**Problem Statement**

1.1  PROJECT PURPOSE:

 In “ONLINE VOTING SYSTEM” a voter can use his/her voting right online without any difficulty. Online Voting System is a system which enables all citizens to cast their vote online.

The purpose is to increase the voting percentage across the country, as in the present system people have to visit the booth to cast their vote and those people who live out of their home town are not able to cast vote during the elections. So due to this the voting percentage across the country is very less. Through this software those people who live out of their home town will also be able to cast their votes as this system is online.

1.2   FUNCTIONALITIES OF THE SYSTEM:

 These functionalities have been implemented in the the program :

* Login for the admin : Admin can be able to check the data base of voters list and candidate list only if he/she enter the correct login credentials with username and password.
* Login for the user : User can be able to cast his/her vote only if he/she enter the correct login credentials with username and password.
* Client : In the client terminal the client will send the data to server which contains the information of name of voter and voter ID.It contains the functions of sockfd() , connect(), send(),receive() for creation of socket , connecting client to server, sending the data to server and receiving the data.
* Server **:** In the server terminal the server checks the list of voters and the voter who is going to vote if it matches they will allow to select the party symbol and can cast their vote,Otherwise they can exit. One who already voted is not be able to vote again.
* Checking the voter details **:** At server side it will check the entered voter details with the voters list , if matched they will be able to select the party to cast their vote else they are not able to vote.
* Selection of candidate symbol **:** The voter will be selecting the party symbol of respective candidate to cast their vote.
* Compiling the code :
* On Server’s terminal $ gcc server.c -o server On Client’s terminal $ gcc client.c -o client
* To run the code, First on the Server’s terminal $ ./server
* Then on the Client's terminal run $ ./client
* Closing Client and Server Use command “QUIT server” to quit client program. This will disconnect the client. You can exit the server by simply using Ctrl + C on the server terminal.

1.3 OPERATING ENVIRONMENT:

    Operating environment using TCP socket are:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu/C++

* 1. SOFTWARE REQUIREMENTS:
  2. UNIT TEST:

1. Admin:

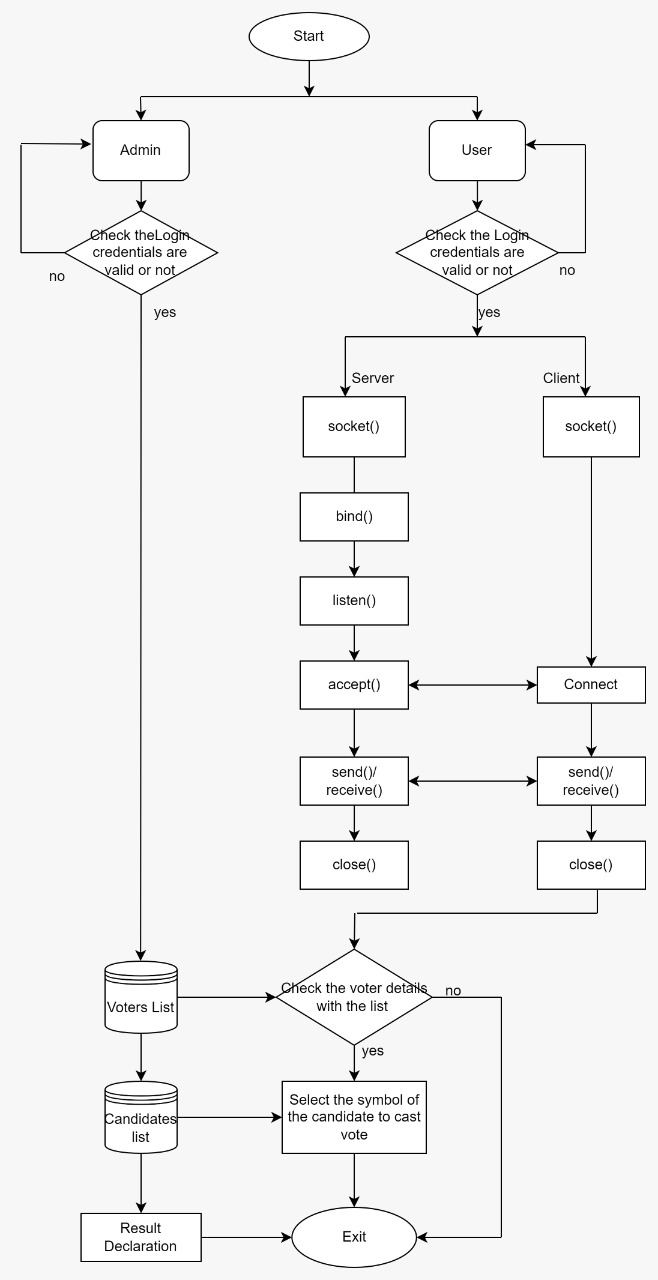
* SR1-UT1 : Login with valid username and password in order to check the database of voters list and candidate list.
* SR2-UT2 : If the login credentials are wrong, it will give message and ask re-login.

1. User :

* SR1-UT1 : Login with valid username and password in order to get the access.
* SR2-UT2 : If the login credentials are wrong, it will give message and ask re-login.

1. Client:
   * SR1-UT1: using the function of sockfd() socket creation will be done
   * SR2-UT2: using the function of connect(), it connects the client socket to server socket. If the connect() function fails then the client must receive an error message of connection failure.
   * SR3-UT3: using send(), the voter name and ID at client side will be send to server side.
   * SR4-UT4 : Using receive(),it receives the data from server.
2. Server:
   * SR1-UT1: Socket creation will be done using sockfd().
   * SR2-UT2: The port no. in the client side and server side should be same if not connection not establish.
   * SR3-UT3: Using bind(),it binds the socket to the address and port number, if fail must return the Binding error while listen() function must listen should listen for the client response.
   * SR4-UT4: Using accept(),it accepts the incoming connection from client.
   * SR5-UT5: Using receive(),it receives the data from client
   * SR6-UT6 : Using send(),it sends the data to the client

**2.3 FLOW DIAGRAM**



**1. Start :** This is the start block which indicates the start of the program, which will accept the admin and user credentials (like the username and password). On validation of these credentials the system will allow the admin and user to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.

**2. Login for Admin and User :** Entering the login credentials for admin to check the voters list and candidate lists , similarly for user is having the login credentials to cast their vote.

**3. List of voters :** Creating a database for voters list which contains their name and voter ID

4. **Candidates list :** , similarly for candidates also a database is creating having the party symbol.

5.**Client-Server communication :** Using TCP Socket a client server communication is established.

* **At client :**
* using the function of sockfd() socket creation will be done
* using the function of connect(), it connects the client socket to server socket
* using send(), the data at client side will be send to server side
* **At server :**
* Socket creation will be done using sockfd()
* Using bind(),it binds the socket to the address and port number
* using listen(),it waits for the client to approach the server to make a connection
* using accept(),it accepts the incoming connection from client
* using receive(),it receives the data from client

1. **Checking the voter details :** At server side it will check the entered voter details matched with the voters list , they will be able to select the party to cast their vote else they are not able to vote.
2. **Selecting the candidate symbol :** The voter is going to be select the party symbol of respective candidate to cast their vote.
3. One who already casted their vote,they are not able to vote again and they will be exit.
4. **Result Declaration :** After completing the voting time the result will be declared with the maximum number of votes.
5. **Exit :** The process will be terminated.

CONCLUSION:

DNS is a client/server network communication protocol. DNS clients send requests to the server while DNS servers send responses to the client.

If a client like a web browser sends a request containing a hostname, then a piece of software such as **DNS resolver** sends a request to the DNS server to obtain the IP address of a hostname. If DNS server does not contain the IP address associated with a hostname, then it forwards the request to another DNS server. If IP address has arrived at the resolver, which in turn completes the request over the internet protocol.

The application has two primary parts - Client and Server - and the reliable transfer is ensured using TCP with help of socket() API specifying SOCK\_STREAM.

The Client’s flow:

Checks for validity of arguments - needs server’s ip and port to fire up

Creates sockfd which controls the flow or the control of the connection

Connects socket ref by sockfd to server\_addr (address and port specified)

Set up data connection in listenfd descriptor and binds the socket to data\_addr port

Then until user quits, listens to commands, parses it using helpers, and executes them using the respective helper functions

The Server’s flow:

Opens a TCP connection and starts listening to incoming connections

Binds the listening file descriptor to the specified port

Creates a fork of its process for every incoming connection, closing listenfd for that particular fork

Receives client’s data and command

Takes the necessary actions