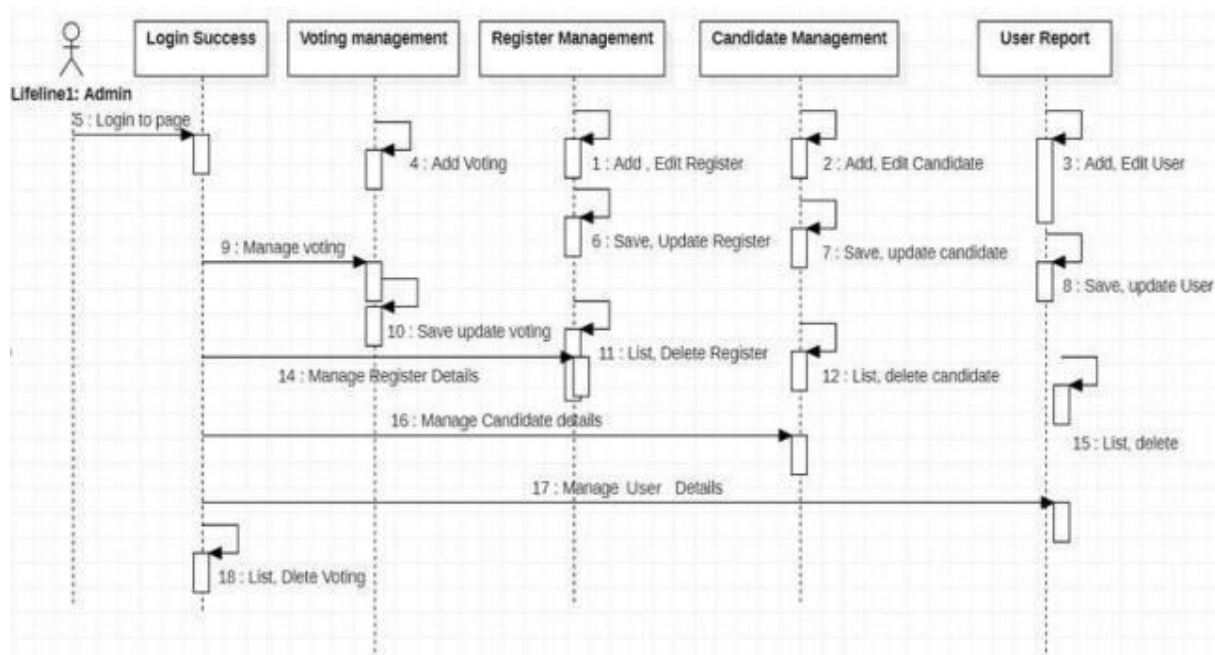
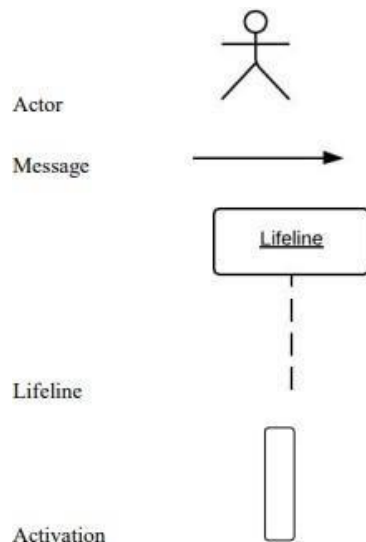
System Boundary



3.7.3.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order, a group of objects works together. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent, and when. These are sometimes known as event diagrams or event scenarios.

Elements of Sequence Diagram:



Chapter 4

System Design

4.1 Basic Modules:

- **Home Page Module**

The Home page has all the modules displayed to the user. It acts as a overview of the whole system. It shows all the modules that can be accessed by the User and the Admin.

- **Login Module**

The Login Module has Login form. This module lets the registered user to login by using their credentials like email id and password. This module helps user to login to the web application.

- **Registration Module**

This module has Registration form for all the users who needs to register their details to cast their vote. This module allows user to create User account.

- **Admin Module**

This page has Login Page for Admin and the Admin have access to Add Candidate Page and ViewResult Page. This Module is Only for Admin i.e., the developer.

4.2 Data Dictionary

1. Admin Table

Name	Type	Key	Description
admin	varchar(20)	Primary key	Login id for Admin
password	Varchar(20)		Password for Login

2. Voter Table

Name	Type	Key	Description
Userid	Int(10)	Primary key	Automatic generate
Dob	date		Date of Birth
Password	Varchar(20)		Password
FirstName	Varchar(25)		Voter first name
LastName	Varchar(25)		Voter last name
Gender	Varchar(10)		Gender of the voter
Address	Varchar(50)		Address
Mobilen0	Int(10)		Mobile number
CanVote	Varchar(3)		Permission to vote
Email_Id	Varchar(50)		E-mail address

3. Election Table

Name	Type	Key	Description
ElectionId	Int(5)	Primary key	Election ID
ElectionName	varchar(20)		Name Of Election
EndDate	date		End date of election

4. Gender Table

Name	Type	Key	Description
GenderValue	Varchar(10)	Primary Key	Gender Value
GenderName	Varchar(50)		Gender Name

5. Candidate Table

Name	Type	Key	Description
Candidate_Id	Int(5)	Primary key	Candidate Id
Firstname	Varchar(30)		Candidate first name
Lastname	Varchar(30)		Candidate last name
Gender	Varchar(10)		Gender
Address	Varchar(50)		Address
Mobile	Int(10)		Mobile number
TypeElection	Int(5)	Foreign Key	Election Id
Date of birth	date		Date of Birth
image	Varchar(50)		Candidate image

6. Election result table

Name	Type	Key	Description
ElectionResultId	Int(5)	Foreign key	Election Result ID
ElectionId	Int(5)		Election ID
CandidateId	Int(5)		Candidate ID
Votes	Int(5)		No of votes

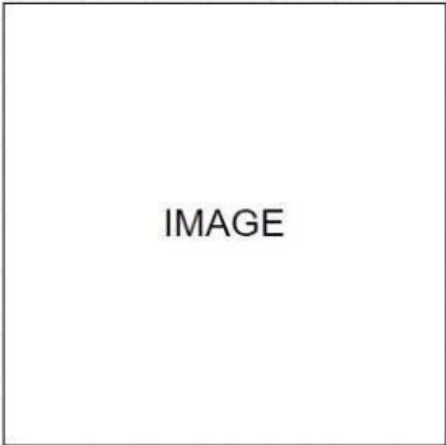
7. Election Votes Table

Name	Type	Key	Description
ElectionVoteId	Int(5)	Primary key	Election Votes ID
UserId	Int(5)	Foreign Key	User ID
CandidateId	Int(5)	Foreign key	Candidate ID
ElectionId	Int(5)	Foreign key	Election ID

4.3 User Interface Design:

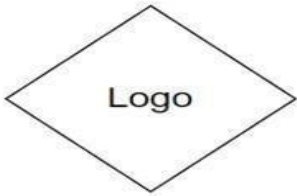
4.3.1 Home Page

Online Voting System		Home	Admin	Login	Register	Contact Us
-----------------------------	--	----------------------	-----------------------	-----------------------	--------------------------	----------------------------



[SIGN UP, To Vote](#)

4.3.2 User Login Page



Email Address

Password

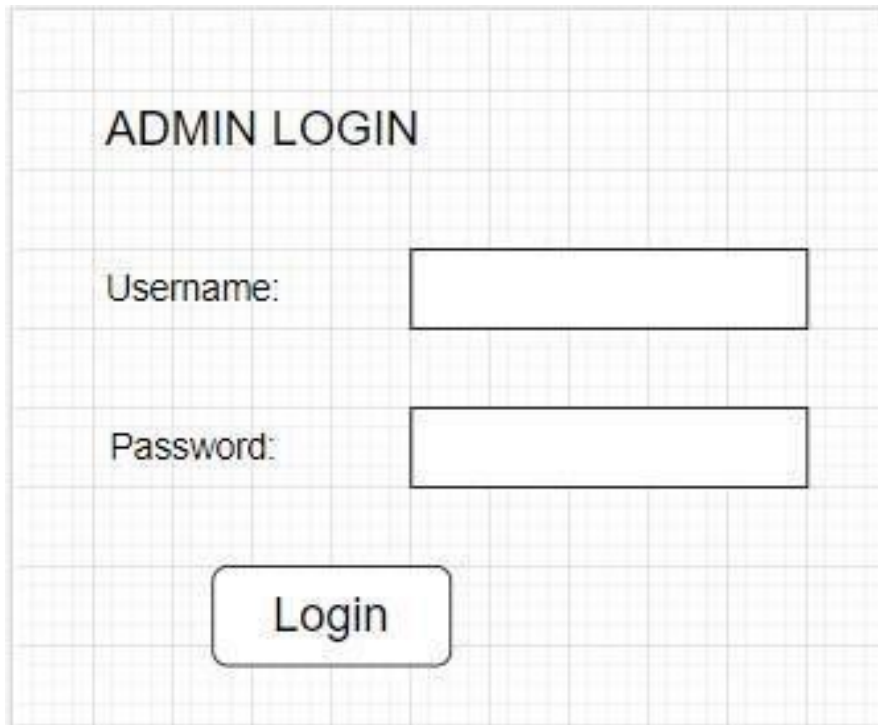
[Login](#)

[Forget Password?](#)

4.3.3 Registration Page

Registration	
First Name	Last Name
<input type="text"/>	<input type="text"/>
Email	Phone Number
<input type="text"/>	<input type="text"/>
Password	Confirm Password
<input type="text"/>	<input type="text"/>
Flat No	Address
<input type="text"/>	<input type="text"/>
DOB	Upload Image
<input type="text"/>	<input type="text"/>
Gender	
<input type="text"/>	
<input type="button" value="SUBMIT"/>	

4.3.4 Admin login Page



A wireframe diagram of an admin login page on a grid background. The title "ADMIN LOGIN" is centered at the top. Below it, the label "Username:" is followed by a rectangular input field. Further down, the label "Password:" is followed by another rectangular input field. At the bottom, a rounded rectangular button labeled "Login" is centered.

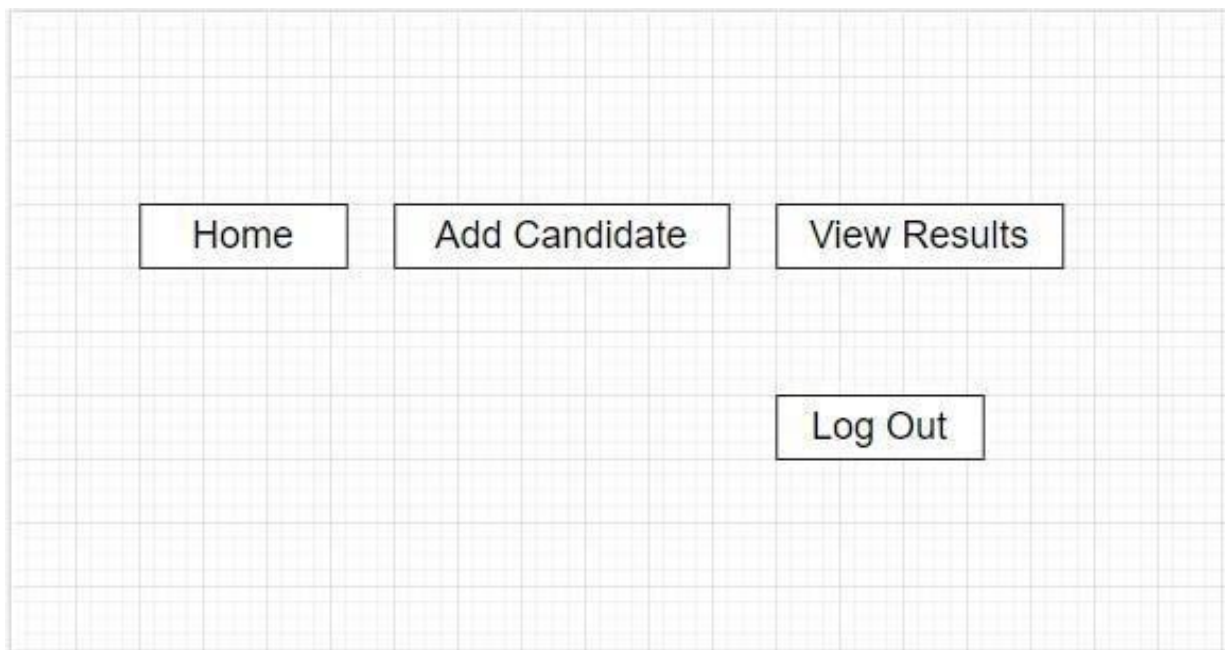
ADMIN LOGIN

Username:

Password:

Login

4.3.5 Admin UI Page



A wireframe diagram of an admin user interface page on a grid background. It features four rectangular buttons arranged in two rows. The top row contains "Home", "Add Candidate", and "View Results" from left to right. The bottom row contains a single "Log Out" button centered horizontally.

Home Add Candidate View Results

Log Out

4.3.6 Add candidate Page

Candidate Name:	<input type="text"/>
Party:	<input type="text"/>
Address:	<input type="text"/>
Phone Number:	<input type="text"/>
Photo:	<div>CANDIDATE PHOTO</div>
<div>SUBMIT</div>	

4.3.7 Voting Page & View Result

Candidate Profile	Candidate Name	Party Name	Vote
<div>Image</div>	<div>Name here</div>	<div>Name here</div>	<div>Vote</div>
<div>Image</div>	<div>Name here</div>	<div>Name here</div>	<div>Vote</div>
<div>Image</div>	<div>Name here</div>	<div>Name here</div>	<div>Vote</div>

4.4 Security Issues:

The most common security issue that raises along with the solution while developing the project are:

SQL injection flaws:

This happens when the developer passes unfiltered data to the SQL server (SQL injection) or the browser. The problem is that the attacker injects the command to the entities which results in loss of data. This problem can be avoided by filtering the inputs properly.

Authentication:

This process runs at the start of the application before accessing the files or the features of the system. The user who wants to access the system should be an authorized user and commonly have a username password combination. Which is known only to the person.

Sensitive data exposure:

Sensitive data should be encrypted at all times. There are chances for people to have access through your personal information, because of which we should be protecting information from being accessed by unauthorized parties. In other words, only the people who are authorized to do so can gain access to sensitive data.

DDoS Attacks

DDoS stands for Distributed Denial of Service and DOS stands for denial of service attacks. They aim to disrupt the website and affect it by flooding the server with numerous requests which leads to the crash of the website.

Chapter 5

Implementation and Testing

5.1 Implementation Approaches:

This online voting system has two ends. One is the admin end and the other is the user end. On the user end, the user can go to the website to register themselves and then log in to the website. There the user can vote to the Candidate they want by selecting the type of Election. The user can see the Candidates image and click on the vote button to cast their vote. The Admin can add Candidates and elections by filling in the essential details. Once the Election is Scheduled, it should have an end date which is also given by the Admin. Also, the Admin can Delete or Edit the candidates. The admin can see all the details provided by the users. As the Deadline of the Election passes the Election result will be displayed to the Users.

Code:

Layout.CSHTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
<title>@ ViewData["Title"] - Onlinevotingsystem</title>
<link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Audiowide">
<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.4.1/css/bootstrap.min.css">
<link rel="icon" style="border-radius:5px" type="image/x-icon" href="~/images/emb.jpg"/>
<link rel="stylesheet" href="~/css/loginstyle.css" />
<link rel="stylesheet" href="~/css/addcandstyle.css" />
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true"
/>
<link rel="stylesheet" href="~/Onlinevotingsystem.styles.css" asp-
append-version="true" />
</head>
<body style="color:black; font-family: Audiowide, sans-serif;">
<header>
<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow
mb-3" style="width:100%">
<div class="container-fluid">
<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Onlinevotingsystem</a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle
```



```

<span class="navbar-toggler-icon"></span>
</button>
<div lass="navbar-collapse collapse d-sm-inline-flex ">
<ul class="navbar-nav flex-grow-1">
<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
</li>
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
</li>
<li class="nav-item">
<a class="nav-link text-dark" asp-area="" asp-controller="Users" asp-action="Login">Logout</a>
</li>
</ul>
</div>
</div>
</nav>
</header>
<div class="container">
<main role="main" class="pb-3">@RenderBody()
</main>
</div>

<div id="lab_social_icon_footer" class="container heading">
<footer>
<!-- Include Font Awesome Stylesheet in Header -->
<link href="//maxcdn.bootstrapcdn.com/font-awesome/4.1.0/css/font-awesome.min.css" rel="stylesheet">
<div class="container page-footer" id="footer">

<div class="text-center center-block">
&copy; 2023 - Onlinevotingsystem - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
<br />
<a href="https://www.facebook.com/login/"><i id="social-fb" class="fa fa-facebook-square fa-3x
social"></i></a>
<a href="https://twitter.com/i/flow/login?input_flow_data=%7B%22requested_vari
ant%22%3A%22eyJY5nIjoiZW4ifQ%3D%3D%22%7D"><i id="social-tw" class="fa fa-twitter-square fa-
3x social"></i></a>
<a href="https://plus.google.com/"><i id="social-gp" class="fa fa-google-plus-square fa-3x social"></i></a>
<a href="https://mail.google.com/mail/u/0/#inbox"><i id="social-em" class="fa fa-envelope-square fa-3x
social"></i></a>
</div>
</div>
</footer>
</div>

<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
<script src="~/js/site.js" asp-append-version="true"></script>@await RenderSectionAsync("Scripts", required:
false)
</body>
</html>

```

Users Registration.CSHTML

@model Onlinevotingsystem.Models.Users

```
@{
    ViewData["Title"] = "Create";
}
```

```
<div class="heading">
<h1 >Create User</h1>
</div>
```

```
<hr />
```

```
<div class="row justify-content-center">
```

```
<div class="col-md-4">
```

```
<form id="off" asp-action="Create" class="frm" >
```

```
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
```

```
<div class="form-group" >
```

```
<label asp-for="FirstName" class="control-label"></label>
```

```
<input asp-for="FirstName" id="FirstName" class="form-
control" />
```

```
    danger"></
```

```
<span asp-validation-for="FirstName" class="text-
```

```
span>
```

```
</div>
```

```
<div class="form-group">
```

```
<label asp-for="LastName" class="control-label"></label><input asp-for="LastName"
control" id="LastName" class="form-
```

```
/>
```

```
<span asp-validation-for="LastName" class="text-
```

```
danger"></span>
```

```
</div>
```

```
<div class="form-group">
```

```
<label asp-for="DateOfBirth" class="control-label"></label>
```

```
<!--<input type="date" asp-for="DateOfBirth" id="DateOfBirth" class="form-control" name="DOB" />-->
```

```
<input type="date" id="datemax" name="datemax" max="1979-12-31" class="form-control">
```

```
<span asp-validation-for="DateOfBirth" class="text-
danger"></span>
```

```
</div>
```

```
<div class="form-group">
```

```
<label asp-for="Gender" class="control-label"> </label>
```

```
<input asp-for="Gender" type="radio" name="gender" value="F" id="gender" />Female
```

```
<input asp-for="Gender" type="radio" name="gender" value="M" id="gender" />Male
```

```
<input asp-for="Gender" type="radio" name="gender" value="O" id="gender" />Other
```

```
@*<input asp-for="gender" type="radio" name="gender" value="M" id="gender" />Male
```

```
<input asp-for="gender" type="radio" name="gender" value="F" id="gender" />Female*@
```

```
<span asp-validation-for="Gender" class="text-
danger"></span>
```

```
</div>
```

```
<div class="form-group">
```

```
<label asp-for="email" class="control-label"></label>
```

```

<input asp-for="email" id="email" class="form-control" />
<span asp-validation-for="email" class="text-
danger"></span>
</div>

<div class="form-group">
<label asp-for="PhoneNo" class="control-label"></label>
<input asp-for="PhoneNo" idPhoneNo class="form-control" pattern="[6-9]{1}[0-9]{9}" />
<span asp-validation-for="PhoneNo" class="text-
danger"></span>
</div>
<div class="form-group">
<label asp-for="address" class="control-label"></label>
<input asp-for="address" id="address" class="form-control"
/>
    danger"></span>
    <span asp-validation-for="address" class="text-
span>

</div>
<br />
<div class="form-group">
<input type="button" onclick="Register()" value="Create" class="btn btn-primary" />
<input type="button" value="Login" onclick="RedirectToLogin()" class="btn btn-primary"
    style="margin-left: 180px" />
</div>
<br />
</form>
</div>
</div>

@*<div>
<a asp-action="Index">Back to List</a>
</div>*@

<script src="~/js/login.js"></script>@section Scripts {
@{ await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

DeleteUser.CSHTML

```

@model Onlinevotingsystem.Models.Users@{
    ViewData["Title"] = "Delete";
}

<h1 class="page-title">Delete</h1>

<h3 class="page-para">Are you sure you want to delete this?</h3>
<div class="blur">
    <h4>Users</h4>
    <hr />

```

```

<dl class="row">
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.FirstName)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.FirstName)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.LastName)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.LastName)
  </dd>
  <dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.DateOfBirth)
  </dt>
  <dd class = "col-sm-10">
    @Html.DisplayFor(model => model.DateOfBirth)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.Gender)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.Gender)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.email)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.email)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.password)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.password)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.PhoneNo)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.PhoneNo)
  </dd>
  <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.address)
  </dt>
  <dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.address)
  </dd>
</dl>

<form asp-action="Delete">
  <input type="hidden" asp-for="Userid" />
  <input type="submit" value="Delete" class="glow-on-hover" /> |
  <a asp-action="Index">Back to List</a>
</form>
</div>

```

UserDetails.CSHTML

```
@model Onlinevotingsystem.Models.Users

@{
    ViewData["Title"] = "Details";
}

<h1 style="color:white">Details</h1>

    <div class="blur">
        <h4>Users</h4>
        <hr />
        <dl class="row" >
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.FirstName)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.FirstName)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.LastName)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.LastName)
            </dd>
            <dt class = "col-sm-2">
                @Html.DisplayNameFor(model => model.DateOfBirth)
            </dt>
            <dd class = "col-sm-10">
                @Html.DisplayFor(model => model.DateOfBirth)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.Gender)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.Gender)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.email)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.email)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.password)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.password)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.PhoneNo)
            </dt>
            <dd class = "col-sm-10"> @Html.DisplayFor(model =>
                model.PhoneNo)
            </dd>
            <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
                model.address)
            </dt>
```

```

        <dd class = "col-sm-10"> @Html.DisplayFor(model =>
            model.address)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Userid">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>

```

EditUser.CSHTML

@model Onlinevotingsystem.Models.Users

```

@{
    ViewData["Title"] = "Vote-Permission";
}

```

<h1 class="page-title">Grant Permission To Vote</h1>

```

<hr />
<div class="row blur">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Userid" />@*<div
            class="form-group">
                <label asp-for="FirstName" class="control-label"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName" class="control-label"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DateOfBirth" class="control-label"></label>
                <input asp-for="DateOfBirth" class="form-control" />
                <span asp-validation-for="DateOfBirth" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Gender" class="control-label"></label>
                <input asp-for="Gender" class="form-control" />
                <span asp-validation-for="Gender" class="text-
danger"></span>
            </div>*@
            <div class="form-group">
                <label asp-for="email" class="control-label"></label>
                <input asp-for="email" class="form-control" />
                <span asp-validation-for="email" class="text-
danger"></span>
            </div>
            @* <div class="form-group">
                <label asp-for="password" class="control-label"></label>

```

```

                <input asp-for="password" class="form-control" />
                <span asp-validation-for="password" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PhoneNo" class="control-label"></label>
                <input asp-for="PhoneNo" class="form-control" />
                <span asp-validation-for="PhoneNo" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="address" class="control-label"></label>
                <input asp-for="address" class="form-control" />
                <span asp-validation-for="address" class="text-
danger"></span>
            </div>*@
            <div class="form-group">
                <label asp-for="CanVote" class="control-label"></label>

                <input asp-for="CanVote" type="checkbox" id="CanVote" />Can
vote

            </div>
            <div class="form-group" style="text-align:center;">
                <input type="submit" value="Save" class="glow-on-hover" />
            </div>
        </form>
    </div>
</div>

<div style="text-align:center; margin-bottom:10%;">
    <a asp-action="Index" >Back to List</a>
</div>
<script>
    $(document).ready(function () {
        //$("#CanVote").is('[disabled]');
        $("#CanVote").prop("disabled", true)
    });
</script>
<script src="~/js/login.js"></script>@section
Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

ForgotPassword.CSHTML

```

@model Onlinevotingsystem.Models.ForgotPass@{
    ViewData["Title"] = "Forgot_Password";
}
<h4 class="page-title heading">Change Password</h4>
<hr />
<div class="row blur heading">
    <div class="col-md-4"><br />
        <form asp-action="ForgotPassword">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <input asp-for="email" placeholder="Enter email" id="email" class="form-control"

```

```

/>
        <span asp-validation-for="email" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <input asp-for="phoneNo" placeholder="Mobile number"
id="phoneNo" class="form-control" />
        <span asp-validation-for="phoneNo" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <input asp-for="newPassword" placeholder="New Password"
id="newPassword" class="form-control" />
        <span asp-validation-for="newPassword" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <button asp-action="UpPassword" style="color:white" asp-route-id="1" asp-
route-id2="2" asp-route-id3="3" class="glow-on- hover">Update</button>
    </div>
</form>
</div>
</div>

<@*div>
    <a asp-action="Login">Back to List</a>
</div>*@
<script type="text/javascript">function
    changep() {
        var email = $("#email").val();
        var phoneNo = $("#phoneNo").val();
        var newPassword = $("#newPassword").val();
        $.post("/Users/UpPassword", { email: email, phoneNo: phoneNo,newPassword:
newPassword }, function (data) {

        });
        alert("Your PassWord have been updated !..");
    }
</script>
<script src="~/js/login.js"></script>@section
Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

UserHome.CSHTML

```

<div class="row blur">

    <p class="addc" id="addc" style=" text-shadow:1px 1px 0
#444;color:ghostwhite; background-color:lawngreen;border-
radius:8px;">@TempData["logs"]</p><br />

```



```

        <hr />
    <p>
    <h1 class="page-title">Welcome Back!</h1>
    <button class="glow-on-hover" style="padding-right:8px"
onclick="ViewElectionsusr()">

        View Elections</button>
    <button class="glow-on-hover" onclick="ViewResult()">ViewResults</button>
</p>
</div>

<script src="~/js/login.js"></script>
<script type="text/javascript"> function
    ViewElectionsusr() {
        $.post("/Elections/UserElection", null, function (data) {

            alert("This is your election list")
        });
        var userurl = "/Election/UserElection";
        window.location.href = userurl;
    }
    function ViewResult(){
        $.post("/ElectionVotes/VuScore", null, function (data) {

            alert("This is your election list")
        });
        var userurl = "/Election/UserElection";
        window.location.href = userurl;
    }
</script> @section
Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

ViewUser.CSHTML

```

@model IEnumerable<Onlinevotingsystem.Models.Users>

@{
    ViewData["Title"] = "Index";
}

<h1 class="page-title">View Users</h1>

<button class="glow-on-hover">
    <a asp-action="Create">Create New User </a>
</button>

<table class="table blur" >
    <p class="addc heading" id="addc" style=" text-shadow:1px 1px 0#444;color:black;
background-color:red;border-radius: 8px;">@TempData["alv"]</p>
    <p class="addc heading" id="addc" style=" text-shadow:1px 1px 0#444;color:black;
background-color:red;border-radius: 8px;">@TempData["Vpf"]</p>
    <p class="addc heading " id="addc" style=" text-shadow:1px 1px 0#444;color:ghostwhite;
background-color:lawngreen;border-radius: 8px;">@TempData["alvs"]</p>
    <p class="addc heading" id="addc" style=" text-shadow:1px 1px 0
#444;color:ghostwhite; background-color:lawngreen;border-radius:
8px;">@TempData["Vps"]</p>

```

```

<thead>
  <tr>
    <th>
      @Html.DisplayNameFor(model => model.FirstName)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.LastName)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.DateOfBirth)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Gender)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.email)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.password)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.PhoneNo)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.address)
    </th>
  </tr>
</thead>
<tbody>
@foreach (var item in Model) {
  <tr>
    <td>
      @Html.DisplayFor(modelItem => item.FirstName)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.LastName)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.DateOfBirth.Month)/
      @Html.DisplayFor(modelItem => item.DateOfBirth.Day)/
      @Html.DisplayFor(modelItem => item.DateOfBirth.Year)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Gender)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.email)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.password)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.PhoneNo)
    </td>
  </tr>
}

```

```

                @Html.DisplayFor(modelItem => item.address)
            </td>
        </td>
        <a style="background-color:lightblue;border-
radius:3px;padding:1px ;color:black" asp-action="Edit" asp-route-
id="@item.Userid">Edit</a> <br>
        <a style="background-color:floralwhite;border-
radius:3px;padding:1px ;color:black" asp-action="Details" asp-route-
id="@item.Userid">Details</a> <br>
        <a style="background-color:darkkhaki;border-
radius:3px;padding:1px ;color:black" asp-action="Delete" asp-route-
id="@item.Userid">Delete</a>
    </td>
</tr>
}
</tbody>
</table>

```

UserLogin.CSHTML

```

@model Onlinevotingsystem.Models.UserLogin@{
    ViewData["Title"] = "Login";
    // Layout = null;
}
<h4 class="page-title">Login</h4>
<hr />
<div class="row blur" >
    <div class="col-md-4">
        <form asp-action="Login">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <p class="addc heading" id="addc" style=" text-shadow:1px 1px
0 #444;color:black; background-color:red;border-
radius:8px;">@TempData["nop"]</p>
            <p class="addc heading" id="addc" style=" text-shadow:1px1px 0
#444;color:black; background-color:red;border- radius:8px;">@TempData["logf"]</p>
            <p class="addc" id="addc" style=" text-shadow:1px 1px 0
#444;color:ghostwhite; background-color:lawngreen;border-
radius:8px;">@ViewData["fps"]</p>
            <div class="form-group">
                <input asp-for="email" placeholder="Email" class="form-control" id="email"
/>
                <span asp-validation-for="email" class="text-
danger"></span>
            </div>
            <div class="form-group">
                <input asp-for="password" placeholder="Password" class="form-
control" id="password"/>
                <span asp-validation-for="password" class="text-
danger"></span>
            </div>
        </form>
    </div>
</div>
<div class="form-group btn-group-sm" style="text-align:center;margin-top:18px">
    <button title="If New USer ? " style="color:white" class="glow-on-hoverheading" value="Register"
onclick="addUser()">Register</button>--

```

```

        <button value="Login" style="color:white" class="glow-on-hover heading" id="btn-id"
onclick="Validate()">Login</button>--
        <button title="If Forgot Password ? " style="color:white" class="glow-on-hover heading"
value="Forgot Password" onclick="UpdatePassWord()">Forgot-Password</button>
</div>

```

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

AdminHome.CSHTML

```

@{
    ViewData["Title"] = "Admin";
}

<div class="text-center ">

    <p class="addc" id="addc" style=" text-shadow:1px 1px 0
#444;color:ghostwhite; background-color:lawngreen;border-
radius:8px;">@TempData["logs"]</p>
    <button onclick="addcandidate()" class="glow-on-hover"
id="container">add candidate</button>
    <button onclick="addelection()" class="glow-on-hover"
id="container">add election</button>
    <button onclick="addUser()" class="glow-on-hover" id="container">addUser</button>
    <button onclick="ViewUser()" class="glow-on-hover" id="container">viewUser</button><br
/><hr />
    <button onclick="ViewElections()" class="glow-on-hover"
id="container">view election</button>

</div>
<div class="page-title">
<h1 >Welcome</h1>
<p >
    Welcome to Admin Dashboard
</p>
</div>
<script src="~/js/login.js"></script>@section
Scripts {
    @{
        await Html.RenderPartialAsync("_ValidationScriptsPartial");
    }
}

```

Privacy.CSHTML

```

@{
    ViewData["Title"] = "Privacy Policy";
}

```

```

}
<body class="bgprivacy">
  <h1 class="page-title">@ ViewData["Title"]</h1>
  <div class="card" style="background: linear-gradient(5deg, grey,
transparent);">

    <div class="card-header">
      </div>
    <div class="card-body">
      <h5 class="card-title" style="color:white">Special title
treatment</h5>
      <p class="card-text" style="color:white">

```

Voting is commonly associated with the sovereign expression of a voter's will—an expression which multiple theorems from voting theory have shown not to be necessarily aligned with their true desires. We now understand that an important precondition for the expression of a voter's will is the privacy of the ballot, which was not always the case and is still debated—e.g. for ballot selfies. However, most individuals' voting experiences happen not in national elections but in small settings such as boardrooms or union meetings—which form the main client base of Neovote.

This creates new issues when it comes to privacy, as authority figures can often easily exert direct coercive power if the vote goes against their wishes. The coercive power of managers within a company—and the related potential for vote buying—should not be neglected, as they are central to large-scale voting fraud campaigns such as those in Russia. Thus, if we seek to change or enforce norms that guarantee voters' privacy, smaller-scale votes are an immediate target. The potential for coercion also means that such settings require even more stringent guarantees of privacy, such as publishing not the full tally (from which one could infer some information) but only the result. Before the Covid-19 pandemic, most votes of this kind still happened with paper ballots. The switch to remote work has made this impossible in many settings, leading to the development and partial adoption of many e-voting systems. Some focus on security and privacy, whereas others are purposefully designed with usability and understandability in mind, discarding entirely the question of coercion

```

</p>

```

```

</div>

```

```

</div>
</body>

```

CreateElections.CSHTML

```

@model Onlinevotingsystem.Models.Election@{
  ViewData["Title"] = "Create";
}

<h1 class="page-title">Create Election</h1>

<hr />
<div class="row blur">
  <div class="col-md-4" id= "form_center">

```

```

        <form asp-action="Create" >
            <p class="addc heading" id="addc" style=" text-shadow:1px 1px 0#444;color:black;
background-color:red;border- radius:8px;">@TempData["elef"]</p>
            <p class="addc heading" id="addc" style=" text-shadow:1px 1px 0#444;color:White;
background-color:lawngreen;border- radius:8px;">@TempData["eles"]</p>
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group" >
                <input asp-for="ElectionName"placeholder="Election Name"class="form-
control" />
                <span asp-validation-for="ElectionName" class="text-
danger"></span>
            </div>
            <div class="form-group" >
                <input type="date" asp-for="EndDate" placeholder="End Date" class="form-
control" min="2023-01-02"><br><br>
                <span asp-validation-for="EndDate" class="text-
danger"></span>
            </div>
            <div class="form-group" style="text-align:center">
                <input type="submit" value="Create" class="glow-on-hover"
/>
            </div>
        </form>
    </div>
</div>
<br />
<div style="text-align:center">
    <a class="page-para" asp-action="Index"> Elections</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

DeleteElections.CSHTML

```

@model Onlinevotingsystem.Models.Election

@{
    ViewData["Title"] = "Delete";
}

<h1 class="page-title">Delete</h1>

<h3 class="page-para">Are you sure you want to delete this?</h3>
<div class="blur">
    <h4>Election</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.ElectionName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ElectionName)

```

```

        </dd>
        <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
            model.EndDate)
        </dt>
        <dd class = "col-sm-10"> @Html.DisplayFor(model =>
            model.EndDate)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Electionid" />
        <input type="submit" value="Delete" class="glow-on-hover" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>

```

ElectionDetails.CSHTML

```

@model Onlinevotingsystem.Models.Election@{
    ViewData["Title"] = "Details";
}

<h1 style="color:white">Details</h1>

<div class="blur">
    <h4>Election</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.ElectionName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ElectionName)
        </dd>
        <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
            model.EndDate)
        </dt>
        <dd class = "col-sm-10"> @Html.DisplayFor(model =>
            model.EndDate)
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Electionid">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>

```

EditElections.CSHTML

```

@model Onlinevotingsystem.Models.Election

```

```

@{
    ViewData["Title"] = "Edit";
}

<h1 style="text-align:center; color:white">Edit Election</h1>

<hr />
<div class="row blur">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Electionid" />
            <div class="form-group">
                <label asp-for="ElectionName" class="control-
label"></label>
                <input asp-for="ElectionName" class="form-control" />
                <span asp-validation-for="ElectionName" class="text-
danger"></span>

            </div>
            <div class="form-group">
                <label asp-for="EndDate" class="control-label"></label>
                <input asp-for="EndDate" class="form-control" />
                <span asp-validation-for="EndDate" class="text-
danger"></span>
            </div>
            <div class="form-group" style="text-align:center;">
                <input type="submit" value="Save" class="glow-on-hover" />
            </div>
        </form>
    </div>
</div>

<div style="text-align:center; margin-bottom:10%;">
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

ViewElection.CSHTML

```

@model IEnumerable<Onlinevotingsystem.Models.Election>@{
    ViewData["Title"] = "Index";
}

<h1 class="page-title" style="text-align:center">View Election</h1>

<p style="text-align:center">
    <a asp-action="Create" >Create New</a>
</p>
<table class="table blur" id="tb">
    <thead>
        <tr>

```



```

        <th>
            @Html.DisplayNameFor(model => model.ElectionName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EndDate)
        </th>
        <th></th>
    </tr>
</thead>
<tbody>
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.ElectionName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EndDate.Month)/
            @Html.DisplayFor(modelItem => item.EndDate.Day)/
            @Html.DisplayFor(modelItem => item.EndDate.Year)
        </td>
        <td>
            <a asp-action="Edit" asp-route-
id="@item.Electionid">Edit</a> |
            <a asp-action="Details" asp-route-
id="@item.Electionid">Details</a> |
            <a asp-action="Delete" asp-route-
id="@item.Electionid">Delete</a>
        </td>
    </tr>
}
</tbody>
</table>

```

UserElectionIndex.CSHTML

```

@model IEnumerable<Onlinevotingsystem.Models.Election>

<p>
    <a asp-action="Create">View Result </a>
</p>
<table class="table blur" id="tb">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.ElectionName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.EndDate)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.ElectionName)
            <p hidden onclick=""> @Html.DisplayFor(modelItem =>
item.Electionid)</p>

```

```

        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EndDate)
        </td>
        <td>
            @*<a asp-action="Edit" asp-route-
id="@item.Electionid">Edit</a> |
            <a asp-action="Details" asp-route-
id="@item.Electionid">Details</a> |
            <a asp-action="Delete" asp-route-
id="@item.Electionid">Delete</a>*@
            <button class="glow-on-hover" onclick="ViewCan()">@*<a asp-
action="VuCandidate" asp-route-
id="@item.Electionid">Cast Vote</a>*@

            </button>
        </td>
    </tr>
}
</tbody>
</table>
<script type="text/javascript">function
ViewCan() {

    $.post("/Elections/VuCandidate", null, function (data) {alert("This is your

        election list")

    });
    var userurl = "/Election/VuCandidate";
    window.location.href = userurl;
}</script>

```

ViewCandidate.CSHTML

```

@model IEnumerable<Onlinevotingsystem.Models.Candidate>

@*<p>
    <a asp-action="Create">Create New</a>
</p>*@
<table class="table blur" id="tb">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.TypeElection)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FirstName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.LastName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.DateOfBirth)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Gender)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Image)
            </th>

```

```

        <th>
            @Html.DisplayNameFor(model => model.PhoneNo)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.address)
        </th>
        <th></th>
    </tr>
</thead>
<tbody>
@foreach (var item in Model) {
    <tr>
        @* <input hidden id="cid" value="@Html.DisplayFor(modelItem
=> item.TypeElection)"/>*@

        <td>
            @Html.DisplayFor(modelItem => item.TypeElection)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.FirstName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.LastName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DateOfBirth)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Gender)
        </td>
        <td>
            
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.PhoneNo)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.address)
        </td>
        <td>
            @* <a asp-action="Details" asp-route-
id="@item.Candidateid">Details</a> |
            <a asp-action="Delete" asp-route-
id="@item.Candidateid">Delete</a>*@
            <button class="glow-on-hover"> <a asp- action="CastVotes" asp-route-
id="@item.Candidateid">Cast Vote</a>
|</button>
        </td>
    </tr>
}
</tbody>
</table>
<div hidden>
    <input id="email" value="@ViewBag.xemail"/>
    <input id="name" value="@ViewBag.yname"/></div>
<script type="text/javascript">function
Vote() {
    var u = $('#email').val();
    var v = $('#name').val();

```

```

    }
</script>

```

ViewScore.CSHTML

```
@model Onlinevotingsystem.Models.Winner
```

```

<div class="blur heading">
    <h4 class="heading">Winner</h4>
    <h3 class="heading">@TempData["wait"]</h3>

    <hr />
    <br />
    <hr />
    <p>
        <b>@Html.DisplayFor(model => model.CandidateName)</b> is the
        <b>Winner in @Html.DisplayFor(model => model.ElectionName)</b> with
        <b>@Html.DisplayFor(model => model.NoofVotes)</b> number of Votes
    ....
    </p>
</div>
@*<div>
    <a asp-action="Edit" asp-route-id="@Model?.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
*@

```

CreateCandidate.CSHTML

```
@model Onlinevotingsystem.Models.Candidate
```

```

@{
    ViewData["Title"] = "Create";
}
<div class="heading">
    <h1 class="page-title" style="text-align:center">Create Candidate
</h1>
</div>

<hr />
<div class="row justify-content-center blur">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <p class="addc" id="addc" style=" text-shadow:1px 1px 0 #444;color:black;
background-color:red;border- radius:8px">@TempData["cf"]</p>
            <p class="addc" id="addc" style=" text-shadow:1px 1px 0
#444;color:ghostwhite; background-color:lawngreen;border-
radius:8px">@TempData["suc"]</p>

```

```

        @*<div class="form-group">
            <label asp-for="TypeElection" class="control-
label"></label>
            <input asp-for="TypeElection" class="form-control" />
            <span asp-validation-for="TypeElection" class="text-
danger"></span>

        </div>*@
    <div class="form-group" >

        <select asp-for="TypeElection" asp-items="@{new
SelectList(@ViewBag.Election,"Electionid","ElectionName"))" class="coIdform-control"
id="Election"></select>
        <span asp-validation-for="TypeElection" class="text-
danger"></span>
    </div>

    <div class="form-group">
        <input asp-for="FirstName" placeholder="First Name" class="form-
control" />
        <span asp-validation-for="FirstName" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <input asp-for="LastName" placeholder="Last Name" class="form-
control" />
        <span asp-validation-for="LastName" class="text-
danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="DateOfBirth" class="control-label font-effect-
outline"></label>
        <input type="date" asp-for="DateOfBirth" id="datemax" name="datemax"
max="1979-12-31" class="form-control" />
        @*<input type="date" id="datemax" name="datemax" max="1979-12-31"
class="form-control">*@
        <span asp-validation-for="DateOfBirth" class="text-
danger"></span>

    </div>
    <div class="form-group">
        <label asp-for="Gender" class="control-label font-effect-outline"></label>
        <input asp-for="Gender" type="radio" name="gender" value="F"
id="gender" />Female
        <input asp-for="Gender" type="radio" name="gender" value="M"
id="gender" />Male
        <input asp-for="Gender" type="radio" name="gender" value="O"
id="gender" />Other
        @*<input asp-for="gender" type="radio" name="gender" value="M"
id="gender" />Male
        <input asp-for="gender" type="radio" name="gender" value="F"
id="gender" />Female*@
        <span asp-validation-for="Gender" class="text-
danger"></span>
    </div>

    <div class="form-group">
        <input type="file" accept="Image/*" asp-for="Image" class="form-
control" />
        <span asp-validation-for="Image" class="text-
danger"></span>

```

```

        </div>
        <div class="form-group">
            <input asp-for="PhoneNo" placeholder="Mobile number" class="form-
control" />
            <span asp-validation-for="PhoneNo" class="text-
danger"></span>
        </div>
        <div class="form-group">
            <input asp-for="address" placeholder="address" class="form-
control" />
            <span asp-validation-for="address" class="text-
danger"></span>

        </div>
        <br />
        <div class="form-group" style="text-align:center">
            <button value="Create" class="glow-on-hover" >Create
Candidate</button>
        </div>
    </form>
</div>
<br />

```

```

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

DeleteCandidate.CSHTML

```

@model Onlinevotingsystem.Models.Candidate

@{
    ViewData["Title"] = "Delete";
}

<h1 class="page-title">Delete</h1>

<h3 class="page-para">Are you sure you want to delete this?</h3>
<div class="blur">
    <h4>Candidate</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.TypeElection)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.TypeElection)
        </dd>
        <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
            model.FirstName)
        </dt>
        <dd class = "col-sm-10"> @Html.DisplayFor(model =>
            model.FirstName)
        </dd>
        <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
            model.LastName)
        </dt>
        <dd class = "col-sm-10">

```

```

        @Html.DisplayFor(model => model.LastName)
    </dd>
    <dt class = "col-sm-2">
        @Html.DisplayNameFor(model => model.DateOfBirth)
    </dt>
    <dd class = "col-sm-10">
        @Html.DisplayFor(model => model.DateOfBirth)
    </dd>
    <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
        model.Gender)
    </dt>
    <dd class = "col-sm-10"> @Html.DisplayFor(model =>
        model.Gender)
    </dd>

    <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
        model.Image)
    </dt>
    <dd class = "col-sm-10"> @Html.DisplayFor(model =>
        model.Image)
    </dd>
    <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
        model.PhoneNo)
    </dt>
    <dd class = "col-sm-10"> @Html.DisplayFor(model =>
        model.PhoneNo)
    </dd>
    <dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
        model.address)
    </dt>
    <dd class = "col-sm-10"> @Html.DisplayFor(model =>
        model.address)
    </dd>
</dl>

<form asp-action="Delete">
    <input type="hidden" asp-for="Candidateid" />
    <input type="submit" value="Delete" class="glow-on-hover" /> |
    <a asp-action="Index">Back to List</a>
</form>
</div>

```

CandidateDetails.CSHTML

```

@model Onlinevotingsystem.Models.Candidate@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div class="blur">
    <h4>Candidate</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.TypeElection)
        </dt>

```

```

<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.TypeElection)
</dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.FirstName)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.FirstName)
</dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.LastName)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.LastName)
</dd>
<dt class = "col-sm-2">
    @Html.DisplayNameFor(model => model.DateOfBirth)
</dt>
<dd class = "col-sm-10">
    @Html.DisplayFor(model => model.DateOfBirth)
</dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.Gender)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.Gender)
</dd>

<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.Image)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.Image)
</dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.PhoneNo)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.PhoneNo)
</dd>
<dt class = "col-sm-2"> @Html.DisplayNameFor(model =>
    model.address)
</dt>
<dd class = "col-sm-10"> @Html.DisplayFor(model =>
    model.address)
</dd>
</dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Candidateid">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>

```

EditCandidate.CSHTML

@model Onlinevotingsystem.Models.Candidate

```
@{
    ViewData["Title"] = "Edit";
}
```



```

}

<h1 style="text-align:center;">Edit Candidate</h1>

<hr />
<div class="row blur">
  <div class="col-md-4">
    <form asp-action="Edit">
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <input type="hidden" asp-for="Candidateid" />
      <div class="form-group">
        <label asp-for="TypeElection" class="control-
label"></label>
        <input asp-for="TypeElection" class="form-control" />
        <span asp-validation-for="TypeElection" class="text-
danger"></span>

      </div>
      <div class="form-group">
        <label asp-for="FirstName" class="control-label"></label>
        <input asp-for="FirstName" class="form-control" />
        <span asp-validation-for="FirstName" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="LastName" class="control-label"></label>
        <input asp-for="LastName" class="form-control" />
        <span asp-validation-for="LastName" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="DateOfBirth" class="control-label"></label>
        <input asp-for="DateOfBirth" class="form-control" />
        <span asp-validation-for="DateOfBirth" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="Gender" class="control-label"></label>
        <input asp-for="Gender" class="form-control" />
        <span asp-validation-for="Gender" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="Image" class="control-label"></label>
        <input asp-for="Image" class="form-control" />
        <span asp-validation-for="Image" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="PhoneNo" class="control-label"></label>
        <input asp-for="PhoneNo" class="form-control" />
        <span asp-validation-for="PhoneNo" class="text-
danger"></span>
      </div>
      <div class="form-group">
        <label asp-for="address" class="control-label"></label>
        <input asp-for="address" class="form-control" />
        <span asp-validation-for="address" class="text-
danger"></span>
    </form>
  </div>
</div>

```

```

        </div>
        <div class="form-group" style="text-align:center;" >
            <input type="submit" value="Save" class="glow-on-hover" />
        </div>
    </form>
</div>
</div>
<br>
<div style="text-align:center; margin-bottom:10%;">
    <a asp-action="Index" >Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}

```

Candidate.CS

```

using System.ComponentModel.DataAnnotations;

namespace Onlinevotingsystem.Models
{
    public class Candidate
    {
        [Key]
        public int Candidateid { get; set; } public int

        TypeElection { get; set; }

        [Required(ErrorMessage = "Please enter Firstname !")] [Display(Name =
        "FirstName ")]
        public string? FirstName { get; set; }

        [Required(ErrorMessage = "Please enter Lastname !")] [Display(Name =
        "LastName ")]
        public string? LastName { get; set; }

        [Required(ErrorMessage = "Please pick birthdate")] [Display(Name =
        "DateOfBirth ")]
        public DateTime DateOfBirth { get; set; }

        [Required(ErrorMessage = "Please select gender !")] [Display(Name =
        "Gender ")]
        public char Gender { get; set; }

        [Required(ErrorMessage = "Please enter Image !")] [Display(Name =
        "Image ")]
        public string? Image { get; set; }

        [Required(ErrorMessage = "Please enter mobile no !")] [Display(Name =
        "PhoneNo ")]

```

```
[RegularExpression(@"^([6-9]{1}[0-9]{9})$", ErrorMessage = "InvalidPhone Number")]
public string? PhoneNo { get; set; } [Required(ErrorMessage = "Please
enter address!")]

[Display(Name = "address ")]
public string? address { get; set; }
}
```

Election.CS

```
using System.ComponentModel.DataAnnotations;

namespace Onlinevotingsystem.Models
{
    public class Election
    {
        [Key]
        public int Electionid { get; set; }

        [Required(ErrorMessage = "Please enter Election Name !")] [Display(Name = "ElectionName ")]
        public string? ElectionName { get; set; }

        [Required(ErrorMessage = "Please pick EndDate")] [Display(Name = "EndDate")]
        public DateTime EndDate { get; set; }
    }
}
```

ElectionResult.CS

```
namespace Onlinevotingsystem.Models
{
    public class ElectionResult
    {
        [System.ComponentModel.DataAnnotations.Key] public int
        ElectionResultId { get; set; }

        public int ElectionId { get; set; }

        public int CandidateId { get; set; } public int votes {
        get; set; }
    }
}
```

ElectionVotes.CS

```
using MessagePack;
using System.ComponentModel.DataAnnotations;
```

```

namespace Onlinevotingsystem.Models
{
    public class ElectionVotes
    {
        [System.ComponentModel.DataAnnotations.Key]public int
        ElectionVoteId { get; set; } public int UserId { get; set; }

        public int ElectionId { get; set; }public int CandidateId {

            get; set; }

    }
}

```

ErrorView.CS

```

namespace Onlinevotingsystem.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

    }
}

```

ForgotPassword.CS

```

using System.ComponentModel.DataAnnotations;

namespace Onlinevotingsystem.Models
{
    public class ForgotPass
    {
        [Key]
        public int Id { get; set; } public string email {
            get; set; }
        public string phoneNo { get; set; } public string
        newPassword { get; set; }

    }
}

```

Gender.CS

```

using System.ComponentModel.DataAnnotations;namespace
Onlinevotingsystem.Models

```

```

{
    public class Gen
    {
        [Key]
        public string? GenderValue { get; set; } public string?
        GenderName { get; set; }
    }
}

```

OnlineVotingDB.CS

```

using Microsoft.EntityFrameworkCore;using
System.Diagnostics.Metrics; using
Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Models
{
    public class OnlineVotingDB:DbContext
    {
        public OnlineVotingDB(DbContextOptions<OnlineVotingDB> options) :base(options)
        {

        }

        public DbSet<Users> Users { get; set; } public
        DbSet<Gen> Gen { get; set; }
        public DbSet<Candidate> Candidate { get; set; }

        public DbSet<Election> Election { get; set; }
        public DbSet<ElectionVotes> ElectionVotes { get; set; } public DbSet<ElectionResult>
        ElectionResult { get; set; }
        public DbSet<Onlinevotingsystem.Models.UserLogin> UserLogin { get;
set; }
        public DbSet<Onlinevotingsystem.Models.Reply> Reply { get; set; } public
        DbSet<Onlinevotingsystem.Models.Winner> Winner { get; set; } public
        DbSet<Onlinevotingsystem.Models.ForgotPass> ForgotPass {
get; set; }
    }
}

```

Notification.CS

```

using System.ComponentModel.DataAnnotations;namespace
Onlinevotingsystem.Models
{
    public class Reply
    {
        [Key]
        public int Userid { get; set; } public bool? status {
get; set; }

        public string? color { get; set; } public string? message {
get; set; } public string? passcode { get; set; }
    }
}

```

}

UserLogin.CS

```
using System.ComponentModel.DataAnnotations;

namespace Onlinevotingsystem.Models
{
    public class UserLogin
    {
        [Key]
        [Required(ErrorMessage = "Please enter email !")] [Display(Name =
        "enter email ")]
        public string? email { get; set; }

        [Required(ErrorMessage = "Please enter Password !")] [Display(Name =
        "Password ")] [DataType(DataType.Password)]
        public string? password { get; set; }
    }
}
```

Users.CS

```
using System.ComponentModel.DataAnnotations;using
System.Xml.Linq;

namespace Onlinevotingsystem.Models
{
    public class Users
    {
        [Key]
        public int Userid { get; set; }

        [Required(ErrorMessage = "Please enter Firstname !")] [Display(Name =
        "FirstName ")]
        public string? FirstName { get; set; }

        [Required(ErrorMessage = "Please enter Lastname !")] [Display(Name =
        "LastName ")]
        public string? LastName { get; set; }

        [Required(ErrorMessage = "Please pick birthdate")] [Display(Name =
        "DateOfBirth ")]
        public DateTime DateOfBirth { get; set; }

        [Required(ErrorMessage = "Please select gender !")] [Display(Name =
        "Gender ")]
        public char Gender { get; set; }
    }
}
```

```

        [Required(ErrorMessage = "Please enter email !")][Display(Name =
        "Email ")]
        [RegularExpression("[a-zA-Z0-9_\\.-]+@([a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$", ErrorMessage =
"E-mail id is not valid")]
        public string? email { get; set; }

        [Required(ErrorMessage = "Please enter mobile no !")][Display(Name =
        "PhoneNo ")]
        [RegularExpression(@"^[6-9]{1}[0-9]{9}$", ErrorMessage = "InvalidPhone Number")]
        public string? PhoneNo { get; set; }

        [Required(ErrorMessage = "Please enter address !")][Display(Name =
        "address ")]
        public string? address { get; set; } [Display(Name =
        "Eligibility to vote ")]public string? CanVote { get; set; }

    }
}

```

Winner.CS

```

using System.ComponentModel.DataAnnotations;

namespace Onlinevotingsystem.Models
{
    public class Winner
    {
        [Key]
        public int Id { get; set; }
        public string CandidateName { get; set; }public string
        CandidateImg { get; set; } public string ElectionName {
        get; set; } public int NoofVotes { get; set; }

    }
}

```

CandidateController.CS

```

using System;

```



```

using System.Collections.Generic; using
System.Diagnostics.Metrics;using System.Linq;
using System.Threading.Tasks; using
Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;using
Microsoft.EntityFrameworkCore; using
NuGet.Protocol.Plugins;
using Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Controllers
{
    public class CandidateController : Controller
    {
        private readonly OnlineVotingDB _context;

        public CandidateController(OnlineVotingDB context)
        {
            _context = context;
        }

        // GET: Candidate
        public async Task<IActionResult> Index()
        {
            TempData["nop"] = null;
            var x = HttpContext.Session.GetString("email"); var y =
            HttpContext.Session.GetString("password");if (x == null || y == null)
            {
                TempData["nop"] = "Please Login To Access this
Functionality!";
                return RedirectToAction("Login", "Users", new { area = ""
});
            }

            if (x != "admin" && y != "admin")
            {
                TempData["nop"] = "Please Login with admin credentials toview whole
Candidate List!";
                return RedirectToAction("Login", "Users", new { area = ""
});
            }
            return View(await _context.Candidate.ToListAsync());
        }

        // GET: Candidate/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Candidate == null)
            {
                return NotFound();
            }

            var candidate = await _context.Candidate
                .FirstOrDefaultAsync(m => m.Candidateid == id);if (candidate ==
            null)
            {
                return NotFound();
            }

            return View(candidate);
        }
    }
}

```

```

// GET: Candidate/Create public
ActionResult Create()
{
    var x = HttpContext.Session.GetString("email"); var y =
    HttpContext.Session.GetString("password");if (x == null || y == null)
    {
        TempData["nop"] = "Please Login To Access this
Functionality!";
        return RedirectToAction("Login", "Users", new { area = ""
    });
    }

    if (x != "admin" && y != "admin")
    {
        TempData["nop"] = "Please Login with admin credentials toCreate Candidate !";
        return RedirectToAction("Login", "Users", new { area = ""
    });
    }
    List<Election> electionList = new List<Election>(); electionList = (from c in
    _context.Election select c).ToList();electionList.Insert(0, new Election { Electionid = 0,
ElectionName = " --Select Election-- " }); ViewBag.Election =
    electionList;return View();
}

// POST: Candidate/Create
// To protect from overposting attacks, enable the specificproperties you want to
bind to.
// For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost] [ValidateAntiForgeryToken]
public async Task<ActionResult>
Create([Bind("Candidateid,TypeElection,FirstName,LastName,DateOfBirth,Gende r,email,password,Image,PhoneNo,address")])
Candidate candidate)
{
    var can = await _context.Candidate.Where(h => h.TypeElection ==
candidate.TypeElection).ToListAsync();
    var elec = await _context.Election.Where(h => h.Electionid ==
candidate.TypeElection).FirstOrDefaultAsync();
    var today = DateTime.Today;if
    (ModelState.IsValid)
    {
        if (elec.EndDate > today.Date) {if
            (can.Count() < 4)
            {
                _context.Add(candidate);
                await _context.SaveChangesAsync; TempData["suc"] = "Candidate
                Added sucessfully!";return RedirectToAction(nameof(Create));
            }
            var v = ViewData["nospace"];
            ViewBag.seats = v;
            TempData["cf"] = "There is no seats for Candidate toreturn
get registered";
            RedirectToAction(nameof(Create));
        }
    }
}

```

```

        TempData["cf"] = "Your selected Election has alreadyreturn
Expired!";
        RedirectToAction(nameof(Create));
    }

    return View(candidate);
}

// GET: Candidate/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Candidate == null)
    {
        return NotFound();
    }

    var candidate = await _context.Candidate.FindAsync(id); if (candidate == null)
    {
        return NotFound();
    }
    return View(candidate);
}

// POST: Candidate/Edit/5
// To protect from overposting attacks, enable the specific properties you want to
bind to.
// For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("Candidateid,TypeElection,FirstName,LastName,DateOfBirth,Gender,email
,password,Image,PhoneNo,address")] Candidate candidate)
{
    if (id != candidate.Candidateid)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(candidate);
            await _context.SaveChangesAsync();
        }

```

```

        catch (DbUpdateConcurrencyException)
        {
            if (!CandidateExists(candidate.Candidateid))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(candidate);
}

// GET: Candidate/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Candidate == null)
    {
        return NotFound();
    }

    var candidate = await _context.Candidate
        .FirstOrDefaultAsync(m => m.Candidateid == id); if (candidate ==
        null)
    {
        return NotFound();
    }

    return View(candidate);
}

// POST: Candidate/Delete/5 [HttpPost,
ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Candidate == null)
    {
        return Problem("Entity set 'OnlineVotingDB.Candidate' is
null.");
    }
}

```

```

        var candidate = await _context.Candidate.FindAsync(id); if (candidate != null)
        {
            _context.Candidate.Remove(candidate);
        }

        await _context.SaveChangesAsync(); return
        RedirectToAction(nameof(Index));
    }

    private bool CandidateExists(int id)
    {
        return _context.Candidate.Any(e => e.Candidateid == id);
    }
}

```

ElectionController.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks; using
Microsoft.AspNetCore.Http; using
Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using
NuGet.Protocol.Plugins;
using Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Controllers
{
    public class ElectionController : Controller
    {
        private readonly OnlineVotingDB _context;

        public ElectionController(OnlineVotingDB context)
        {
            _context = context;
        }

        // GET: Election
        public async Task<IActionResult> Index()
        {
            TempData["nop"] = null;
            var x = HttpContext.Session.GetString("email"); var y =
            HttpContext.Session.GetString("password"); if (x == null || y == null)
            {
                return RedirectToAction("Index");
            }

            return View();
        }
    }
}

Functionality!";

});

```

```

TempData["nop"] = "Please Login
To Perform this          return RedirectToAction("Login", "Users", new { area = ""

        if (x != "admin" && y != "admin")
        {
            TempData["nop"] = "You dont Have Permission to access thisreturn
Functionality!";
            RedirectToAction("Login", "Users", new { area = ""
        });
    }
}

```

```

        return View(await _context.Election.ToListAsync());
    }
    public async Task<IActionResult> UserElection()
    {
        var Elections = await _context.Election.ToListAsync();var today =
        DateTime.Today;
        List<Election> elec = new List<Election>();foreach (var e in
        Elections)
        {
            if (today <= e.EndDate)
            {
                elec.Add(e);
            }
        }
        return View(elec);
    }
    public async Task<IActionResult> VuScore(int? id)
    {
        var electionResults =
        _context.ElectionResult.Where(s=>s.ElectionId==id).ToList(); int numberOfVotes=0 ;
    }
}

```

```

int winnerCandidateID = 0;
Winner winnerCandidate= new Winner();

int count = 0;
foreach (var x in electionResults) {if (count == 0) {
    numberOfVotes = x.votes; winnerCandidateID =
    x.CandidateId;
}
if (count > 0)
{
    if(numberOfVotes < x.votes)
    {
        numberOfVotes = x.votes; winnerCandidateID =
        x.CandidateId;
    }
}
count++;
}
//get Election

var election = await _context.Election
    .FirstOrDefaultAsync(m => m.Electionid == id);

var today = DateTime.Today;
//get winner Candidate

var candidate = await _context.Candidate
    .FirstOrDefaultAsync(m => m.Candidateid ==
winnerCandidateID);
if (election.EndDate>today.Date)
{
    winnerCandidate.CandidateName = "Nobody"; winnerCandidate.CandidateImg
    = "Wait.jfif"; winnerCandidate.NoofVotes = 0; winnerCandidate.ElectionName
    = election.ElectionName; TempData["wait"] = "...Votings Are Still Going On...";
    return View(winnerCandidate);
}
winnerCandidate.CandidateName = candidate.FirstName + "" +
candidate.LastName;
winnerCandidate.CandidateImg = candidate.Image; winnerCandidate.NoofVotes =
numberOfVotes; winnerCandidate.ElectionName = election.ElectionName;

return View(winnerCandidate);
}

public async Task<IActionResult> VuCandidate( int? id)
{
    var Elections= _context.Election.ToList();

    if (      Elections == null)
    {
        return NotFound();
    }
    Election elect = null; foreach(var x in
    Elections)
    {
        if (x.Electionid == id)

```

```

        {
            elect = x;
        }
    }
    // var election = _context.Election.FirstOrDefault(m => m.Electionid == id);
    if (elect == null)
    {
        return NotFound();
    }
    // ViewBag.xemail = HttpContext.Session.GetString("email");
    HttpContext.Session.SetString("elecname", "" + elect.Electionid); ViewBag.yname = id;
    var candidates = _context.Candidate.Where(m => m.TypeElection
==id);
        // .SingleOrDefault(m => m.TypeElection == id);
        // if (candidates != null)
        // {
        return View(candidates);
        // return Json(new { status = true, message = "Vote Foreligible candidate
you think ", CandidateList = candidates });
        // }
        // return Json(new { status = false, message = "No CandidatesRegistered " });
        // return RedirectToAction(nameof(Index));
    }
    // GET: Election/Details/5
    public async Task<IActionResult> CastVotes(int? id)
    {
        Reply r = new Reply();
        ElectionResult er = new ElectionResult();
        var votes = await _context.ElectionVotes.ToListAsync(); ElectionVotes ev =
        new ElectionVotes();
        var x = HttpContext.Session.GetString("email"); var y =
        HttpContext.Session.GetString("elecname"); var z =
        HttpContext.Session.GetString("password");
        var election = await _context.Election.FirstOrDefault(m => m.Electionid ==
Int16.Parse(y));
        var today = DateTime.Today; var usr =
        await _context.Users
        .FirstOrDefaultAsync(m => m.email == x && m.password == z); if (usr != null && y
        != null && id != null)
        {
            ev.UserId = usr.UserId; ev.CandidateId = (int)id;
            ev.ElectionId = Int16.Parse(y);
            var loggedin = _context.ElectionResult.Where(s => s.ElectionId ==
ev.ElectionId && s.CandidateId == ev.CandidateId).SingleOrDefault();
            var alreadyVoted = false; if
            (usr.CanVote == "true")
            {
                foreach (var i in votes)
                {
                    if (i.UserId == ev.UserId &&
i.ElectionId == ev.ElectionId)
                    {
                        alreadyVoted = true;

```



```

    }
}
if(!alreadyVoted)
{
    if (election.EndDate > today.Date) {
        _context.Add(ev);
        await _context.SaveChangesAsync();if (loggedin
        == null)
        {
            er.ElectionId = Int16.Parse(y);
            er.CandidateId = (int)id; er.votes = 1;
            _context.Add(er);
            await _context.SaveChangesAsync();

        }
        else
        {
            int i = loggedin.votes;
            er.ElectionResultId =

loggedin.ElectionResultId;

            er.ElectionId = Int16.Parse(y);
            er.CandidateId = (int)id; er.votes = i + 1;
            _context.Update(er);
            await _context.SaveChangesAsync();

        }

        r.status = true;
        r.color = "alert alert-success"; r.message = "Your vote
        have been casted!";return View(r);
    }
    r.status = false;
    r.color = "alert alert-danger"; r.message =
    "..Election is Over..";
    // return Json(new { status = false, message = "Your vote have
    been already casted!" });

    return View(r);

}
r.status = false;
r.color = "alert alert-danger";
r.message = "Your vote have been already casted!";
// return Json(new { status = false, message = "Yourvote have been
already casted!" });

return View(r);
}
r.status = false;
r.color = "alert alert-warning";
r.message = "You dont have permission to vote!";
//return Json(new { status = false, message = "You donthave permission to
vote!" });

return View(r);
}

r.status = false;
r.color = "alert alert-info";

```

```

        r.message = "Sorry for the inconvenience please check after
sometime";    // return Json(new { status = false, message = "You dont have
permission to vote!" });
        return View(r);
    }
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null || _context.Election == null)
        {
            return NotFound();
        }

        var election = await _context.Election
            .FirstOrDefaultAsync(m => m.Electionid == id); if (election ==
            null)
        {
            return NotFound();
        }

        return View(election);
    }

    // GET: Election/Create public
    IActionResult Create()
    {
        return View();
    }

    // POST: Election/Create
    // To protect from overposting attacks, enable the specific properties you want to
bind to.
    // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost] [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("Electionid,ElectionName,EndDate")] Election election)
    {
        TempData["eles"] = null; if
        (ModelState.IsValid)
        {
            _context.Add(election);
            await _context.SaveChangesAsync();
            TempData["eles"] = "..Election Created Successfully.."; return
            RedirectToAction(nameof(Create));
        }
        TempData["elef"] = "..Election Creation Failed.."; return View(election);
    }

    // GET: Election/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Election == null)
        {
            return NotFound();
        }

        var election = await _context.Election.FindAsync(id); if (election == null)

```

```

        {
            return NotFound();
        }
        return View(election);
    }

    // POST: Election/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to
bind to.
    // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id, [Bind("Electionid,ElectionName,EndDate")] Election election)
    {
        if (id != election.Electionid)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(election);
                await _context.SaveChangesAsync();
            }

            catch (DbUpdateConcurrencyException)
            {
                if (!ElectionExists(election.Electionid))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(election);
    }

    // GET: Election/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null || _context.Election == null)
        {
            return NotFound();
        }

        var election = await _context.Election
            .FirstOrDefaultAsync(m => m.Electionid == id); if (election ==
null)
        {
            return NotFound();
        }

        return View(election);
    }

```

```

// POST: Election/Delete/5 [HttpPost,
ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Election == null)
    {
        return Problem("Entity set 'OnlineVotingDB.Election' is
null.");
    }
    var election = await _context.Election.FindAsync(id); if (election != null)
    {
        _context.Election.Remove(election);
    }

    await _context.SaveChangesAsync(); return
    RedirectToAction(nameof(Index));
}

private bool ElectionExists(int id)
{
    return _context.Election.Any(e => e.Electionid == id);
}
}
}

```

ElectionVotesController.CS

```

using System;
using System.Collections.Generic; using
System.Linq;
using System.Threading.Tasks; using
Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering; using
Microsoft.EntityFrameworkCore; using
Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Controllers
{
    public class ElectionVotesController : Controller
    {
        private readonly OnlineVotingDB _context;

        public ElectionVotesController(OnlineVotingDB context)
        {
            _context = context;
        }

        // GET: ElectionVotes
        public async Task<IActionResult> Index()
        {
            return View(await _context.ElectionVotes.ToListAsync());
        }
    }
}

```

```

// GET: ElectionVotes/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null || _context.ElectionVotes == null)
    {
        return NotFound();
    }

    var electionVotes = await _context.ElectionVotes
        .FirstOrDefaultAsync(m => m.ElectionVoteId == id); if (electionVotes
        == null)
    {
        return NotFound();
    }

    return View(electionVotes);
}

// GET: ElectionVotes/Createpublic
IActionResult Create()
{
    return View();
}

// POST: ElectionVotes/Create
// To protect from overposting attacks, enable the specific properties you want to
bind to.
// For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost] [ValidateAntiForgeryToken]
public async Task<IActionResult>
Create([Bind("ElectionVoteId,UserId,ElectionId,CandidateId")] ElectionVotes electionVotes)
{
    if (ModelState.IsValid)
    {
        _context.Add(electionVotes);
        await _context.SaveChangesAsync; return
        RedirectToAction(nameof(Index));
    }
    return View(electionVotes);
}

// GET: ElectionVotes/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.ElectionVotes == null)
    {
        return NotFound();
    }

    var electionVotes = await _context.ElectionVotes.FindAsync(id); if (electionVotes == null)
    {
        return NotFound();
    }
    return View(electionVotes);
}

```

```

        // POST: ElectionVotes/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to
bind to.
        // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("ElectionVoteId,UserId,ElectionId,CandidateId")] ElectionVotes electionVotes)
        {
            if (id != electionVotes.ElectionVoteId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(electionVotes); await
                    _context.SaveChangesAsync();
                }

                catch (DbUpdateConcurrencyException)
                {
                    if (!ElectionVotesExists(electionVotes.ElectionVoteId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(electionVotes);
        }

        // GET: ElectionVotes/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.ElectionVotes == null)
            {
                return NotFound();
            }

            var electionVotes = await _context.ElectionVotes
                .FirstOrDefaultAsync(m => m.ElectionVoteId == id); if (electionVotes
                == null)
            {
                return NotFound();
            }

            return View(electionVotes);
        }

        // POST: ElectionVotes/Delete/5 [HttpPost,
        ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)

```

```

        {
            if (_context.ElectionVotes == null)
            {
                return Problem("Entity set 'OnlineVotingDB.ElectionVotes'
is null.");
            }
            var electionVotes = await _context.ElectionVotes.FindAsync(id); if (electionVotes != null)
            {
                _context.ElectionVotes.Remove(electionVotes);
            }

            await _context.SaveChangesAsync(); return
            RedirectToAction(nameof(Index));
        }

        private bool ElectionVotesExists(int id)
        {
            return _context.ElectionVotes.Any(e => e.ElectionVoteId == id);
        }
    }
}

```

HomeController.CS

```

using Microsoft.AspNetCore.Mvc; using
NuGet.Protocol.Plugins; using
Onlinevotingsystem.Models; using
System.Diagnostics;

namespace Onlinevotingsystem.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            var x = HttpContext.Session.GetString("email"); var y =
            HttpContext.Session.GetString("password");

            if (x != "admin" && y != "admin")
            {
                return RedirectToAction("Login", "Users", new { area = ""
});
            }
            else if (x == "admin" && y == "admin")
            {
                return View();
            }
            return RedirectToAction("Homepage", "Users", new { area = ""
});
        }
    }
}

```

```

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}

```

UserController.CS

```

using System;
using System.Collections.Generic;using
System.Linq;
using System.Runtime.InteropServices;using
System.Text;
using System.Threading.Tasks; using
Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;using
Microsoft.EntityFrameworkCore; using
NuGet.Protocol.Plugins;
using Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Controllers
{
    public class UserController : Controller
    {
        private readonly OnlineVotingDB _context;const string
        chars =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"; public
        UserController(OnlineVotingDB context)
        {
            _context = context;
        }

        // GET: User
        public async Task<IActionResult> Index()
        {
            return View(await _context.Users.ToListAsync());
        }

        // GET: User/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Users == null)
            {
                return NotFound();
            }

            var users = await _context.Users
                .FirstOrDefaultAsync(m => m.Userid == id);if (users ==
                null)

```



```

        {
            return NotFound();
        }

        return View(users);
    }

    // GET: User/Create
    public IActionResult Create()
    {
        List<Gen> glist = new List<Gen>(); glist =
        _context.Gen.ToList();
        glist.Insert(0, new Gen { GenderValue = "x", GenderName = "--Select Gender-- " });

        ViewBag.Gender = glist;

        return View();
    }

    // POST: User/Create
    // To protect from overposting attacks, enable the specific properties you want to
bind to.
    // For more details, see
    http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost] [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("FirstName,LastName,DateOfBirth,Gender,email,PhoneNo,address")]
Users users)
    {
        TempData["regf"] = null;
        TempData["regs"] = null; if (users
!=null)
        {
            StringBuilder sb = new StringBuilder(); Random rnd =
            new Random();

            for (int i = 0; i < 10; i++)
            {
                int index = rnd.Next(chars.Length);
                sb.Append(chars[index]);
            }
            users.CanVote = "false";
            var password = sb.ToString();
            users.password = password;
            if (!UsersEmailExists(users.email)) {
                _context.Add(users);
                await _context.SaveChangesAsync(); TempData["regs"] =
                "Registration Success!! Your
Password -->" + users.password;
                return RedirectToAction(nameof(Create));
            }
            TempData["regf"] = "Entered Email already Exists!!"; return
            RedirectToAction(nameof(Create));
        }
        TempData["regf"] = "Registration Failed!!";
    }

```

```

        return View(users);
    }

    // GET: User/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Users == null)
        {
            return NotFound();
        }

        var users = await _context.Users.FindAsync(id); if (users == null)
        {
            return NotFound();
        }
        return View(users);
    }

    // POST: User/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to
bind to.
    // For more details, see
    http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id,
[Bind("Userid,FirstName,LastName,DateOfBirth,Gender,email,password,PhoneNo, address,CanVote")] Users users)
    {
        if (id != users.Userid)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(users);
                await _context.SaveChangesAsync();
            }

            catch (DbUpdateConcurrencyException)
            {
                if (!UsersExists(users.Userid))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(users);
    }

    // GET: User/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {

```

```

        if (id == null || _context.Users == null)
        {
            return NotFound();
        }

        var users = await _context.Users
            .FirstOrDefaultAsync(m => m.Userid == id); if (users ==
            null)
        {
            return NotFound();
        }

        return View(users);
    }

    // POST: User/Delete/5 [HttpPost,
    ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.Users == null)
        {
            return Problem("Entity set 'OnlineVotingDB.Users' is
null.");
        }
        var users = await _context.Users.FindAsync(id); if (users != null)
        {
            _context.Users.Remove(users);
        }

        await _context.SaveChangesAsync(); return
        RedirectToAction(nameof(Index));
    }

    private bool UsersExists(int id)
    {
        return _context.Users.Any(e => e.Userid == id);
    }
    private bool UsersEmailExists(string email)
    {
        return _context.Users.Any(e => e.email == email);
    }
}

```

UsersController.CS

```
using System;
using System.Collections.Generic using System.Linq;
using System.Runtime.InteropServices;using
System.Text;
using System.Threading.Tasks; using
Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;using
Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;using
Microsoft.EntityFrameworkCore; using
NuGet.Protocol.Plugins;
using Onlinevotingsystem.Models;

namespace Onlinevotingsystem.Controllers
{
    public class UsersController : Controller
    {
        private readonly OnlineVotingDB _context;const string
        chars =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"; public
        UsersController(OnlineVotingDB context)
        {
            _context = context;
        }

        // List<Users> regusers = _context.Users.ToListAsync();
        // ViewBag.regusers=regusers;
        // GET: Users
        public async Task<IActionResult> Index()
        {
            var x = HttpContext.Session.GetString("email"); var y =
            HttpContext.Session.GetString("password");if (x == null || y == null)
            {
                TempData["nop"] = "Please Login To Perform thisreturn
Functionality!";
                RedirectToAction(nameof(Login));
            }

            if (x != "admin" && y != "admin")
            {
                TempData["nop"] = "You dont Have Permission to access thisreturn RedirectToAction(nameof(Login));
            }

            var uslist = _context.Users.ToList();
            return View(await _context.Users.ToListAsync());
        }
    }
}
```

```

        public IActionResult ForgotPassword()
        {
            //var user = await _context.Users
            //    .FirstOrDefaultAsync(m => m.email.Equals(email) &&
            m.PhoneNo.Equals(phoneNo));
            return View();
        }

        public ActionResult UpPassword(string id,string id2,stringid3,string?
        email,string? phoneNo,string? newPassword)
        {
            Reply r = new Reply();

            var usrList = _context.Users.ToList();
            if (usrList.Count() > 0)
            {
                foreach (var s in usrList)
                {
                    if (s.email.Equals(email) && s.PhoneNo.Equals(phoneNo))
                    {
                        s.password =newPassword;
                        _context.Update(s);
                        _context.SaveChangesAsync();

                        r.status = true;
                        r.color = "alert alert-success";
                        r.message = "Your Password have been updatedreturn
Sucessfully!";
                        View(r);
                    }
                }
            }

            r.status = false;
            r.color = "alert alert-danger";
            r.message = "Your Password update Failed!";

            return View(r);
        }

        // GET: Users/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            var x = HttpContext.Session.GetString("email"); var y =
            HttpContext.Session.GetString("password");if (x == null || y == null)
            {
                TempData["nop"] = "Please Login To Perform thisreturn
Functionality!";
                RedirectToAction(nameof(Login));
            }

            if (x != "admin" && y != "admin")
            {
                TempData["nop"] = "You dont Have Permission to access thisreturn
Functionality!";
                RedirectToAction(nameof(Login));
            }
        }

```

```

        if (id == null || _context.Users == null)
        {
            return NotFound();
        }

        var users = await _context.Users
            .FirstOrDefaultAsync(m => m.Userid == id); if (users ==
            null)
        {
            return NotFound();
        }

        return View(users);
    }

    // GET: Users/Create
    public IActionResult Create()
    {
        return View();
    }

    // POST: Users/Create

    // To protect from overposting attacks, enable the specific properties you want to
bind to.
    // For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.

    [HttpPost]
    // [ValidateAntiForgeryToken]
    public ActionResult Createu(Users users)
    {
        Reply r = new Reply(); if (users
        != null)
        {
            StringBuilder sb = new StringBuilder(); Random rnd =
            new Random();

            for (int i = 0; i < 10; i++)
            {
                int index = rnd.Next(chars.Length);
                sb.Append(chars[index]);
            }
            users.CanVote = "false";
            var password = sb.ToString();
            users.password = password;

            _context.Add(users);
            _context.SaveChangesAsync();
            var usrpassword = "This is your password : "+
users.password;

            r.status = true;
            r.color = "alert alert-success"; r.message = " Registration
            Sucessful! "; r.passcode = usrpassword;
            return Json(new { status = true, message = "Registration
            Sucessful! ", pass = usrpassword });
            //return View(r);
        }
    }

```

```

        // return View(r);
        return Json(new { status = false, message = "Registration
Failed!"));
    }

    public ActionResult Login()
    {
        HttpContext.Session.Remove("password");
        HttpContext.Session.Remove("email"); return View();
    }

    public ActionResult UsersLogin(UserLogin login)
    {
        if (login == null)
        {
            TempData["logf"] = "...!! Please Enter credentials !!..";return Json(new { status =
false, message = "email and
Password Not Match!" });
        }

        var loggedin = _context.Users.Where(s => s.email.Equals(login.email) && s.password ==
login.password).Count();
        var users = _context.Users.Where(s => s.email.Equals(login.email) && s.password ==
login.password);
        ViewBag.regusers = users;

        // if (users.Any())
        if (loggedin == 1)
        {
            // var loggedin = _context.Users.Where(s =>
s.email.Equals(login.email) && s.password == login.password).SingleOrDefault();
            //Homepage();
            HttpContext.Session.SetString("email", login.email);
            HttpContext.Session.SetString("password",
login.password);

            TempData["logs"] = "...!!Login Successful!!..";return Json(new {
status = true, message = "Login
Sucessful!" });
        }
        else
        {
            TempData["logf"] = "...!! Entered Credentials dont
match!!..";
            return Json(new { status = false, message = "email and
Password Not Match!" });
        }
        //}
        // else
        // {
        //     TempData["logf"] = "...!!User Dont Exists!!..";
        //     return Json(new { status = false, message = "UserNotFound!" });
        // }

        //}
    }
    public ActionResult Homepage()
    {
        var x = HttpContext.Session.GetString("email"); var y =
HttpContext.Session.GetString("password");if (x == null || y == null)
        {

```

```

        TempData["nop"] = "Please Login To Access Home Page!";return
        RedirectToAction(nameof(Login));
    }
    return View();
}

// GET: Users/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    TempData["nop"] = null;
    TempData["Vps"] = null;
    TempData["Vpf"] = null;
    var x = HttpContext.Session.GetString("email"); var y =
    HttpContext.Session.GetString("password");if (x == null || y == null)
    {
        TempData["nop"] = "Please Login To Perform this
Functionality!";
        return RedirectToAction(nameof(Login));
    }

    if (x != "admin" && y != "admin")
    {
        TempData["nop"] = "You dont Have Permission to access thisreturn
Functionality!";
        RedirectToAction(nameof(Login));
    }

    if (id == null || _context.Users == null)
    {
        return NotFound();
    }

    var users = await _context.Users.FindAsync(id);if (users == null)
    {
        return NotFound();
    }
    return View(users);
}

// POST: Users/Edit/5
// To protect from overposting attacks, enable the specificproperties you want to
bind to.
// For more details, see
http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("Userid,FirstName,LastName,DateOfBirth,Gender,email,password,PhoneNo, address,CanVote")] Users users)
{
    TempData["Vps"] = null;
    TempData["Vpf"] = null;
    var usr = await _context.Users.FindAsync(id);users.Userid = id;
    users.FirstName = usr.FirstName; users.LastName
    = usr.LastName; users.DateOfBirth =
    usr.DateOfBirth;users.email = usr.email;
    users.password = usr.password; users.PhoneNo =
    usr.PhoneNo; users.Gender = usr.Gender;
    users.address = usr.address;

```



```

        if (id != users.Userid)
        {
            return NotFound();
        }

        if (users != null)
        {
            try
            {
                _context.Update(users);
                await _context.SaveChangesAsync();
                if (users.CanVote == "false")
                {
                    TempData["Vpf"] = "Permission to Vote is Deprecated  
to " + users.FirstName + " " + users.LastName + " user!";
                }
                else
                {
                    TempData["Vps"] = "Permission to Vote is Granted to " +  
users.FirstName + " " + users.LastName + " user!";
                }
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!UsersExists(users.Userid))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return RedirectToAction(nameof(Index));
        }
        return View(users);
    }

    // GET: Users/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        var x = HttpContext.Session.GetString("email"); var y =  
HttpContext.Session.GetString("password"); if (x == null || y == null) {  
    TempData["nop"] = "Please Login To Perform this  
Functionality!";
        return RedirectToAction(nameof(Login));
    }

    if (x != "admin" && y != "admin")
    {
        Delete!";
    }
}

```

```

TempData["nop"] = "You dont Have Permission to access thisreturn RedirectToAction(nameof(Login));

        if (id == null || _context.Users == null)
        {
            return NotFound();
        }

        var users = await _context.Users
            .FirstOrDefaultAsync(m => m.Userid == id);var elecs =
            _context.ElectionVotes.ToList(); foreach(var e in elecs)
        {
            if(e.UserId == id)
            {
                TempData["alv"] = users.FirstName + " " + users.LastName + " have Already
Voted for election now u cannot delete him
!..";

                return RedirectToAction(nameof(Index));
            }
        }

        return View(users);
    }

    // POST: Users/Delete/5 [HttpPost,
    ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> DeleteConfirmed(int id)
    {
        if (_context.Users == null)
        {
            return Problem("Entity set 'OnlineVotingDB.Users' is
null.");
        }
        var users = await _context.Users.FindAsync(id);

        if (users != null)
        {
            TempData["alvs"] = users.FirstName + " " + users.LastName + " have been deleted
successfully !..";
            _context.Users.Remove(users);
        }

        await _context.SaveChangesAsync(); return
        RedirectToAction(nameof(Index));
    }

    private bool UsersExists(int id)
    {
        return _context.Users.Any(e => e.Userid == id);
    }
}

```

ConnectionString.JSON

```
{
  "ConnectionStrings": {
    "Myconnection": "Data Source=.;Initial Catalog=OVSDDB;IntegratedSecurity=True;
Encrypt=False"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information", "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Program.CS

```
using Microsoft.EntityFrameworkCore;using
Onlinevotingsystem.Models; using
Microsoft.EntityFrameworkCore;
var builder = WebApplication.CreateBuilder(args);var
connectionString =
builder.Configuration.GetConnectionString("Myconnection");
// Add services to the container.
builder.Services.AddMvc(opt => opt.EnableEndpointRouting = false);
builder.Services.AddSession(options => {
    options.IdleTimeout = TimeSpan.FromMinutes(60);
});
builder.Services.AddControllersWithViews(); builder.Services.AddDbContext<OnlineVotingDB> (options =>
{
    options.UseSqlServer(connectionString); options.UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking);
});

var app = builder.Build();

// Configure the HTTP request pipeline.if
(!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this forproduction scenarios, see
https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
```

```

app.UseSession(); app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");app.Run();

```

COCOMO:

Cocomo (Constructive Cost Model) is a regression model based on LOC i.e. number of Lines of Code. The COCOMO in terms of effort (resources required to complete the project work) and schedule (time required to complete the project work) based on the size of the software product. It estimates the required number of Man-Months (MM) for the full development of software products. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations.

- Organic
- Semi detached
- Embedded

Mode	"a" variable	"b" variable	"c" variable	"d" variable	KLOC
Semi-Detached	3	1.12	2.5	0.35	3.128
Effort (person-months)		Duration (months)		Staffing	
10.76		5.74		1.87	

Staffing=effort/duration, effort=a*KLOC^b, KLOC=lines of code (thousands), duration=c*effort^d

estimates the cost for software product development

5.2 Testing Approaches:

Testing is the basic activity aimed at detecting and solving technical issues in the software source code and assessing the overall product usability, performance, security, and compatibility. A test approach is the test strategy implementation of a project which defines how testing would be carried out, defines the strategy which needs to be implemented and executed, to carry out a particular task. To make our software perform well it should be error-free. If testing is done successfully, it will remove all the errors from the software. It should be well-planned, defined, and documented.

There are different methods that can be used for software testing. Black box testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. White box testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

5.2.1 Unit Testing

Unit testing is a software development process in which smallest unit of programs are individually and independently tested to determine whether any issues occur. The objective of Unit Testing is to verify the functional correctness of each unit. A unit may be an individual function or procedure. The main benefit of Unit Testing is, it reduces defects in the early stage and also improves the design. It is important to test every units/module of the application to build the project correctly. In this project, we have tested each unit separately like Registration, Home page, Logout, etc. These parts are tested individually and independently. While testing each of these modules, it became easy to correct the mistakes and we have traced the errors easily. By doing this, we have learnt how to write a code while implementing it.

5.2.2 Integrated Testing

Integration testing is the process of testing the interface between two software units or module. It's focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plans to those aggregates, and delivers as its output the integrated system ready for system testing. We started are integration testing by verifying that after successful registration user is able to login with his/her login details and after successful login user is redirected to the voting page.

5.2.3 Beta Testing

Beta testing is one of the final steps in software development lifecycle (SDLC) before a product goes live. Also referred to as user testing or customer validation, beta testing aims to ensure that end users are satisfied with a software product before you make it generally available (GA).

While beta tests want to catch any software bugs and errors that have snuck through the testing process, it is more about understanding and improving the product's full end user experience before it becomes GA. That means thoroughly investigating the experience flow and understanding any pain points that will hinder enjoyment 54 of the experience for your end user. For this project, we had to test whether the voter details are getting stored into the database and after successful voting, voter is restricted to caste another vote.

5.2.4 System Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing developed system using various kinds of data. System is the stage of implementation that is aimed at assuring at the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests such as recover, security and usability tests. A series of testing is performed for the proposed system before the system is ready for the user acceptance testing. Implementation ends with formal tests. The test data are very crucial to this process. They must be realistic and cover extreme conditions as well.

Ideally, vary alternative path through the program should be exercised at least once beyond the test data. The system test must involve all the elements that compose the system including program validation checking, files, and forms and triggers procedures.

Test Cases

- Login Test Case

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Enter Valid Email Address and valid password	1.Enter Valid Email Address 2.Enter Valid Password. 3.Click Login	Email Id: Ram123@gmail.com Password: 123456	Redirected to User Home page	Redirected to Home page	Pass
2	Enter invalid Email Address And valid password	1.Enter Invalid EmailAddress 2.Enter Valid password 3.Click Login	Username: ram@gmail.com Password: 123456	The user won't be redirected to the user homepage.	The user was not redirected to the user home page.	Pass
3	Enter valid Email Address and invalid password	1.Enter Valid email Address 2.Enter Invalid password. 3.Click Login	Username: Ram123@gmail.com Password: 12345	The user won't be redirected to the user homepage.	The user was not redirected to the user home page.	Pass

4	Enter invalid Email Address and invalid password	1.Enter invalid EmailAddress 2.Enter invalid password 3.Click Login	Username: ram@gmail.com Password 12345	The user won't be redirected to the user homepage.	The user won't be redirected to the user home page.	Pass
---	--	---	---	--	---	------

• Registration Test Case:

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Enter all fields correctly and valid.	1. Enter FirstName 2. Enter LastName 3. Enter DOB 4. Enter Gender 5. Enter Email 6. Enter the Mobile number. 7. Enter Address 8. Click on register button	Ram Charan 07/07/02 Male Ram123@gmail.com 981959608 Mumbai	Signup page will be refreshed and the user details will be stored in the user table	Signup page was refreshed and the user details were stored in the user table	Pass

2	Enter all fields correctly expect email id	<ol style="list-style-type: none"> 1. Enter FirstName 2. Enter LastName 3. Enter DOB 4. Enter Gender 5. Enter Email 6. Enter theMobile number. 7. Enter Address 8. Click on register button 	<p>Ram</p> <p>Charan</p> <p>07/07/02</p> <p>Male</p> <p>Ramgmail.com</p> <p>981959608</p> <p>Mumbai</p>	Enter valid Email-Id	Enter valid Email-Id	Pass
3	Enter all fields keeping any one field empty	<ol style="list-style-type: none"> 1. Enter FirstName 2. Enter LastName 3. Enter DOB 4. Enter Gender 5. Enter Email 6. Enter theMobile number. 7. Enter Address 8. Click on register button 	<p>Charan</p> <p>07/07/02</p> <p>Male</p> <p>Ramgmail.com</p> <p>981959608</p> <p>Mumbai</p>	Please enter your firstname	Please enter your firstname	Pass

- Admin Test Case:

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Status
1	Enter Valid Email Address and valid password	1.Enter Valid Email Address 2.Enter Valid Password. 3.Click Login	Email Id: Ram123@gmail.com Password: 1234	Redirected to Admin Homepage	Redirected to Admin Home page	Pass
2	Enter invalid username And valid password	1.Enter Invalid EmailAddress 2.Enter Valid password 3.Click Login	Username: Ram07@gmail.com Password: 1234	The admin won't be redirected to the admin home page.	The admin was not redirected to the user home page.	Pass
3	Enter valid username and invalid password	1.Enter Valid email Address 2.Enter Invalid password. 3.Click Login	Username: Ram12@gmail.com Password: 12345	The admin won't be redirected to the admin home page.	The admin was not redirected to the user home page.	Pass
4	Enter invalid username and invalid password	1.Enter invalid email Address 2.Enter invalid password 3.Click Login	Username: Ram5@gmail.com Password 12345	The admin won't be redirected to the user homepage.	The admin was not redirected to the user home page.	Pass

- Add Candidate, Add Election test case

	Test Case Description	Test Case Description	Test Data	Expected result	Actual result	Status
1	Enter all field correctly and valid.in Candidate registration	Admin adds 1. Select Elections 2. FirstName 3. LastName 4. Enter DOB 5. Gender 6. Image 7. Mobile number. 8. Address 9. Click onCreate Candidate button	PM election Modi Ji 07/07/02 Male Modi.png 8459345726 Gujarat	Candidate registration page will be refreshedand the candidate details will bestored in the candidate table	Candidate page wasrefreshedand thecandidate Details were stored in the candidate table	Pass
2	Enter all field correctly and valid.in Candidate registration except firstname	Admin adds 1. Select Elections 2. FirstName 3. LastName 4. Enter DOB 5. Gender 6. Image 7. Mobile number. 8. Address 9. Click onCreate Candidate button	PM election Ji 07/07/02 Male Modi.png 8459345726 Gujarat	Please enter your first name	Please enter your firstname	Pass
3	Enter all field correctly and valid.in Candidate registration except Mobile Number	Admin adds 1. Select Elections 2. FirstName 3. LastName 4. Enter DOB 5. Gender 6. Image 7. Mobile number. 8. Address 9. Click onCreate Candidate button	PM election Modi Ji 07/07/02 Male Modi.png 84593457 Gujarat	Please enter a valid Mobile Number	Please enter a valid Mobile Number	Pass
4	Enter Election name and End Date	Admin adds 1. Election name 2. End date 3. Click on create	PM election 08/08/2024	Redirected to Admin Homepage	Was Redirected to Admin Homepage	Pass

5	User enters all fields correctly and valid	1. Email 2. Mobile no. 3. New password	Ram123@gmail.com 8722374873 12334564	Password will get updated and will be redirected to login page	Password was updated and redirected to login page	Pass
6	User Cast vote	1. Clicks on Cast Vote Button	-----	“Your vote has been casted successfully” message will be displayed	“Your vote has been casted successfully” message was displayed	Pass

Chapter 6

Results and Discussions

6.1 Test Reports

Test Report is a document that contains a summary of all test activities and the final test results of a testing project. A test report is an assessment of how well the Testing is performed. The main purpose of a test case is to ensure that different features within an application are working as expected. It helps to test the validation of the system. In the following module, the test cases are performed. Varying test cases were taken for each module. However, the number of test cases can be increased to test very simple and not-so-necessary conditions.

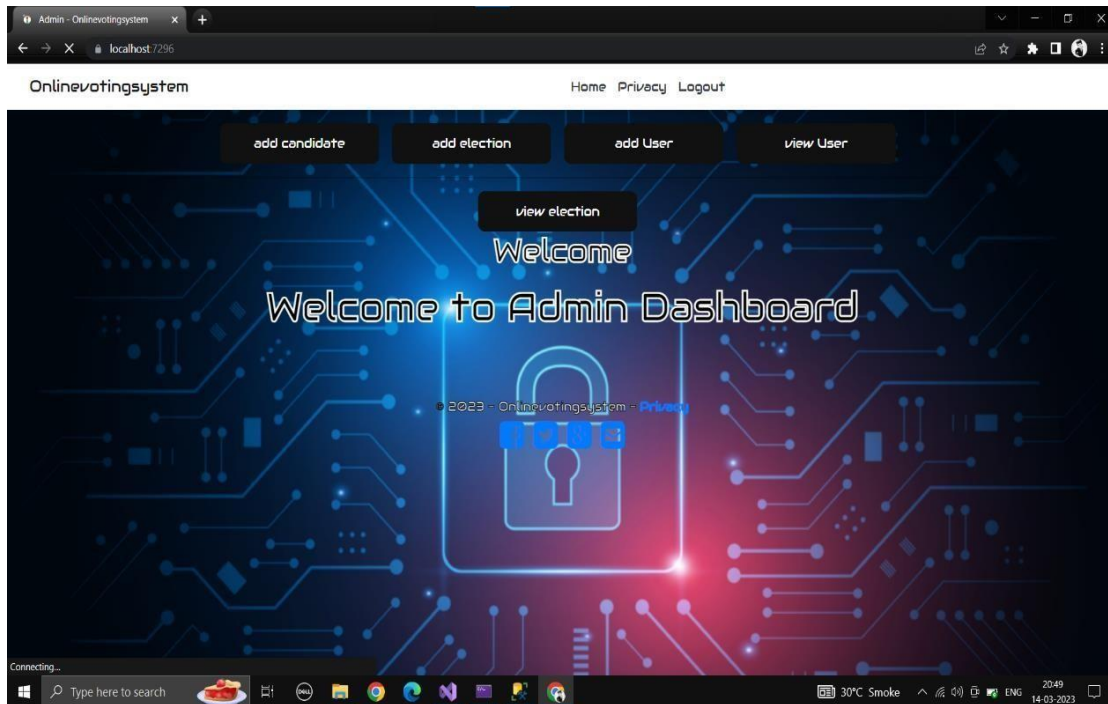
Some of the modules are:

- Home page module
- Registration module
- Login module
- User home module
- User details module
- Admin home module
- Add Candidate Module
- Create Election module
- Forgot Password
- Delete Election
- Election Details module
- Approval to vote module
- Privacy
- View Users module
- View Election module
- View Candidates module
- Election Result module

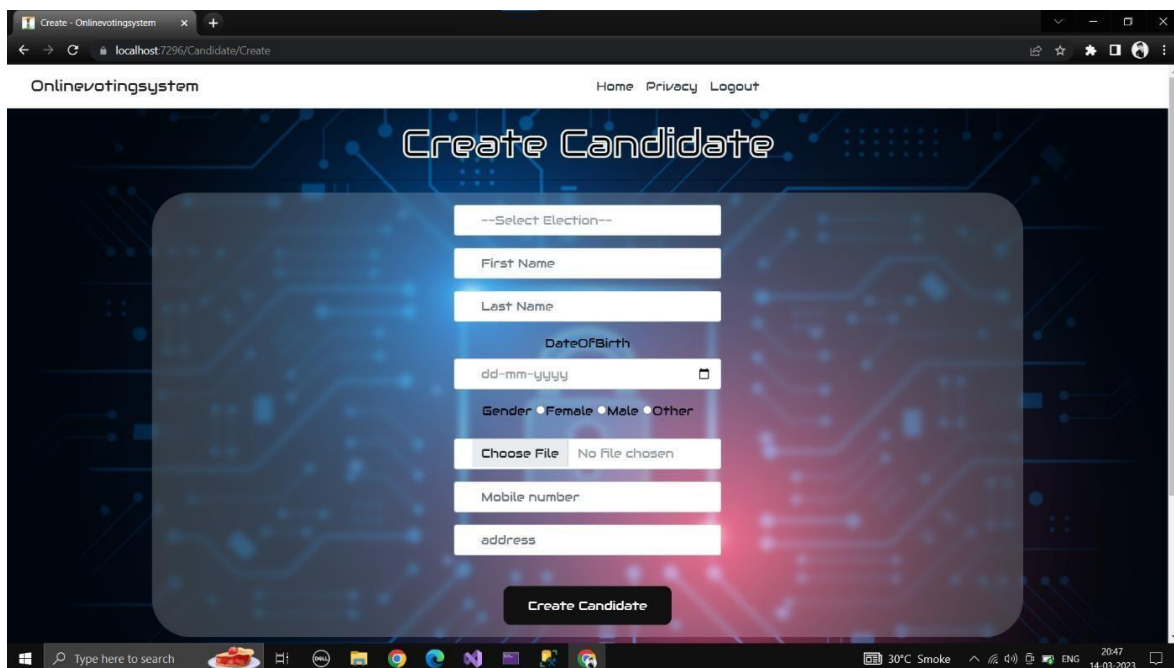
6.2 User Documentation

The main functions of this application are as follows:

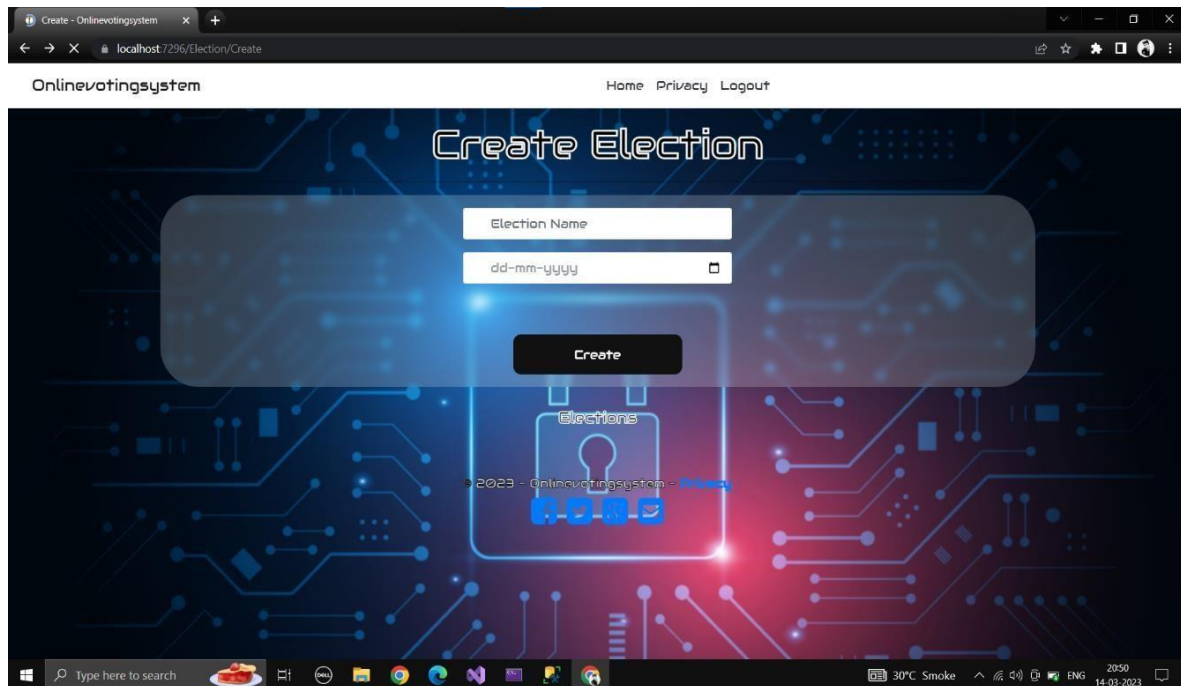
- **Admin Page**



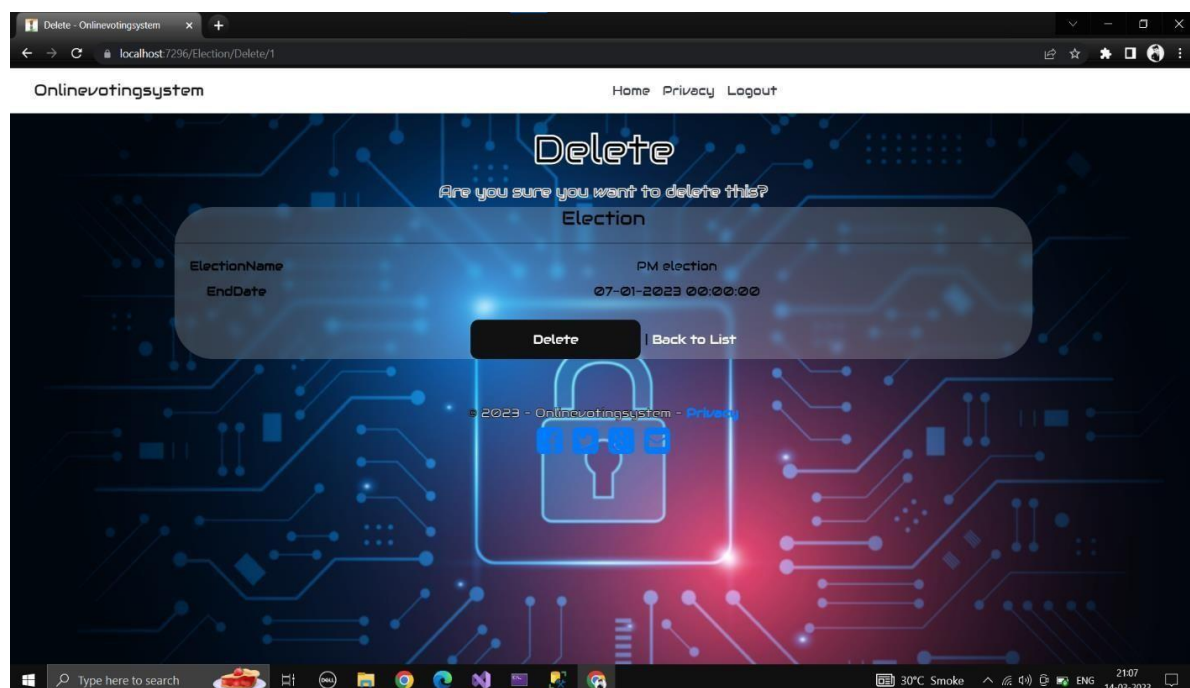
- **Create Candidate**



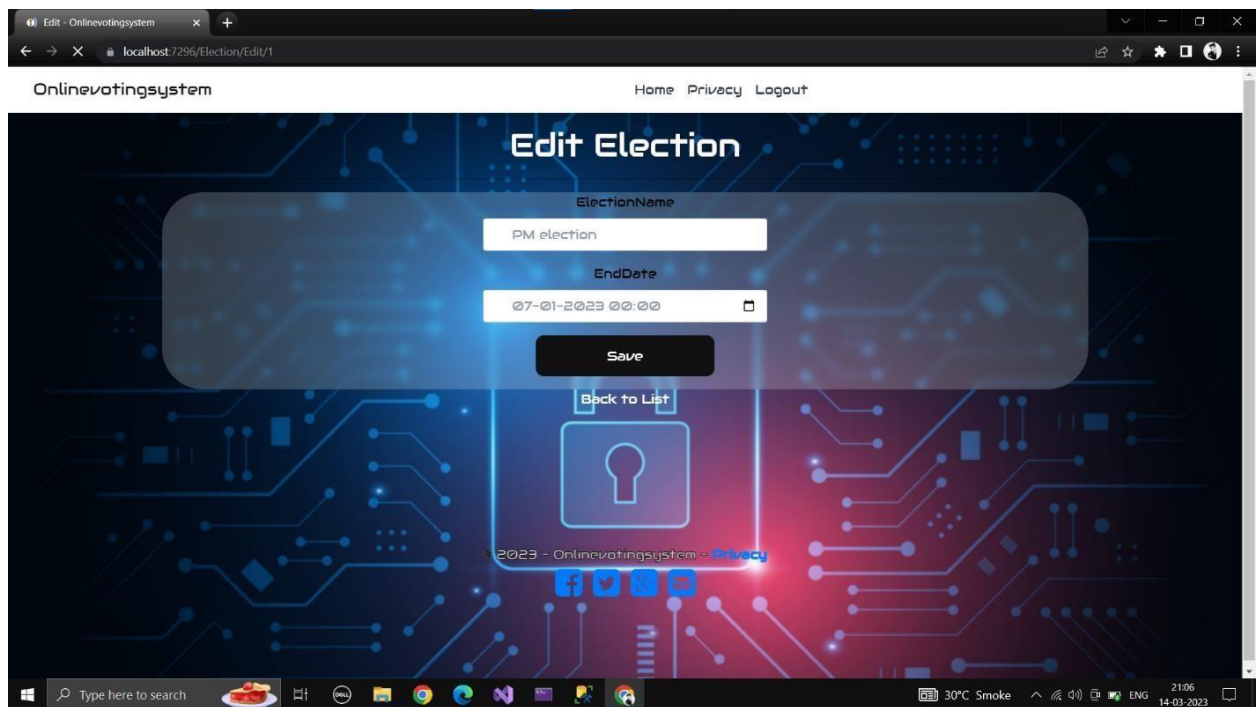
- **Create Election**



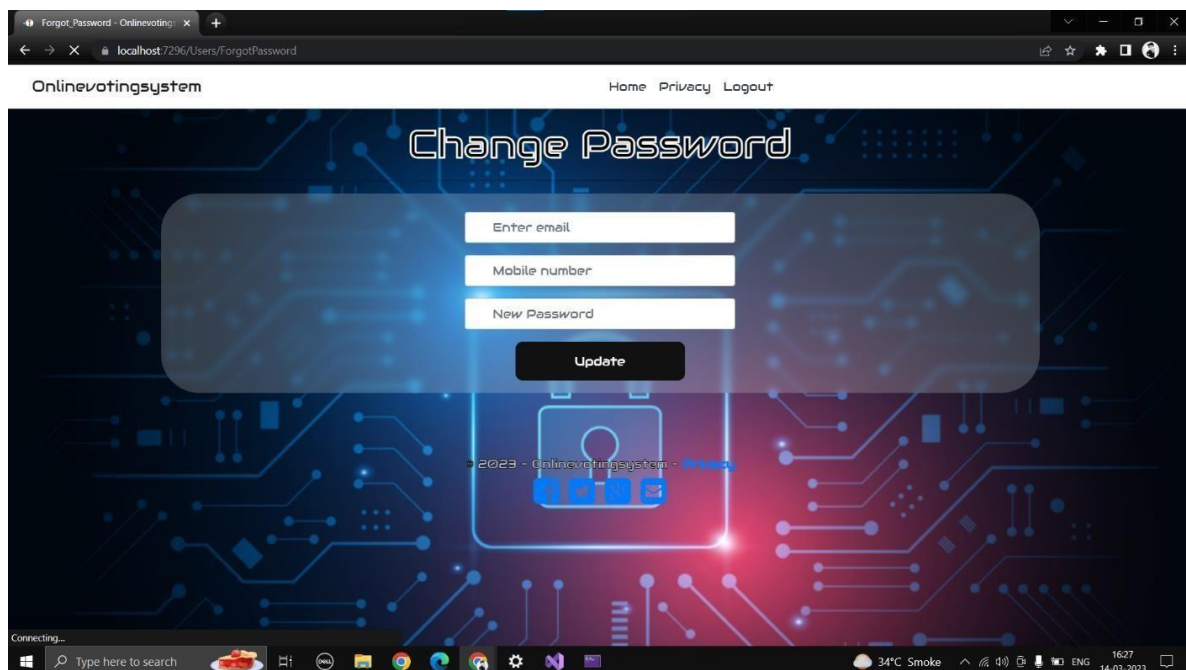
- **Delete Electio**



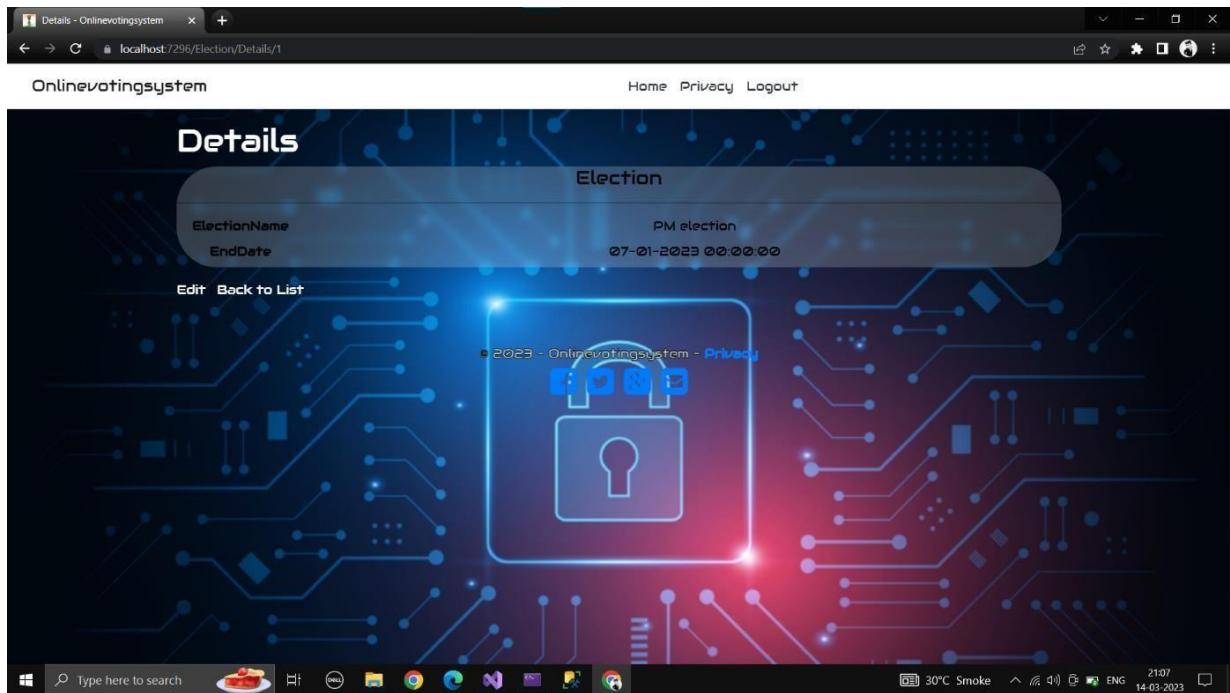
- **Edit Election**



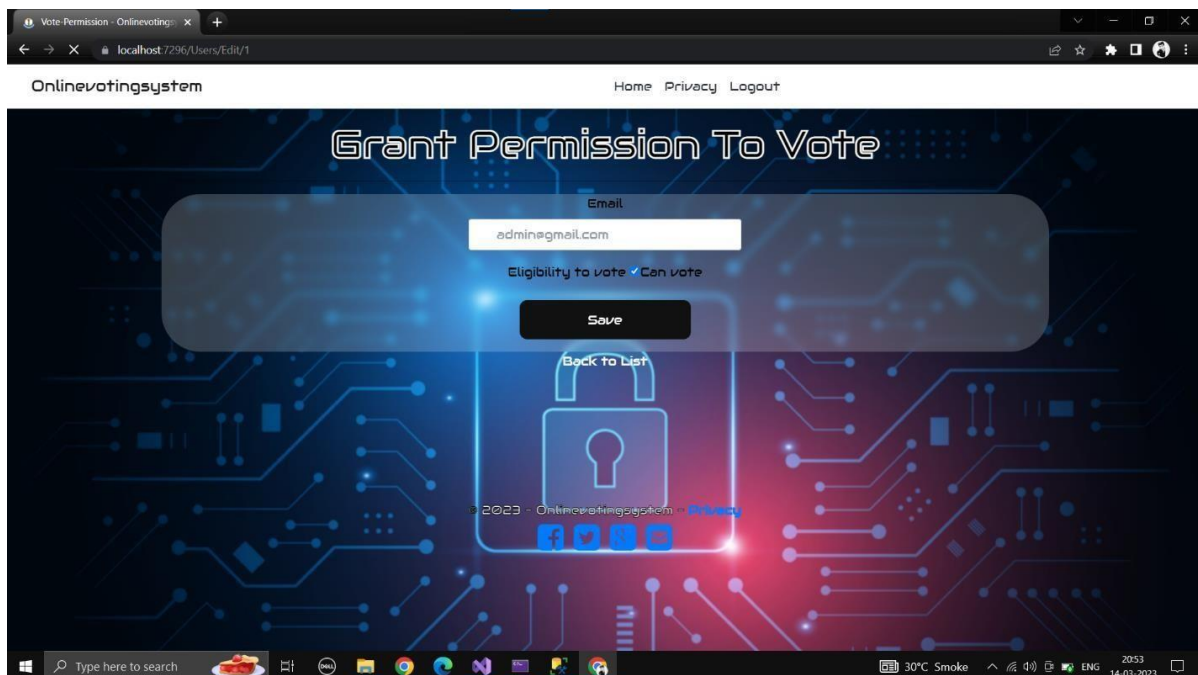
- **Forgot Password**



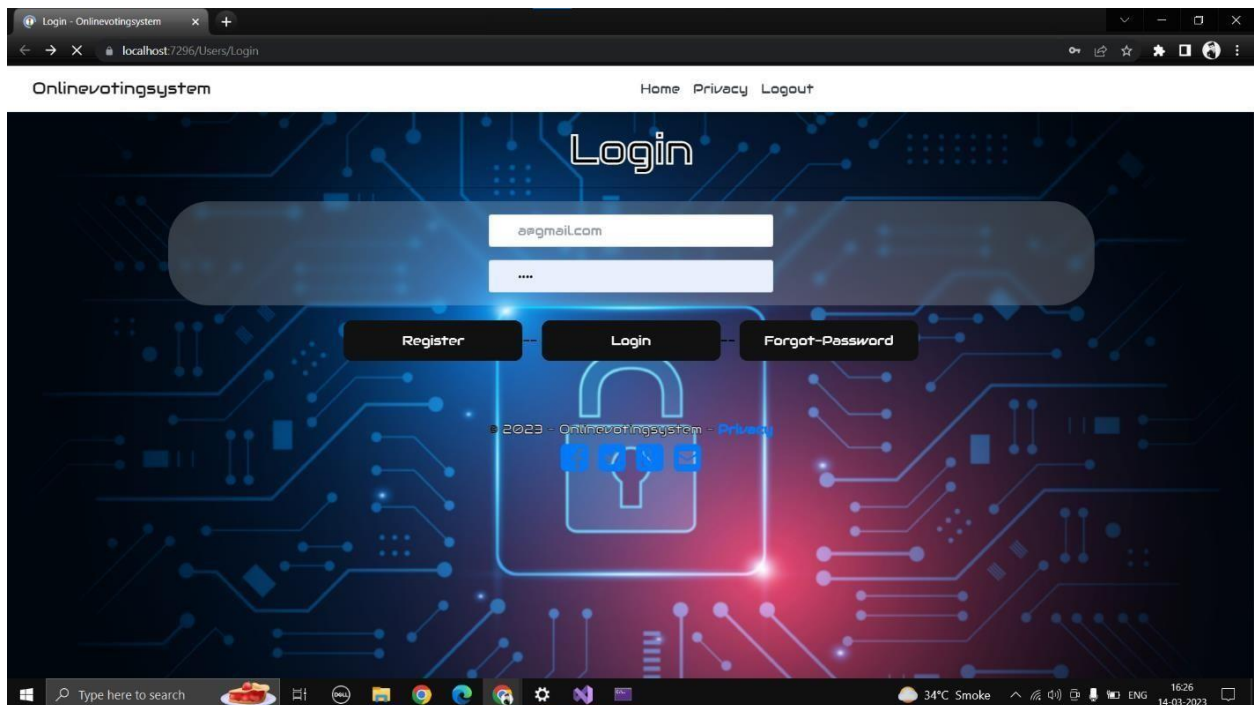
- Election details



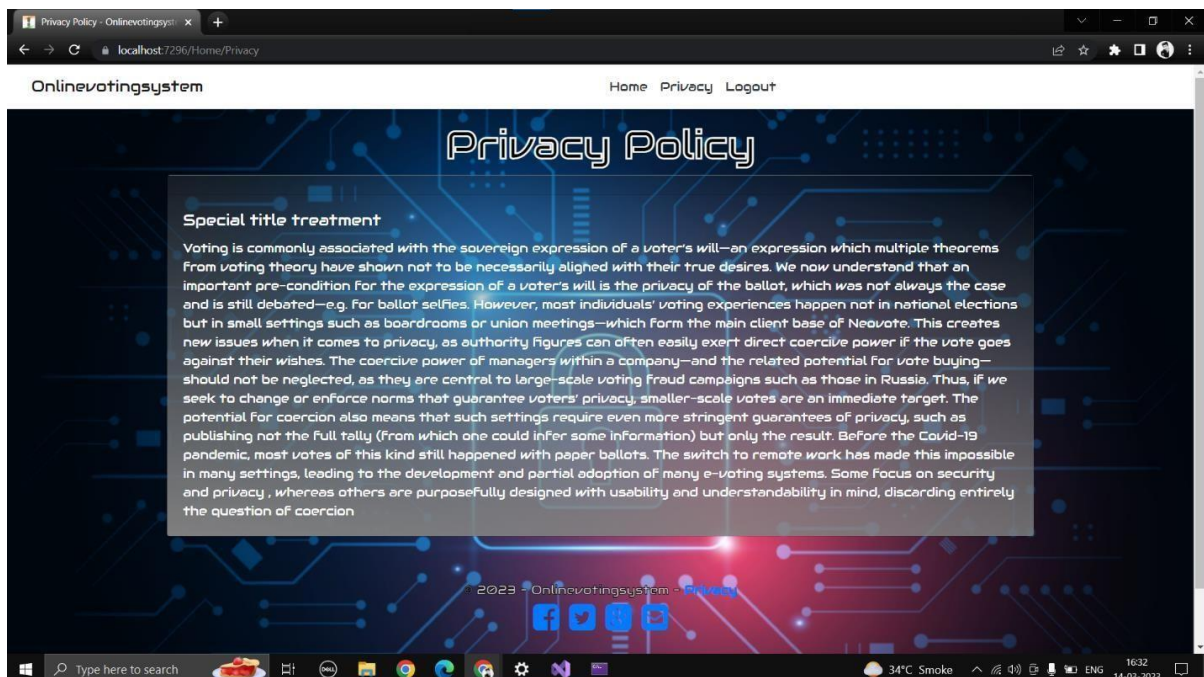
- Grant Permission to Vote



- Login



- Privacy



- **Register**

Onlinevotingsystem Home Privacy Logout

Create Users

First Name

Last Name

DateOfBirth

dd-mm-yyyy

--Select Gender--

Email

Mobile number

address

Register

IF Existing User ?

Login

2023 - Onlinevotingsystem - Privacy

- **User Election**

Onlinevotingsystem Home Privacy Logout

View Election

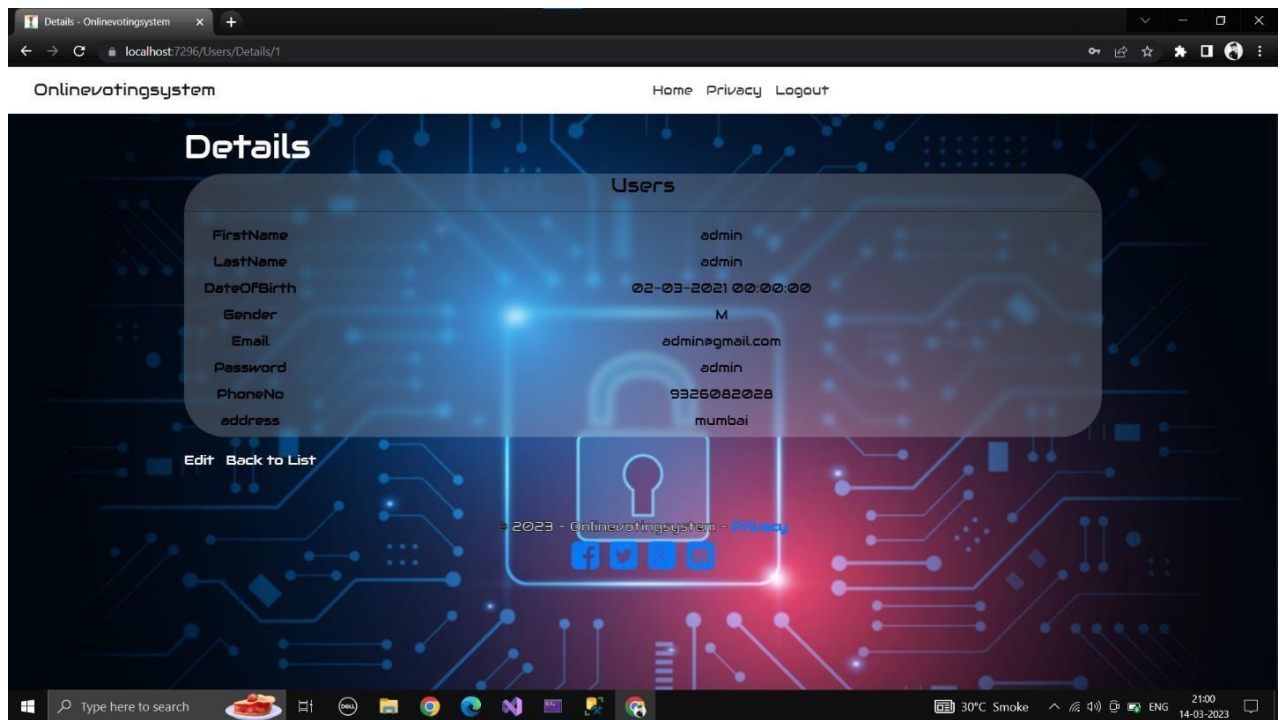
ElectionName	EndDate
PM Election	13-04-2023 00:00:00

see candidates |

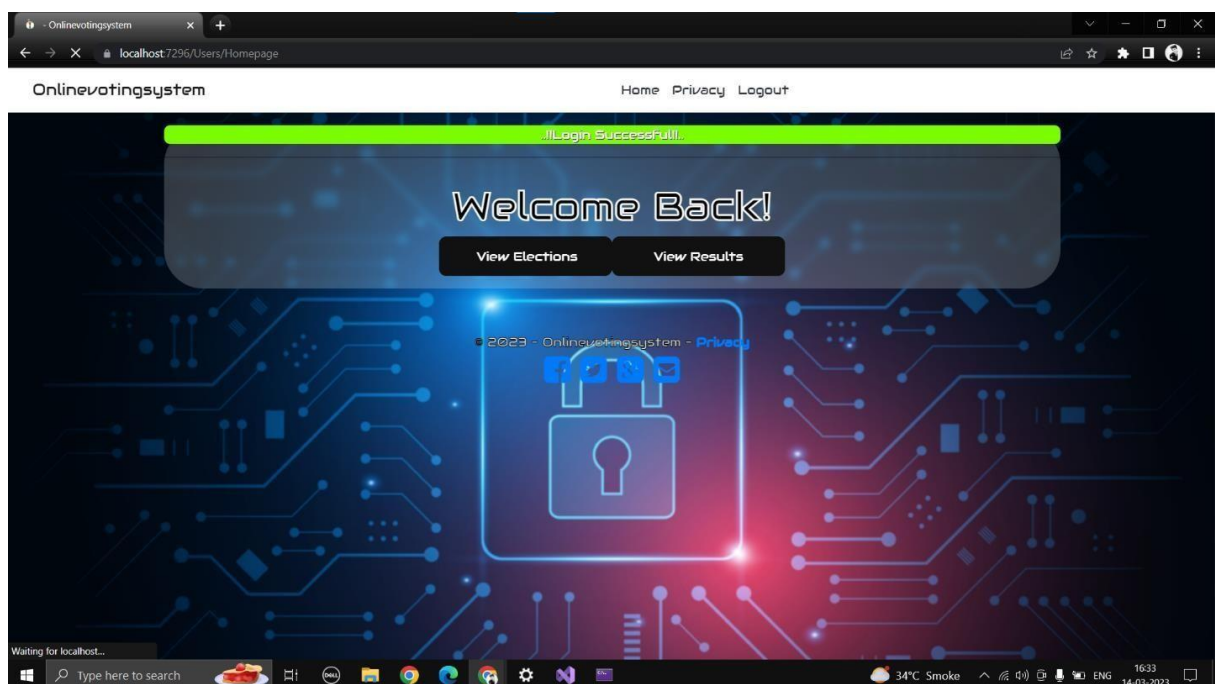
see Result |

2023 - Onlinevotingsystem - Privacy

- **User Details**



- **Users Homepage**



- View Candidates

Onlinevotingsystem Home Privacy Logout

TypeElection	FirstName	LastName	DateOfBirth	Gender	Image	PhoneNo	address	
4011	Narendra	Modi	01-01-0001 00:00:00	M		9892700764	Chawl No 31	Cast Vote
4011	Rahul	Gandhi	01-01-0001 00:00:00	M		9892700764	Chawl No 31	Cast Vote

2023 - Onlinevotingsystem - Privacy

- View Users

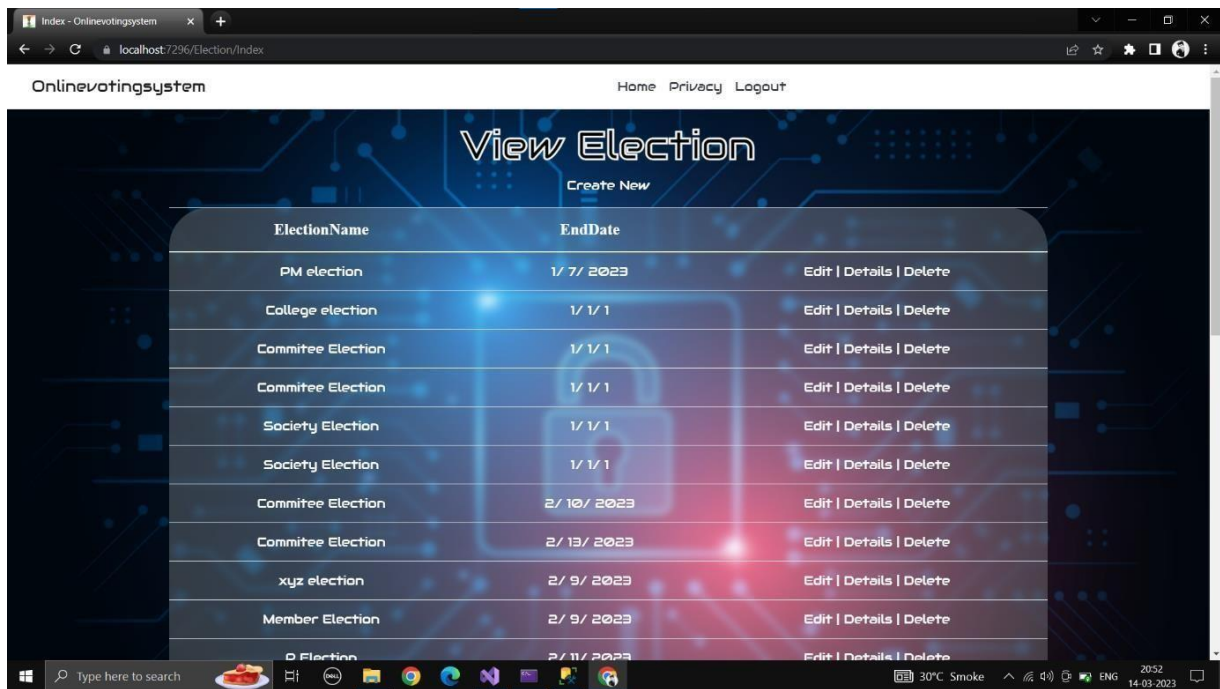
Onlinevotingsystem Home Privacy Logout

View Users

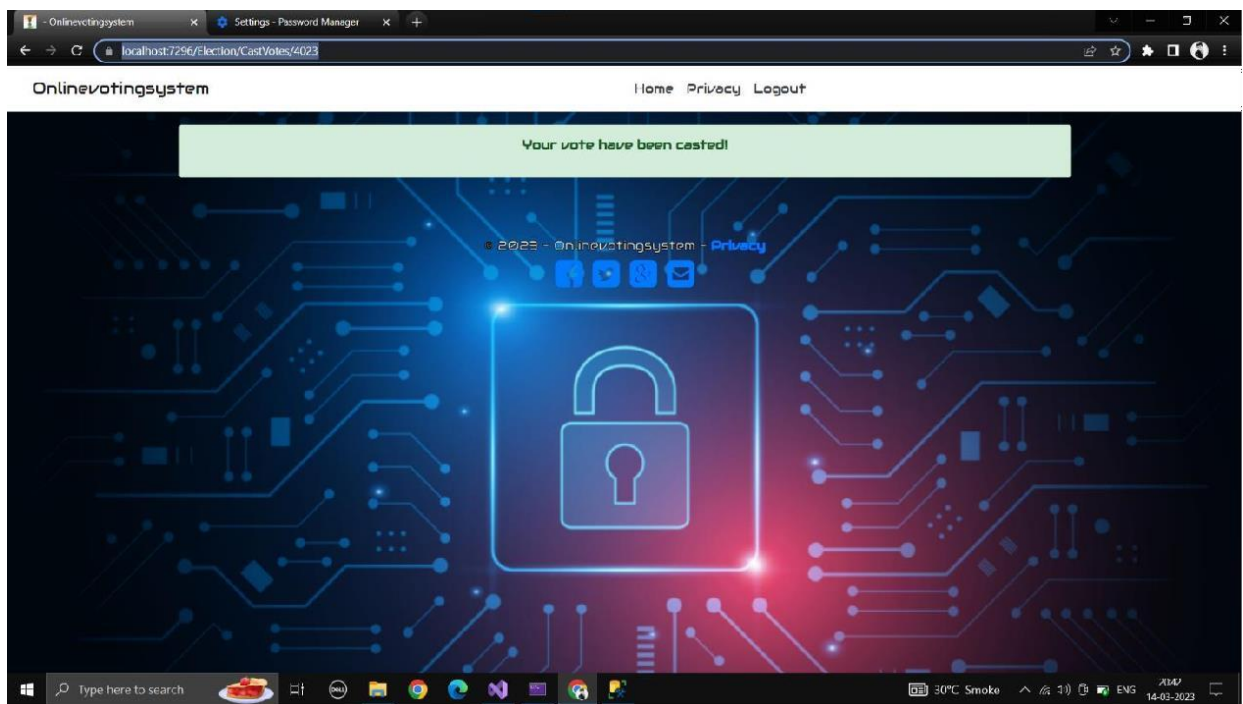
[Create New User](#)

FirstName	LastName	DateOfBirth	Gender	Email	Password	PhoneNo	address	
admin	admin	3/ 2/ 2021	M	admin@gmail.com	admin	9326082028	mumbai	Edit Details Delete
James	Nadarr	1/ 6/ 2023	M	jamesnadar41@gmail.com	james1234	9326082028	jakakes	Edit Details Delete
Subash	Nadar	1/ 20/ 2023	M	subashnadar199@gmail.com	QJvJnPLZey	9326082028	jakakes	Edit Details Delete
Ramesh	Nadar	1/ 20/ 2023	M	jamesnadar199@gmail.com	X9ET&CeMU	9326082028	jakakes	Edit Details Delete
James	Nadar	1/ 13/ 2023	M	admin	admin	9326082028	jakakes	Edit Details Delete

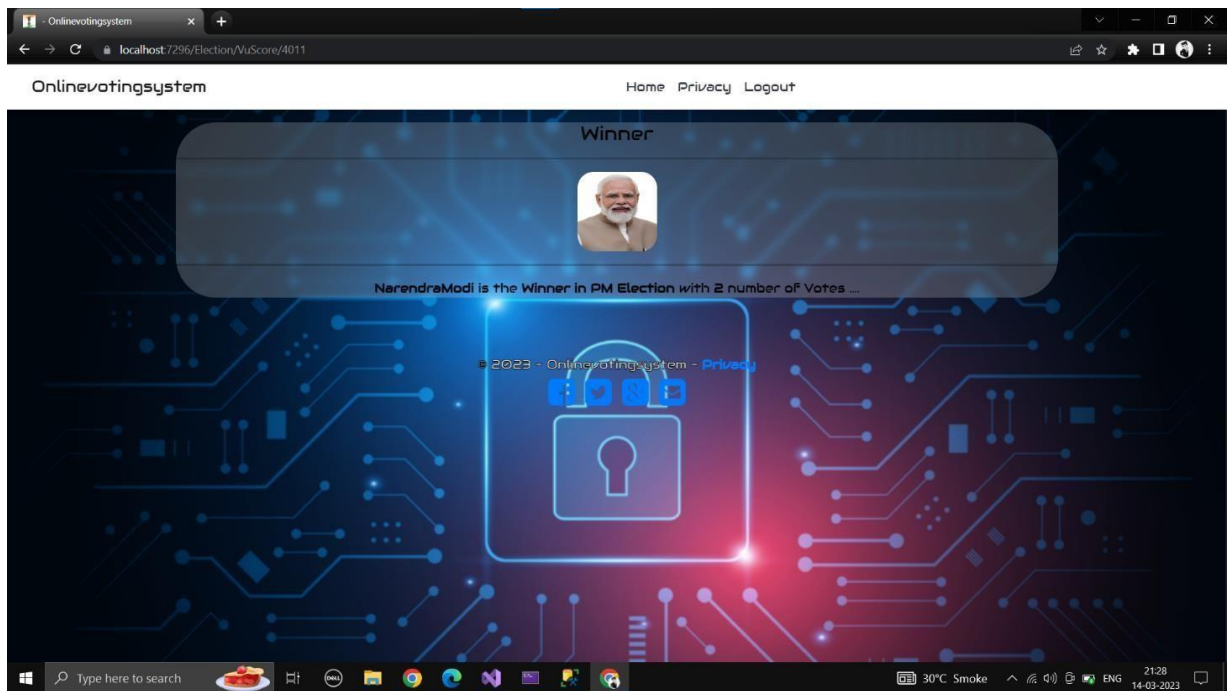
- **View Elections**



- **Vote Casted**



- **Winner**



Chapter 7

Conclusion

7.1 Conclusion

This online voting system will manage the voter's information by which the voter can log in and use his voting rights. The system will incorporate all features of the voting system. It provides the tools for maintaining voter's votes for every party and it counts the total no. of every party. In this user who is above 18 years' register his/her information on the database and when he/she wants to vote he/she has to log in with their id and password and can vote for any party only a single time. Voting detail is stored in the database and the result is displayed by calculation.

With the online voting system percentage of voting is increased. It decreases the cost and time of the voting process. It is very easy to use and it is very less time-consuming. It is very easy to debug. The traditional method of the manual voting system has a few drawbacks. This method is obviously not efficient as it wastes the voter's energy and is quite slow in terms of completion. This smart system allows the voters to cast their votes easily.

7.1.1 Significance of the System

Online Voting is a web-based voting system that will help you manage your elections easily and securely. This voting system can be used for casting votes during elections held in colleges, etc. In this system, the voter does not have to go to the polling booth to cast their vote. They can use their personal computer to cast their votes. There is a database that is maintained in which all the names of the voters with their complete information are stored. The voter registers himself by simply filling out a registration form to register the voters. If invalid/wrong details are submitted, then the person is not registered to vote. The site will be activated only on the day of voting. The advantage of online voting is that the voters have the choice of voting in their own free time and there is reduced congestion. It also minimizes errors in counting vote. The individual votes are submitted in a database which can be queried to find out which of the aspirants for a given post has the highest number of votes.

7.2 Limitations of the system

There are also several drawbacks relating to the online voting system. This shows that there is no 'golden solution' to facilitating access to the ballot and that each option has its own advantages and shortcomings. The main risks related to online voting solutions include:

- They may require an additional application or registration.
- Online voting solutions which take place in an uncontrolled environment may present a higher risk of fraud, coercion, family voting, impersonation, violation of ballot secrecy, or other compromises to the integrity of the vote.
- They may have financial and administrative consequences for members who are particularly in institutions such as hospitals or prisons.
- There is the risk that another person votes on behalf of the voter (It is difficult to identify the voter).
- It is difficult to observe the whole voting process. Observing the online voting may be more complex/difficult to organize the person voting. There may be information asymmetry between voters.
- There may be political disagreement over the method and extent of voting by a diaspora, particularly if this is seen to be politically advantageous to a particular party.

7.3 Future Scope of the projects

Data can be managed on the cloud so that it will be secured and managed efficiently. With the rise of mobile devices, the potential for mobile voting systems is significant. Mobile devices could be used for authentication, voting, and vote counting, making the voting process more convenient and accessible.

REFERENCES

1. <https://www.evoteco.in/>
2. <https://electionbuddy.com/>
3. <https://ieeexplore.ieee.org/document/8889792>
4. <http://www.ijarcs.info/index.php/Ijarcs/article/download/3072/2595>
5. <https://www.ijcaonline.org/archives/volume140/number8/24197-2018917736>