Team V-Elect - Group 3

**V-Elect**

Software Design Document

Authors:

* Ahmed Wael (900183285)
* Menna Ghalwash (900181104)
* Mariam Daabis (900182265)
* Mostafa Elsharkawy (900182569)

Date: (28/11/2020)

|  |  |  |
| --- | --- | --- |
|  |  | Software Design Document |
| **TABLE OF CONTENTS** |  |  |
| **1.** | **INTRODUCTION** | **3** |
| 1.1 | Purpose | 3 |
| 1.2 | Scope | 3 |
| 1.3 | Overview | 3 |
| 1.4 | Reference Material | 3 |
| 1.5 | Definitions and Acronyms | 4 |
| **2.** | **SYSTEM OVERVIEW** | **6** |
| **3.** | **SYSTEM ARCHITECTURE** | **9** |
| 3.1 | Architectural Design | 9 |
| 3.2 | Decomposition Description | 10 |
| 3.3 | Design Rationale | 12 |
| **4.** | **DATA DESIGN** | **14** |
| 4.1 | Data Description | 14 |
| 4.2 | Data Dictionary | 14 |
| **5.** | **COMPONENT DESIGN** | **26** |
| **6.** | **HUMAN INTERFACE DESIGN** | **30** |
| 6.1 | Overview of User Interface | 30 |
| 6.2 | Screen Images | 31 |
| 6.3 | Screen Objects and Actions | 44 |
| **7.** | **REQUIREMENTS MATRIX** | **48** |
| **8.** | **APPENDICES** | **49** |

**1. INTRODUCTION**

**1.1 Purpose**

This software design document describes the architecture and system design of V-Elect, the program designed to replace current election systems and opt for a more virtual experience. The document itself will include the hierarchy of our program’s design, including class diagrams, sequence diagrams, context diagrams, and sample screenshots from our user interface.

**1.2 Scope**

V-Elect represents a revolutionary approach to the voting process we know now. V-Elect is an online platform that will not only manage the voting process but also give the candidates the chance to share content, track statistics and engage in live sessions. It will give them the chance to have an interactive environment with fellow candidates as well as the voters where they can engage in debates for all of the voters to see.

All of this is done completely virtually where users, whether candidates, voters or campaigners, have the luxury to conduct an election experience using only their mobile phones/web browsers. V-Elect will also accommodate the people who prefer to vote in voting centers by providing multiple authorized locations.

**1.3 Overview**

This SDD has been organized for readers to view a general overview of our system (sections 2-3), then a more detailed segment regarding the implementation and structure of V-Elect (sections 4-6). We encourage readers to follow the chronological sequence of the document, except for section 8, which users may reference at any time after reading sections 2 and 3.

**1.4 Reference Material**

The documents used in this SDD primarily include the previously submitted SRS, which has already been provided.

The tools used for this SDD involve the following:

* Visual Paradigm - Standard Edition

* + <https://www.visual-paradigm.com/>
* Lucidchart

* + <https://www.lucidchart.com/pages/>
* Wireframe.cc

* + <https://wireframe.cc/>
* MySQL

* + <https://www.mysql.com/>
* Framer

* + <https://www.framer.com/>

**1.5 Definitions and Acronyms**

* **Admin:**
  + Administrator, or account with highest access in the V-Elect application/website
* **AES:**
  + Advanced Encryption Standard, which is a modern encryption algorithm that helps secure data being transferred over networks.
* **bcrypt:** 
  + A password-hashing function, which is known to be secure and one of the best for modern website password hashing. Hashing itself is the process of converting a password into a random series of characters based on the given hash function, which helps with identifying whether an input password is correct, without storing the password in plaintext.
* **Distro:**
  + Computer software distribution package, primarily a Linux distribution that entails certain features.
* **eKYC:**
  + electronic Know Your Customer, which is an automated third-party system that can provide validation in user identity, which will relieve a lot of manual verification if implemented.
* **MySQL:**
  + My Structured Query Language, a relational database management system used to create web databases.
* **OS:**
  + Operating system, a system software that manages computer hardware, software resources, and provides services for various programs.
* **SMS:**
  + Short Message Service, which is a popular text-messaging system used on most (if not all) mobile phones.
* **VEI:**
  + Virtual Elections Inc, the name of the company that has hired us to create V-Elect
* **Wireshark:**
  + A popular network packet analyzer that can acquire packets over a certain network. If packets are sent using plaintext, programs like Wireshark can allow other users to see passwords being sent to and from a server.

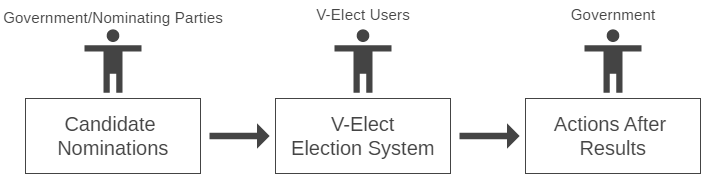
**2. SYSTEM OVERVIEW**

As expressed thoroughly in our SRS document, our system will involve a website and an optional application in further development, which are connected to multiple servers (depending on nation), for a better reach of users. In addition, there will be a centralized database to hold the information (encrypted and hashed in the case of passwords) of users of V-Elect. Finally, for our voting process itself, the application will connect to a blockchain peer-to-peer network, to keep track of a ledger of the candidates and the number of votes for those specific candidates. This will ensure the security, reliability, and authenticity requirements mentioned in our SRS.

To provide more reach to users without internet access, our product will utilize available computer centers to give those voters the opportunity to vote. These centers will contain V-Elect staff, or government staff required by the nation, depending on its laws, to ensure the safety of the voters as well as integrity of the system.

The application behaves as a stand-alone system that will be introduced to modern societies (until expansion), and can be described as a component of the election process within a region or nation, depending on the election itself. While V-Elect does display candidates and campaigners, it has no say in determining these members, and that authority remains with the nation’s government, or entity’s rules. The entity that agrees to use V-Elect for a specific purpose will be tasked with selecting the candidates for a vote, and determining voting days, as well as restrictions (if any) on a specific vote. After the vote takes place, V-Elect has no say in the following procedures, and it is up to the entity to take action depending on the results of said vote.

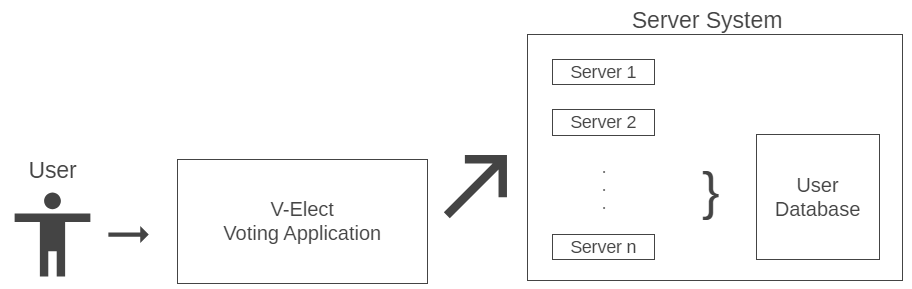
System Integration:



*Figure 2.1*

Figure 2.1 demonstrates V-Elect’s integration with the process of a national election. It only acts as a link between the two general processes of selecting candidates and placing them in office.

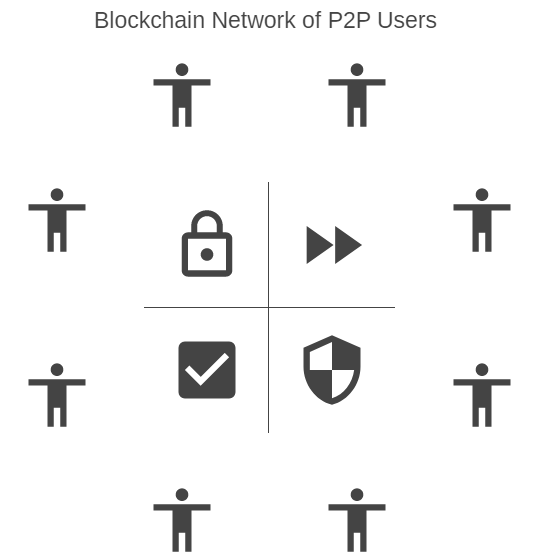
General Structure of V-Elect:



*Figure 2.2*

Figure 2.2 demonstrates the general overview of the application and its link with the multiple servers for broader reach. It also demonstrates a **centralized database** designed to store sensitive information, while also keeping said information encrypted, and in the case of passwords, hashed.

Blockchain Representation:



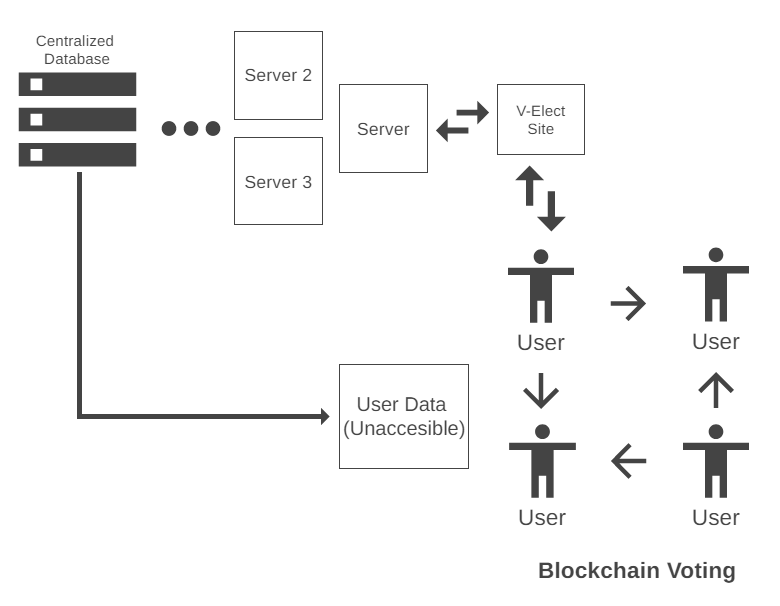
*Figure 2.3*

Figure 2.3 represents the benefits of using a blockchain approach, specifically the security it provides and authenticity when voting. Each person in the diagram above represents a user, or a block in the blockchain. The users are connected and all have a ledger containing the number of votes for each candidate during a specific vote. There is also a timer limit on the amount of blocks that can be added per hour (6 per hour is the standard). This guarantees that no votes can be duplicated, as this type of architecture hasn’t failed yet in modern society.

**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

General Overview of Architecture Design:



*Figure 3.1.2*

The layout above further describes how the individual components mentioned in section 2 are connected. As mentioned, the voting process through the blockchain system is as loosely coupled as possible with the centralized database, to ensure no traceability between the users who vote, and the number itself when tallied.

As mentioned earlier, users will be able to access our website/application through one of the nearest servers to the user’s location, while also maintaining a connection to the blockchain, which will be used to verify and keep track of votes during an election phase.

**Client-Server Communication:**

As a major subsystem involves users connecting to the V-Elect servers, we have decided to provide high-level encryption of any data being sent over the network (primarily AES, Advanced Encryption Standard).

**Centralized Database:**

To keep track of user records and hashed passwords, we have developed a centralized database to be hosted on a centralized server, or a highly recommended server provided by Google, depending on the budget provided. This database will generally be using MySQL due to the team’s knowledge of the database management system.

**Peer to Peer (P2P) Network:**

In addition to the network provided through the client-server components, a peer to peer network will be established to enable a blockchain voting system required by V-Elect.

**3.2 Decomposition Description**

**Subsystems and Modules in V-Elect:**

* **Servers:**
  + Each region will contain multiple servers extending reach to the entire region, or all centers and populated regions that must be involved in a voting process.
  + **Functions:**
    - *APIs to and from Database:*
      * Mainly involving any SELECT queries to retrieve security questions, or verifying user login information.
    - *Allowing connections to V-Elect Web Application:*
      * The main functions involve allowing users to access the website/application, with minimal errors and delays.
    - *Preparing and Connecting to Backups:*
      * In the case of a server error or failure, backup servers should be properly configured to run without delay, as the system itself is extremely sensitive, and cannot be down for more than 1 hour per day.
* **Blockchain:**
  + All users will be connected via a Peer-to-Peer network, and a decentralized blockchain, which helps keep track of votes while also adding a layer of security that no centralized system can top in our current era.
  + **Functions:**
    - *Verify User Vote:*
      * This will be done through the blockchain, by verifying that a new vote can be added securely and that it has not been tampered with.
    - *Update Ledger:*
      * After verifying a vote, the ledger containing the number of votes for the candidate voted for should be updated.
* **Web Application:**
  + This section contains a variety of functions, which will be listed in this section and described in section 5. This is due to the fact that the main component of our software resides in the V-Elect web application.
  + **Functions (Indexed by SRS):**
    - *4.1: Register Account*
    - *4.2-3: Approve Registration*
    - *4.4: User Log in*
    - *4.5: Vote for candidate*
    - *4.6: View Results and Statistics*
    - *4.7: Search for Content*
    - *4.8: View Content*
    - *4.9 - 4.11: Managing Content*
    - *4.12: Pay for V-Elect Tool*
    - *4.13: Use V-Elect Tool*
    - *4.14: Request/Join Live Talk*
    - *4.15: Moderate Discussions*
    - *4.16 - 4.17: Edit Profile*
    - *4.18: Report Bug*
    - *4.19: Add Campaigner*
  + **Note:**
    - Most of these functions have been covered in depth through the SRS document, which is why section 5 will focus on the main functions that can help illustrate the design of the subcomponents.
    - These functions will mainly include voting, registering, logging in, viewing content, and any other major functions we feel need describing in pseudocode or regular text.
* **Voting Centers:**
  + Our voting centers will provide the same functions as those in the web application, because the centers mainly consist of computers connected using V-Elect. The additional functions required are listed below, but they are external, so they won’t be implemented through our software.
  + **Functions:**
    - *Security Screening*
    - *Allow Users to Log In to Computers*
    - *Connect to V-Elect Servers*
* **Centralized Database:**
  + The centralized database does not perform many functions, but receives queries from the mainController in our V-Elect system. As such, the functions primarily involve receiving and executing statements.
  + **Functions:**
    - *Receive Query from mainController*
    - *Execute received query*
    - *Return results*
    - *Insert, Modify, Update, Delete relations (tables).*

**3.3 Design Rationale**

The rationale behind choosing these approaches is the level of security that is required to ensure a successful, untampered vote. Critical issues we may have primarily include the idea of a centralized database in the application design. It can be argued that a centralized database can cause a **single point of failure** in such an important process such as this. However, we decided on this in order to limit the reach that any user can have on the database, and try to keep it isolated in general, except V-Elect administrators with the highest privilege level (select few who may have access for maintenance, but not data).

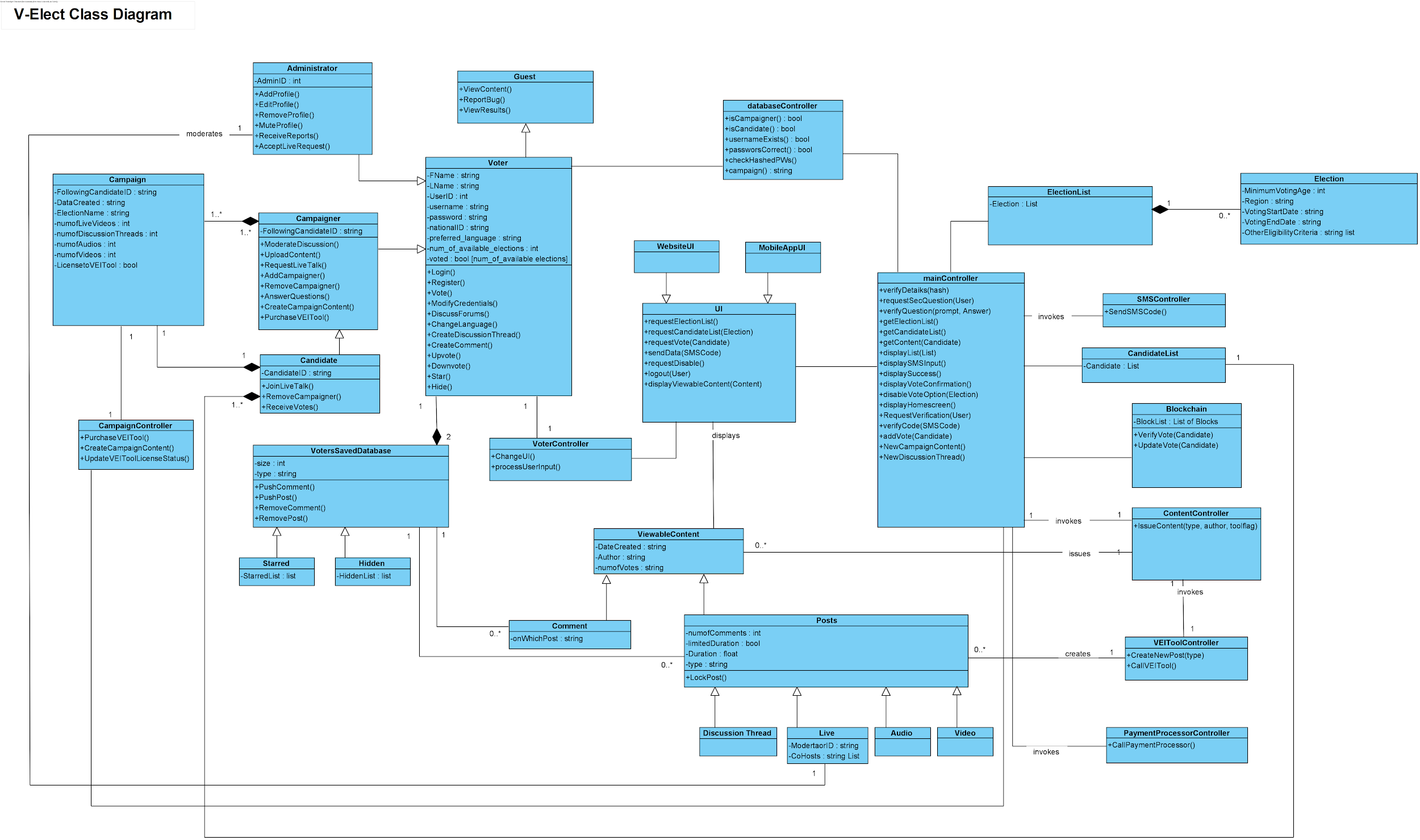
The idea of a centralized database has little to do with the voting process itself, which was why we left it in the original design. While user information may be stored in a database (as all user information should), voting takes place through the ledger on the blockchain, which has a separate process entirely. The database cannot affect the vote in any way, as users can login through backup servers, should the database fail, and user information is never needed due to the requirement of votes being anonymous and untraceable. The centralized database, in essence, is loosely coupled with the voting option in V-Elect, but can be used to store the results after scores are tallied through the blockchain.

Other service-oriented architectures and distributed systems were suggested, but we believe that our approach provides a simple hybrid with less points of failure for the overall system. As the website/application will be supported through the multiple servers, this helps with maintaining reliability and accessibility, should one of the servers fail. The blockchain itself is decentralized by definition, so it cannot have a single point of failure unless all machines in an entire region are destroyed simultaneously, and the chances of that are slim to none.

**4. DATA DESIGN**

**4.1 Data Description**

**V-Elect Class Diagram:**

****

*Figure 4.1.1*

**4.2 Data Dictionary**

**Voter Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | FName | First Name |
| string | Lname | Last Name |
| int | UserID | User Identification Number |
| string | username | Username to log in |
| string | password | Password to log in |
| string | nationalID | User’s national ID |
| string | preferred\_language | English or Arabic until future version release |
| int | num\_of\_available\_elections | Number of available elections currently |
| boolean[] | voted | Number of elections user has already voted int |
| bool | Login() | Function to log in current user, returns true if successful |
| bool | Register() | Function to register user |
| bool | Vote() | Function to perform vote for a candidate |
| bool | ModifyCredentials() | Allows user to edit profile |
| bool | DiscussForums() | Allows user to view discussion forums, where other activities occur |
| bool | ChangeLanguage() | Allows user to change language based on preference |

**Campaigner Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | FollowingCandidateID() | Determines which candidate this campaigner follows |
| bool | ModerateDiscussion() | Allows user to moderate discussion threads |
| bool | RequestLiveTalk() | Allows user to schedule a live talk |
| bool | AddCampaigner() | Allows user to add a campaigner |
| bool | AnswerQuestions() | Allows user to enter an answer |
| bool | CreateCampaignContent() | Allows user to create content |
| bool | PurchaseVEITool() | Allows user to purchase a VEI tool |

**Campaign Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | FollowingCandidateID | Determines which candidate this campaigner follows |
| string | DataCreated | Assigns date to the created content |
| string | ElectionName | Allows user to set the election name |
| int | NumofLiveVideos | Returns number of live videos |
| int | numofDiscussionThreads | Returns number of discussion threads |
| int | numofAudios | Returns number of audios available |
| bool | LicensetoVEITool | Checks whether a license is available or not |

**Administrator Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| int | AdminID | Admin ID |
| bool | AddProfile() | Function to add new profile |
| bool | EditProfile() | Function that allows user to edit profile |
| bool | RemoveProfile() | Functions that allows user to remove a profile |
| bool | MuteProfile() | Functions that allows user to mute a profile |
| bool | ReceiveReports() | Returns report |
| bool | AcceptLiveRequests() | Returns true if live is accepted |

**Campaign Controller Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | PurchaseVEITool() | Function to purchase the VEI Content tool for campaigners |
| bool | CreateCampaignContent() | Function for campaigners to generate content on the website |
| bool | UpdateVEIToolLicenseStatus() | Function to determine the status of a campaigner, whether tool has been purchased or not |

**VotersSavedDatabase Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| int | size |  |
| string | type |  |
| bool | PushComment() | Pushes comment onto website |
| bool | PushPost() | Pushes and posts a post on the website |
| bool | RemoveComment() | Removes comment from website |
| bool | RemovePost() | Removes post from website |

**Starred and Hidden Classes:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| list | StarredList | Contains list of starred posts and comments for voter |
| list | HiddenList | Contains list of hidden posts and comments for voter |

**Comment Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | onWhichPost | Determines which post by ID that this comment belongs to |

**Guest Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | ViewContent() | Function that allows guest users to view content |
| bool | ReportBug() | Function that allows guest users to report bugs |
| bool | ViewResults() | Function that allows guest users to view results |

**Voter Controller:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | ChangeUI() | Determines what to display on the UI by connecting to it |
| string | processUserInput() | Processes input string by user and returns it in desired format (depending on class requiring it) |

**UI Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| list | requestElectionList() | Returns list of possible active elections |
| list | requestCandidateList(election) | Returns list of possible candidates in selected election |
| bool | requestVote(Candidate) | Performs a request to vote for selected candidate through blockchain |
| bool | sendData(SMSCode) | Sends data from user to be requested by SMS controller |
| bool | requestDisable() | Requests from the mainController to disable voting option for user |
| bool | logout(User) | Requests a log out for selected user, and returns true if successful |
| bool | displayViewableContent(content) | After retrieving content, this function displays it on screen |

**WebsiteUI Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| void | readjust() | Readjusts the styling (dimensions, positioning, padding, margins, and alignment) of elements to fit the website view |

**MobileAppUI Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| void | readjust() | Readjusts the styling (dimensions, positioning, padding, margins, and alignment) of elements to fit the mobile app view |

**DatabaseController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | isCampaigner() | Checks validity of campaigner |
| bool | isCandidate() | Checks validity of candidate |
| bool | usernameExists() | Checks validity of username |
| bool | passwordCorrect() | Checks validity of entered password |
| bool | checkHashedPWs() | Checks hashed passwords |
| string | campaign() | Checks the campaign name |

**Viewable Content Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | DateCreated | Contains the date when content was created |
| string | Author | Contains the author of the post |
| string | numofVotes | Contains the number of votes for the specific post |

**Posts Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| int | numofComments | Number of comments on post |
| bool | limitedDuration | Whether content is limited or not |
| float | Duration | Time left for content to be visible |
| string | type | Type of post contained in a string (possible campaign) |
| bool | LockPost() | Disallow users from commenting or reacting to post, done by moderators mainly |

**Discussion Thread, Live, Audio, and Video classes:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| string | ModeratorID | The ID of the moderator of this live discussion |
| string List | CoHosts | The cohosts presenting the live discussion |

**mainController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | verifyDetails(hash) | Verifies the details of a user who just entered his password |
| string | requestSecQuestion(user) | Requests the security question for the selected user, returns the question as a string |
| bool | verifyQuestion(prompt, answer) | Verifies whether the given answer to the security question was true or false |
| List | getElectionList() | Returns list of possible elections |
| List | getCandidateList() | Returns list of possible candidates for specific election |
| bool | getContent(Candidate) | Acquires the content for a specific candidate, and sends it to UI |
| bool | displayList(List) | Displays any inputted list on UI, whether it is for candidates or elections |
| bool | displaySMSInput() | Displays SMS input on screen for user |
| bool | displaySuccess() | Displays Success prompt through UI on screen |
| bool | displayVoteConfirmation() | Displays confirmed vote if true, and sends display to UI |
| bool | disableVoteOption(Election) | Disables the voter’s option to vote by marking that voter as voted |
| bool | displayHomeScreen() | Command sent to UI when user logs out |
| bool | RequestVerification(User) | Requests necessary verification, through SMS, email, or centralized database |
| bool | verifyCode(SMSCode) | Verifies SMS code through SMS controller |
| bool | addVote(Candidate) | Adds a vote to a specific candidate through blockchain |
| bool | newCampaignContent() | Finds updated content for a campaigner |
| bool | newDiscussionThread() | Creates new discussion thread in a particular section. |

**ElectionList Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| List <Election> | Elections | A list of all the elections that are in progress right now (in progress means in campaigning, voting, or short-term post-voting stage) |

**Election Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| int | MinimumVotingAge | Age Restriction on Elections (some elections are 16+, some are 18+, etc.) |
| string | Region | Location where the election is taking place |
| string | VotingStartDate | Day and Time that voting begins |
| string | VotingEndDate | Day and Time that voting ends |
| string | OtherEligibilityCriteria | Eg: Being a resident for a minimum number of years, being a member of a sporting club whose elections are ongoing, etc. |
| string | TypeofElection | How votes should be counted and scores calculated (eg: Electoral College Voting, First-Past-the-Post, other forms of Popular Vote, etc.) |

**SMSController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | SendSMSCode() | Function that sends SMS code |

**Candidate List Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| List | Candidate | Contains the list of possible candidates for a specific election |

**Blockchain Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| List | BlockList | List of blocks in the blockchain |
| bool | VerifyVote(Candidate) | Verifies that a certain vote is valid for a candidate through P2P blockchain checking |
| bool | UpdateVote(Candidate) | Updates the number of votes for a certain candidate through ledger on the blockchain decentralized system |

**ContentController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | IssueContent(type, author, toolflag) | Function that issues content using 3 parameters type, author and toolflag. |

**VEIToolController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Post | CreateNewPost(type) | Receives the type of post that should be created |
| bool | CallVEITool | Uses previously received type (in previous functiom) to call the VEITool responsible for creating posts. Returns false if campaign does not have license to use VEITool, and true if it does (along with the post) |

**PaymentProcessorController Class:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| bool | CallPaymentProcessor() | Calls payment processor (3rd party) to perform necessary transaction and returns true if successful |

**5. COMPONENT DESIGN**

This section will primarily focus on the functions that concern the web application V-Elect. These functions have a higher priority than the general system functions that are performed by the servers or the blockchain network. We have provided a reference to these codes in section 7, as the functions themselves come from the SRS document, which has previously been provided.

As mentioned previously, in section 3.2, this section will also focus on the functions that require further clarification, and if the readers wish to learn more about the other functions with a lesser priority, they can visit the previously submitted SRS document, which provides more information in detail regarding the goals and functionalities of the given functions.

* **Functions (Referred to in Section 7 and SRS):**
  + *4.1: Register Account*
    - REQ-1: Verify user information

Ensure unique National ID and phone number

Ensure password length is of 16 char minimum

Ensure valid age:

Today.date - input.date > 18

Depending on user location

Return true if all of above

* + - REQ-2: Send SMS verification code

Contact SMSController for verification(number)

SMSController.sendCode(number)

AwaitInput…

Upon receiving input:

If (input == verificationCode):

Return true

Else:

Return false

* + - REQ-3: Send email verification link

Contact EmailController for verification(email)

EmailController.sendCode(email)

AwaitInput . . .

Upon receiving input:

If (input == verificationCode):

Return true

Else:

Return false

* + - REQ-4: Add user to registered database

Contact DBController

Result = Send(“INSERT INTO registers” + user\_details)

If (result == true)

Return true

Else

Return false

* + *4.4: User Log In*
    - REQ-1: System verifies user information

#Include hash\_function.h

User\_info = hash(User)

send(user\_info)to MySQL db

Check if user\_info exists

If exists, return true

Else return false

* + - REQ-2: System allows user access to voter privileges

Request mainController for verification

mainController returns true to GUI

GUI displays Home Screen with user\_info

GUI displays all options for “Voter” Class

* + *4.5: Vote for Candidate*
    - REQ-1: Allow voting for election

Include election\_dates.h

If (today.date WITHIN election.dates)

Return true

Else

Return false

* + - REQ-2: Verify User

STAGE 1:

Contact DB Controller

Send encrypted(user\_details)to DB

Check if user\_exists, and if so, return true

Else return false

STAGE 2:

Contact SMSController for verification(number)

SMSController.sendCode(number)

AwaitInput…

Upon receiving input:

If (input == verificationCode):

Return true

Else:

Return false

STAGE 3:

If (Stage1 && Stage2) return true;

else return false

* + - REQ-3: Display Election/Candidate Options

Contact UI Controller (GUI)

GUI.display(ElectionList)

Input user selection

GUI.display(selection.CandidateList)

Input user selection

GUI.prompt(“Are you sure you wish to vote for “ + candidate\_name + “ ?”)

Input user click on “Okay”

If clicked:

Blockchain.vote(Candidate)

Else:

Return to Candidate list

* + - REQ-4: Add vote to candidate’s votes

Contact Blockchain

Connect to P2P Network

Try {

Candidate.votes++

Verify()

} catch (error)

If (error = error\_code\_a)

Return false //This error code implies a tampered vote

Else

Return true

* + *4.8: View Content*
    - REQ-1: Fetching Discussions

Contact contentController

While (page = scrolled)

retrieveContent()

* + - REQ-2: Star Thread / Comment

Contact contentController

Contact DBController

If (user\_selection = ‘star’)

Post.stars++

commit() //Save stars in DB

* + - REQ-3: Hide Thread (simple)

If (post.hide == false)

display(post)

* + - REQ-4: Hide Comment

If (comment.hide == false)

display(comment)

* + *4.18: Report Bug*
    - REQ-1: Reporting bug

If (user.select == ‘Report Bug’)

GUI.display(report\_form)

Await text input from user

Await ’Submit’ click from user

If (submit == true && text != “”)

Return true

Else:

Return false //no text input

**6. HUMAN INTERFACE DESIGN**

**6.1 Overview of User Interface**

* Functionality 1:

Signup interface: the user will be asked to sign up to their accounts the first time they access the website/application by entering their information and details to be able to access the website/application.

* Functionality 2:

Login interface: the user will be asked to login to their accounts by entering their national ID / mobile number, password and security question.

* Functionality 3:

The main interface (before voting period): which includes the main options the user can use; an option for Q/A discussions, an option for viewing content (which includes watching recorded videos, discussion forums, recorded webinars and requesting live scheduled talks) and finally an option for the live sessions which will be conducted.

* Functionality 4:

The main interface (during voting period): the main interface during the voting period will include a button that will allow the users to vote, the results and statistics of the elections, viewing content, going live and Q/A.

* Functionality 5:

Help interface: this interface embraces buttons that helps the user with any technical issues. It includes reporting any problems the user is facing, how to access specific features with a list of all the features on the website/application, how to use the application/website and finally a ‘frequently asked questions’ button.

* Functionality 6:

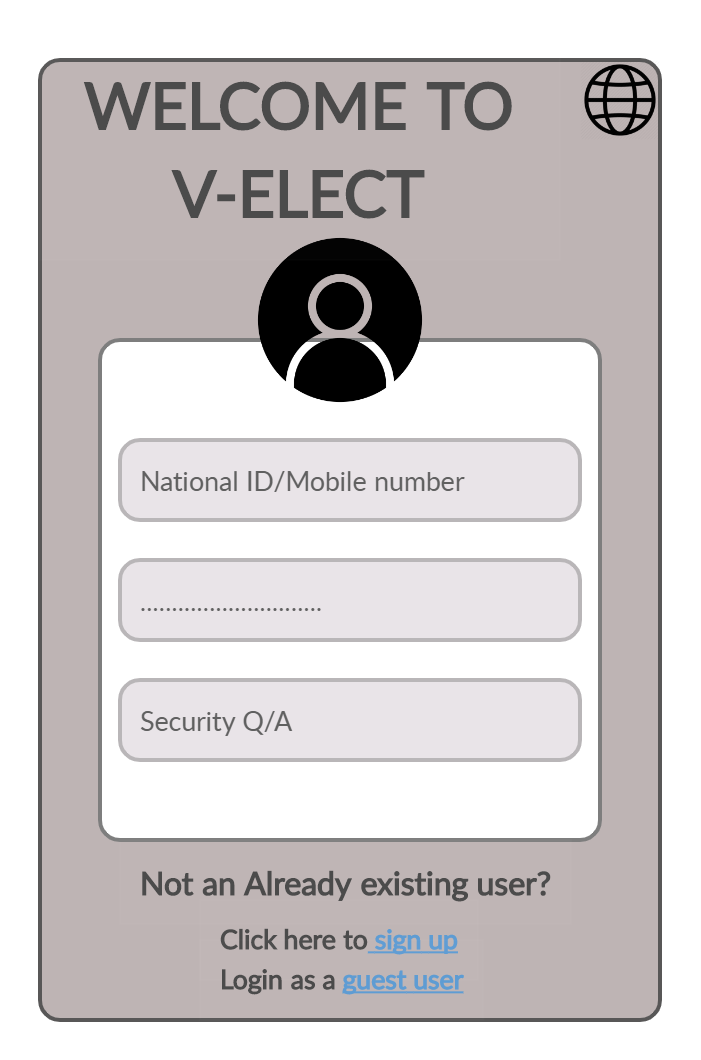
Profile interface: the profile interface includes all the personal details about the user, in addition to having access to changing some of the information.

* Functionality 7:

Changing the Language interface: this button allows the user to access the website/ application in more than one language. These languages are limited to English and Arabic.

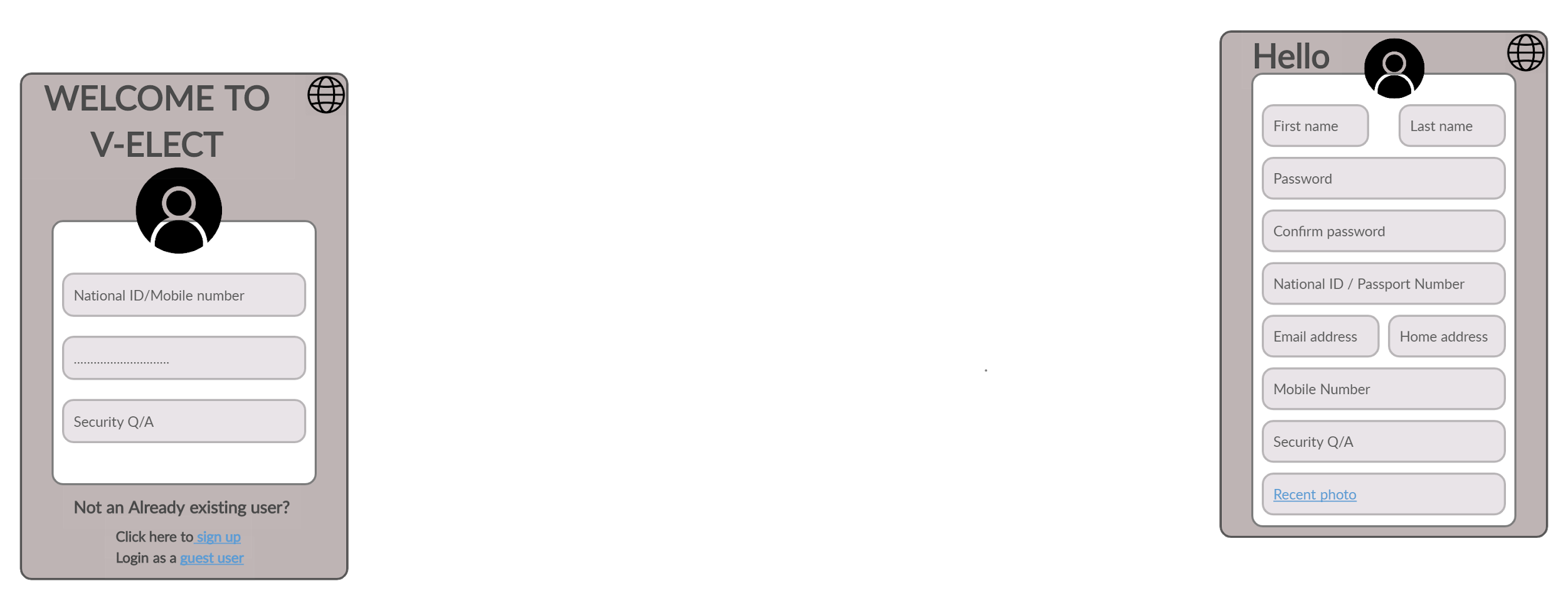
**6.2 Screen Images**

**Login Interface**



**This figure shows the login page which will appear to the user when they open the application. If it's the first time for the user to use the application, they will be given a choice of whether to sign up by filling all the details and information about themselves or login as a guest user. The guest user only views but cannot neither vote nor interact in any way possible.**

**Sign up Interface**



**This is the sign up page which will appear to the user that chooses to sign up to an account rather than logging in to their accounts. By signing up the user will have to fill all their personal details in addition to attaching some documents to verify that the information they filled is valid to make them an actual user that will be able to vote.**

**Main Interface (before voting)** 

**This figure shows the main page before the voting period. This page includes:**

1. **Q/A button which allows the user to submit a form with any question they want to ask the candidates about.**
2. **View content button which contains several content that the users will be interested to observe.**
3. **Go live button which allows the users to join any live sessions by either the candidates or the campaigners**

**Main Interface (During Voting)**



**This figure shows the main user interface during the voting period. In case the user wants to vote they should press on ‘vote now’ which will direct them to the page where they will be allowed to vote for their chosen candidate. Other features are results and statistics which contain the voting results for every candidate and the statistics between candidates. In addition to all the previous / old features of viewing content, going live with candidates or campaigners and finally Q/A.**

**Help Interface**



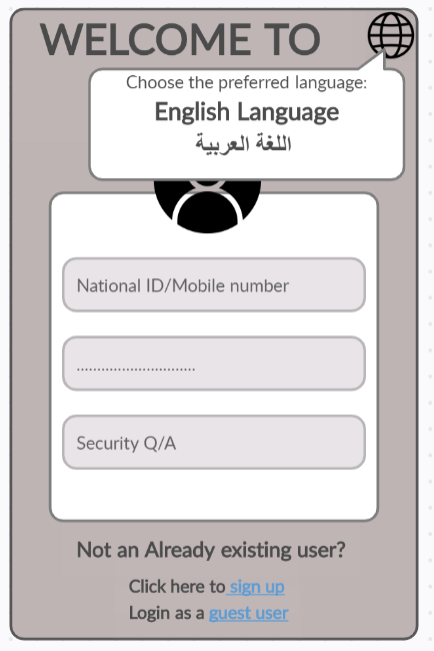
**This feature is for any technical issues faced by the users, they should report the case through the form in ‘report a problem’. Other buttons help the user with using the application / website. In addition to this interface offers a guide on how to access a list of features and how to use them. Finally this figure shows that the application offers a list of all the previously asked questions from the users.**

**Profile Interface**

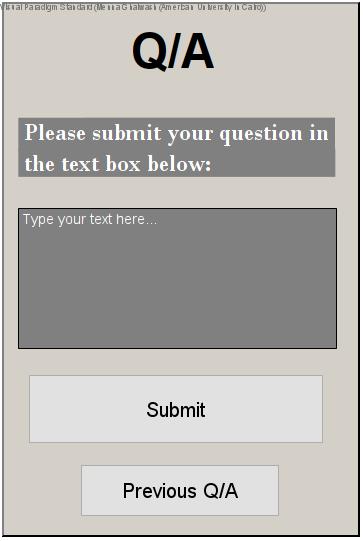


**This interface is for showing the user all the personal information they have filled which is verified by the documents they attached. The user can change limited information in case of any change; for example, home address, mobile number,...etc. Not all their information can be edited by thm since there are things that cannot change like their name and national id/ passport number for example.**

**Changing language Interface**

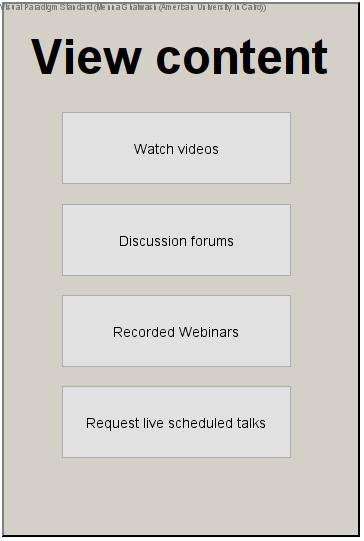


**This interface shows how the user can change the language from English to Arabic or from Arabic to english using the icon in the top right corner. This icon only appears in the login page and the sign-up page.**

**َQ/A Interface**

**The Q/A button allows the user to submit their questions which will be answered by either the candidates or the campaigners. In addition, it also allows the user to check previously submitted questions.**

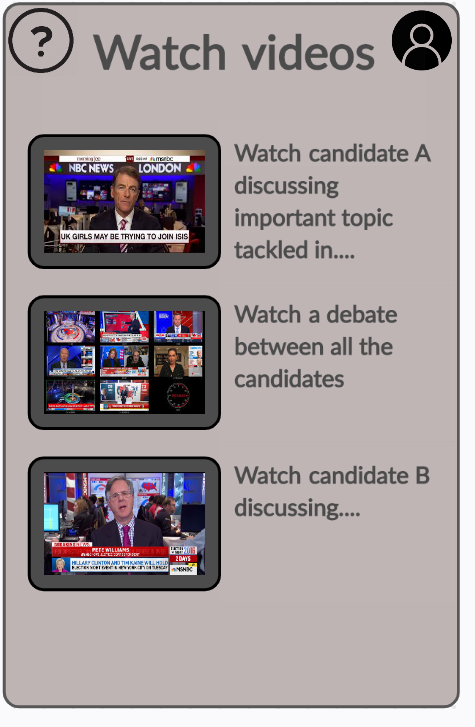
**View Content Interface**



**The view content page includes:**

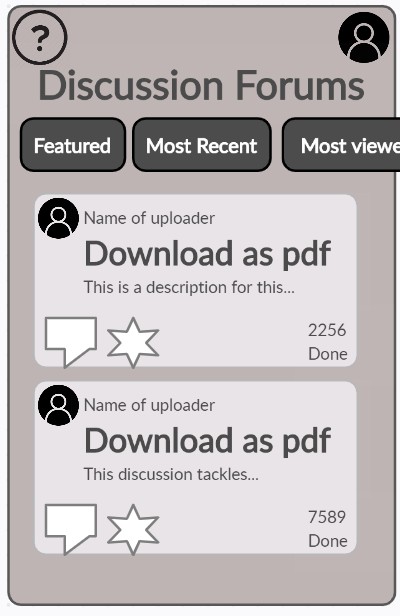
1. **Watch videos button: this button is for the users to watch recorded videos from the campaigners or the candidates aimed for the voters.**
2. **Discussion forums are forums / threads the users submit to generate discussions with other users about a specific topic**
3. **The recorded webinars are either recorded meetings done between the users and the candidates or the recording of previously conducted live sessions.**
4. **Finally requesting a live session talks with the candidates.**

**Watch videos Interface**



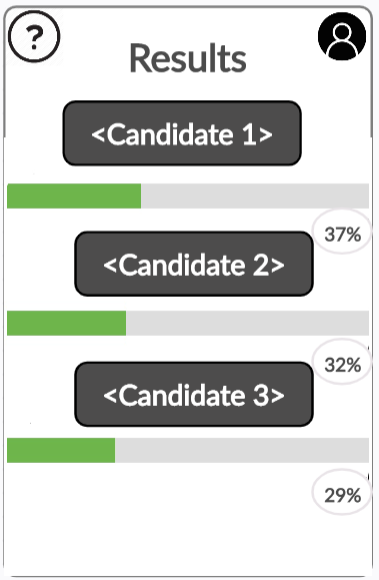
**This interface resembles the page the user will be directed to when they press on ‘watch videos’ in the main menu. It contains videos of various different content about all the candidates and the elections period in general.**

**Discussion Forums Interface**



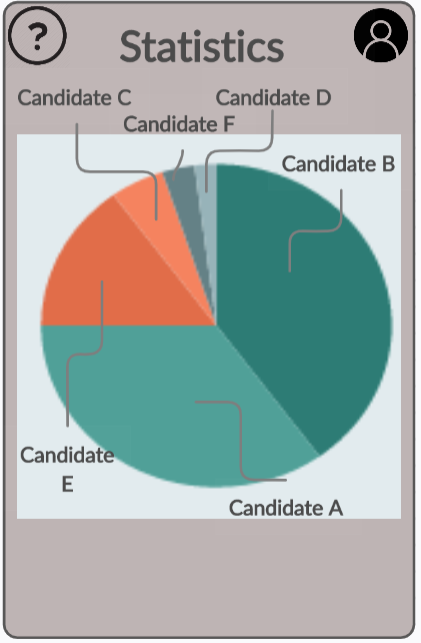
**This interface is the page the user will be directed to when they click on ‘Discussion Forums’ in the main menu page. Simply it allows the user to submit new threads or participate in an existing thread submitted by other users.**

**Results Interface**



**The results page will only be visible during the voting period. It shows the users the percentage of votes for every candidate. The user will be directed to this page from the main menu page during the elections phase.**

**Statistics Interface**



**The statistics page will illustrate the analysis of the numerical data between the candidates. This feature will be visible during the elections period as well.**

**6.3 Screen Objects and Actions**

1. **Login Screen:**
   1. National ID / Mobile number field: this field enables the user to enter their national id or mobile number to login their account
   2. Password field: this field allowed=s the user to verify that this is their account by entering their password.
   3. Security question: this field verifies the credentials of the person as well like the password field
   4. Sign up link: the field is for signing up for whoever doesn’t have an account
   5. Guest user link: this link is for users who are not authorized to participate in the elections, it's just for viewing content.
   6. Change language button: this field allows the user to change language from arabic to english or vice versa for ease of use.
2. **Signup Screen**
   1. First Name and Last name field: allows the user to enter their first and last name
   2. Password field: allows the user to enter their password which must be at least 8 characters with upper cases, lower cases, numbers and characters.
   3. Confirm password field: this allows the user to confirm that they have entered the password they want. In other words it prevents the user from entering something incorrect by fault.
   4. National ID / Passport Number: this field allows the user to enter their personal id / passport number to verify their identity.
   5. Email address field: this field enables the user to enter their email address to receive any news or updates about the elections.
   6. Home address field: this allows the user to enter their home address
   7. Mobile Number: enables the user to enter their phone number.
   8. Security Q/A:enables the user to enter a security question and answer. This question will be asked to be answered during the login phase.
   9. Profile picture attach button link: this link enables the user to attach a document of their profile picture.Tis link will also include other attachment documents that the user will be asked to include.
   10. Change language button: this field allows the user to change language from arabic to english or vice versa for ease of use.
3. **Main menu before voting:**
   1. Help button: this button will direct the user to the help page
   2. Profile button: this button will direct the user to the page which includes all their profile information.
   3. Q/A button: this field allows the user to submit their questions which will be answered by either the candidates or the campaigners.
   4. View content button: This button will direct the users to a page which views content.
   5. Go live button:This button enables the user to ask for live sessions to be conducted.
4. **Main menu after voting:**
   1. Vote Now: this button is only visible in the voting period, it enables the user to vote.
   2. Results and statistics: this button is also only visible in the voting period, it allows the user
   3. Help button: this button will direct the user to the help page
   4. Profile button: this button will direct the user to the page which includes all their profile information.
   5. Q/A button: this field allows the user to submit their questions which will be answered by either the candidates or the campaigners.
   6. View content button: This button will direct the users to a page which views content.
   7. Go live button:This button enables the user to ask for live sessions to be conducted.
5. **Help screen:** 
   1. Report a problem button: this button will enable the user to submit a form of ny technical problem they faced while using the application
   2. How to access button: this button will demonstrate how to access the features in the application
   3. How to use button:this button will contain the list of features and how to use each one of them
   4. Different feature button: This button will explain every feature in the application
   5. FAQs button: this button will include the frequently asked questions about the application from the users.
6. **Profile screen:**
   1. Full name field: the user will find their full name in this field
   2. ID number: the user will find their id number in this field
   3. Mobile number: the user will find their mobile number in this field
   4. Email address: the user will find their email address in this field
   5. Home address: the user will find their home address in this field
   6. Security question: the user will find their security question in this field and in order to access the answer they’ll have to do some security checks like enter their password for example.
   7. Attached documents: this button includes all the attached documents the user has entered.
7. **Change language:**
   1. English language button: this button will allow the user to change language to english.
   2. Arabic button (written in arabic): this button will allow the user to change language of the application to arabic language.
8. **Q/A:**
   1. Text box: the user will be enable to enter their question in the text box
   2. Submit button to allow the user to submit their question
   3. Previous Q/A button to allow the user to access previously answered questions which will make the process faster as a user would want to ask a question that another user has asked before.
9. **View Content:**
   1. Watch videos: this button will direct the user to view content page
   2. Discussion forums: this button will direct the user to discussion threads and forums
   3. Recorded webinars: this button will allow the user to access previously recorded meetings.
   4. Request live scheduled talks: this button will enable the users to request live talk with the candidates or the campaigners.
10. **Watch Videos:**
    1. This page only includes different scattered videos posted by candidates and campaigners
11. **Discussion Forums:**
    1. Help button: this button will direct the user to the help page
    2. Profile button: this button will direct the user to the page which includes all their profile information.
    3. Sorting buttons: this enables the user to access the discussion forums they please of whether featured forums, most recent, most viewed,...etc
12. **Recorded Webinars**:
    1. This page only includes different scattered videos of previously conducted meetings and webinars posted by candidates and campaigners.
13. **Request live sessions**:
    1. This page contains a form asking the user about the name of the candidate they would like to come live.
14. **Results** 
    1. Help button: this button will direct the user to the help page
    2. Profile button: this button will direct the user to the page which includes all their profile information.
    3. No more buttons, the rest are viewing only the results
15. **Statistics**
    1. Help button: this button will direct the user to the help page
    2. Profile button: this button will direct the user to the page which includes all their profile information.
    3. No more buttons, the rest are viewing only the statistics.

**7. REQUIREMENTS MATRIX**

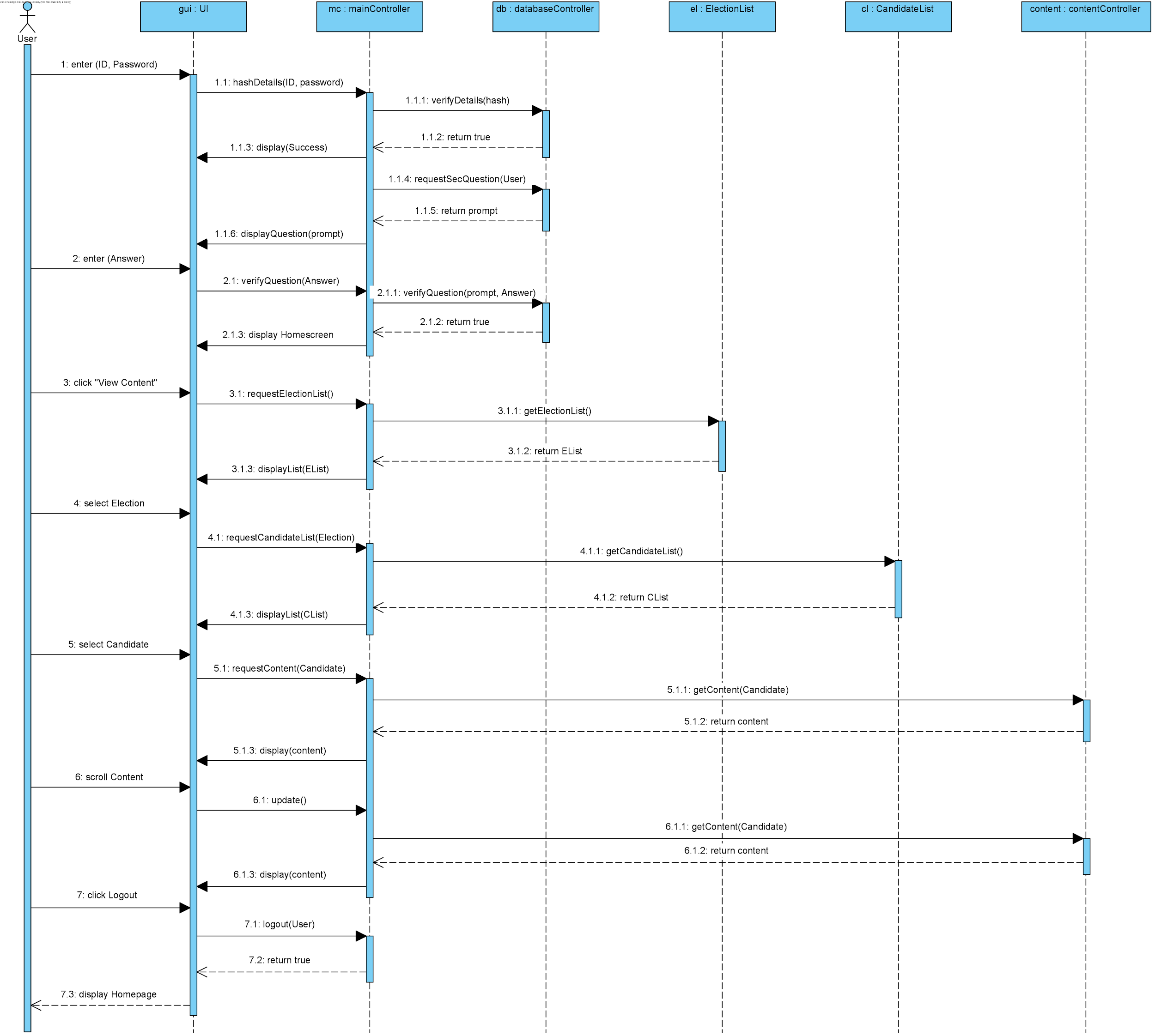
**SRS Functional Requirements Reference:**

|  |  |
| --- | --- |
| **Requirement** | **Components/Data Structure** |
| 4.1/REQ-1 | **Verify User Information** |
| 4.1/REQ-2 | **Send Mobile Verification Code** |
| 4.1/REQ-3 | **Send Email Verification Link** |
| 4.1/REQ-4 | **Add User to Register Database** |
| 4.4/REQ-1 | **System Verifies User Information** |
| 4.4/REQ-2 | **System Allows User Access to Voter Privileges** |
| 4.5/REQ-1 | **Allow Voting for Election** |
| 4.5/REQ-2 | **Verify User Information** |
| 4.5/REQ-3 | **Display Election/Candidate Options** |
| 4.5/REQ-4 | **Add Vote to Candidate’s Votes** |
| 4.8/REQ-1 | **Fetching Discussion Thread** |
| 4.8/REQ-2 | **Star Thread / Comment** |
| 4.8/REQ-3 | **Hide Thread** |
| 4.8/REQ-4 | **Hide Comment** |
| 4.18/REQ-1 | **Reporting Bug** |

**8. APPENDICES**

**Sequence Diagrams:**

User Logging in and Viewing Content:

******

*Figure 8.1*

Figure 8.1 highlights the overview of how the components in our system will communicate to provide seamless service to our end users. The UI controller is in constant communication with the mainController, which requests and receives through function calls to the other classes.

The user first logs in by entering his credentials, primarily their ID and password, which the GUI promptly hashes and sends to the mainController for verification. The main controller sends the hashed data to the databaseController, which checks to see if the hash exists with that specific user (allowing no plaintext fallacies or failures to occur).

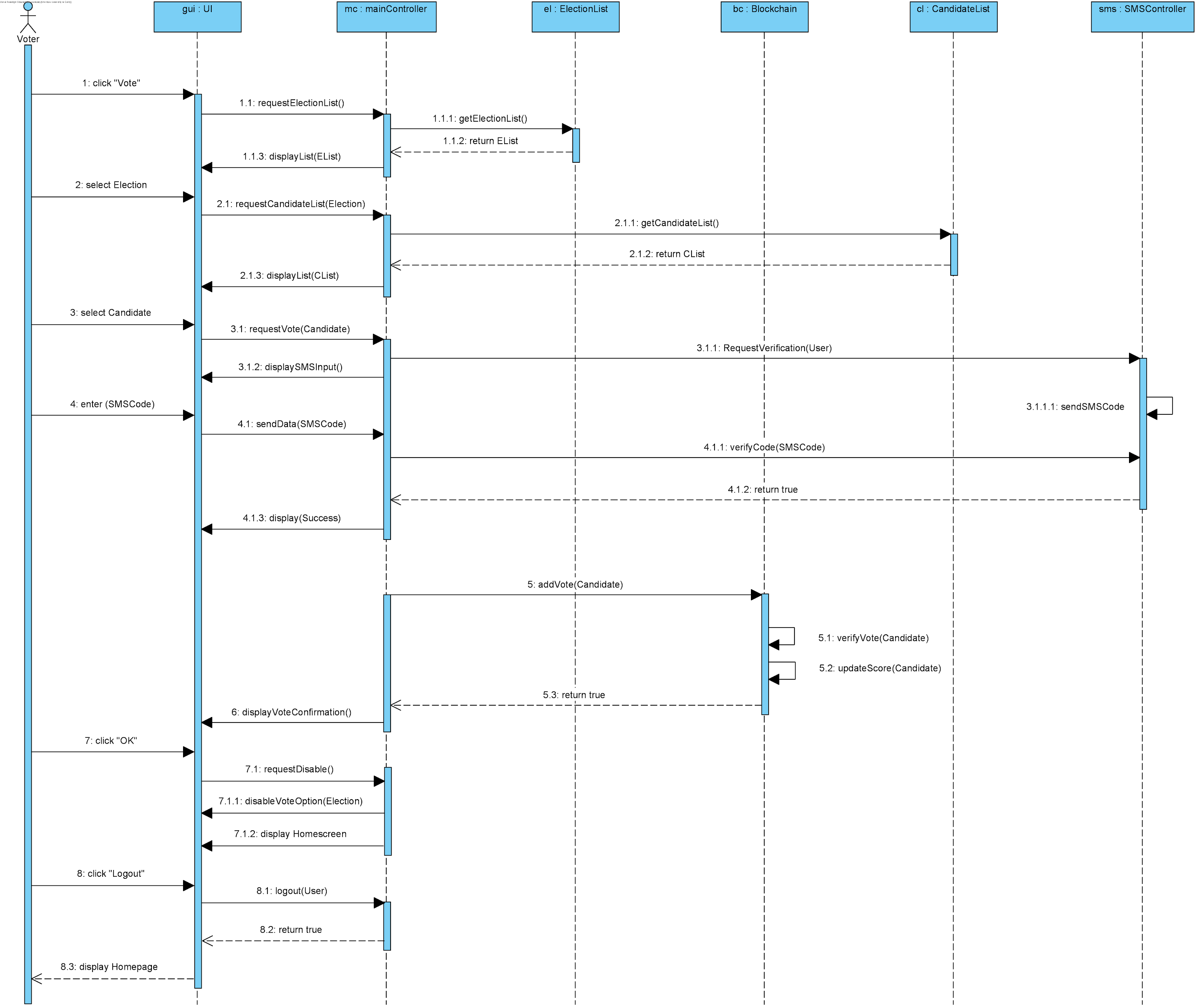
In the example above, we assume that the user’s credentials are correct, so the mainController requests that user’s security question from the databaseController, and the GUI asks the user for the answer to that question in 1.1.6 in the diagram. Assuming that this user answers the question correctly (and disregarding the general SMS confirmation in this case), the GUI displays the home screen for that user as a logged in participant.

Here, in sequence **3**, the user clicks “View Content,” and afterwards a series of connections are made from the GUI to the mainController, requesting and returning both the list of available elections, and the candidates from the selected election. The user then clicks on his preferred filters, and waits for the content to load.

The content in this case **(5.1)** is requested by the GUI, and the mainController fetches it from the contentController, which is in charge of managing posts and their attributes (likes, dislikes, reports, etc.). The content is also displayed in a scroll-like motion, similar to most social media sites like Facebook, Reddit, and Twitter. When the user approaches the end of the screen through scrolling **(6)**, the mainController automatically updates the GUI with updated unseen content from the contentController.

After a period of scrolling, in this example, the user finally decides to logout **(7)**, by which the mainController approves the logout of the user, deletes caches, and requests that the GUI display the original Homepage as a logged out user.

Logged in Voter, Voting for a Candidate:



*Figure 8.2*

In Figure 8.2, we represent a logged in Voter (skipping the login phase previously shown), who wishes to vote in a specific election. The process starts by the user selecting the “Vote” option **(1)**. The GUI and the mainController then proceed to get the details of all available elections currently taking place in the ElectionList **(2)**.

The GUI awaits for the user to select a specific candidate **(3)**, and then requests to vote for that specific candidate through the mainController. Before proceeding with any votes, the mainController requests verification from the SMSController, which sends an SMS code to the user’s mobile phone. In this case **(4)** we assume that the user has entered the proper verification code, so the GUI and mainController send that code to the SMSController to verify it. Once verified the GUI displays a success message (indicating a successful SMS third-party verification.

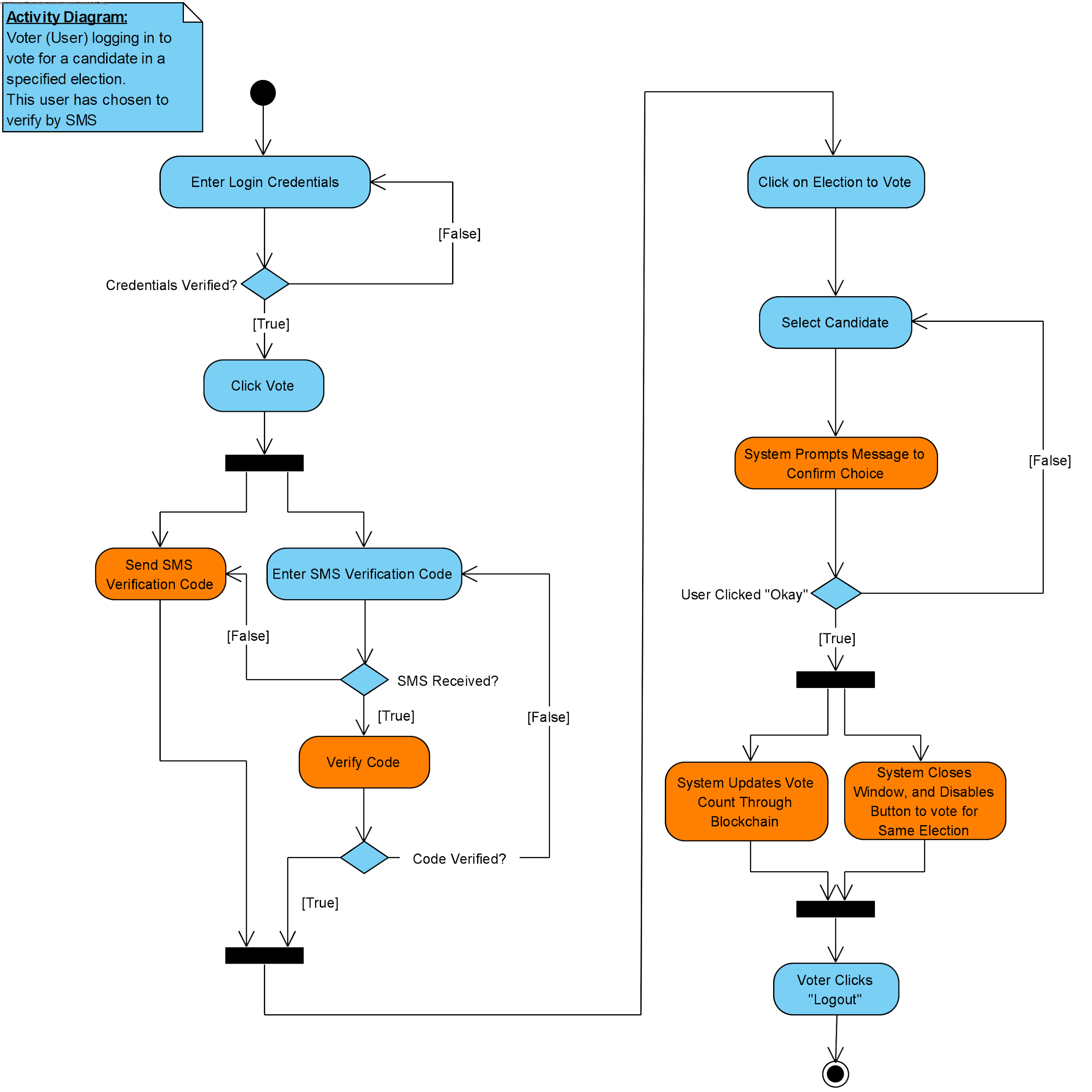
In **(5)**, the mainController then requests to increase the vote of the selected candidate through the Blockchain. Since the Blockchain uses P2P verification, it basically spends time verifying the vote, then increasing the number of received votes for the selected candidate. Notice that this diagram contains no mention of any database controller or centralized database, this is because the entire voting process occurs through the Blockchain. It also proves the loose coupling goal we achieved, and successfully isolates the voting process from any “hackable” database on a network.

Afterwards, the GUI displays that the vote has been confirmed in **(6)** and the user clicks OK in **(7)**. However, this does not complete the voting process, as the mainController also requests to disable the voting option for that specific user in this specific election, and basically marks the user as “voted.” This is done to also further prevent the vulnerability in allowing the same user to vote multiple times.

This sequence diagram generally expresses the voting process for users who are either on the website, mobile application, or even voting at the center. Of course, additional processes are included for center-voting, but those mainly involve procedures that are external to the software system itself (mainly governmental and national laws ensuring physical security).

**Activity Diagrams (Provided by SRS):**

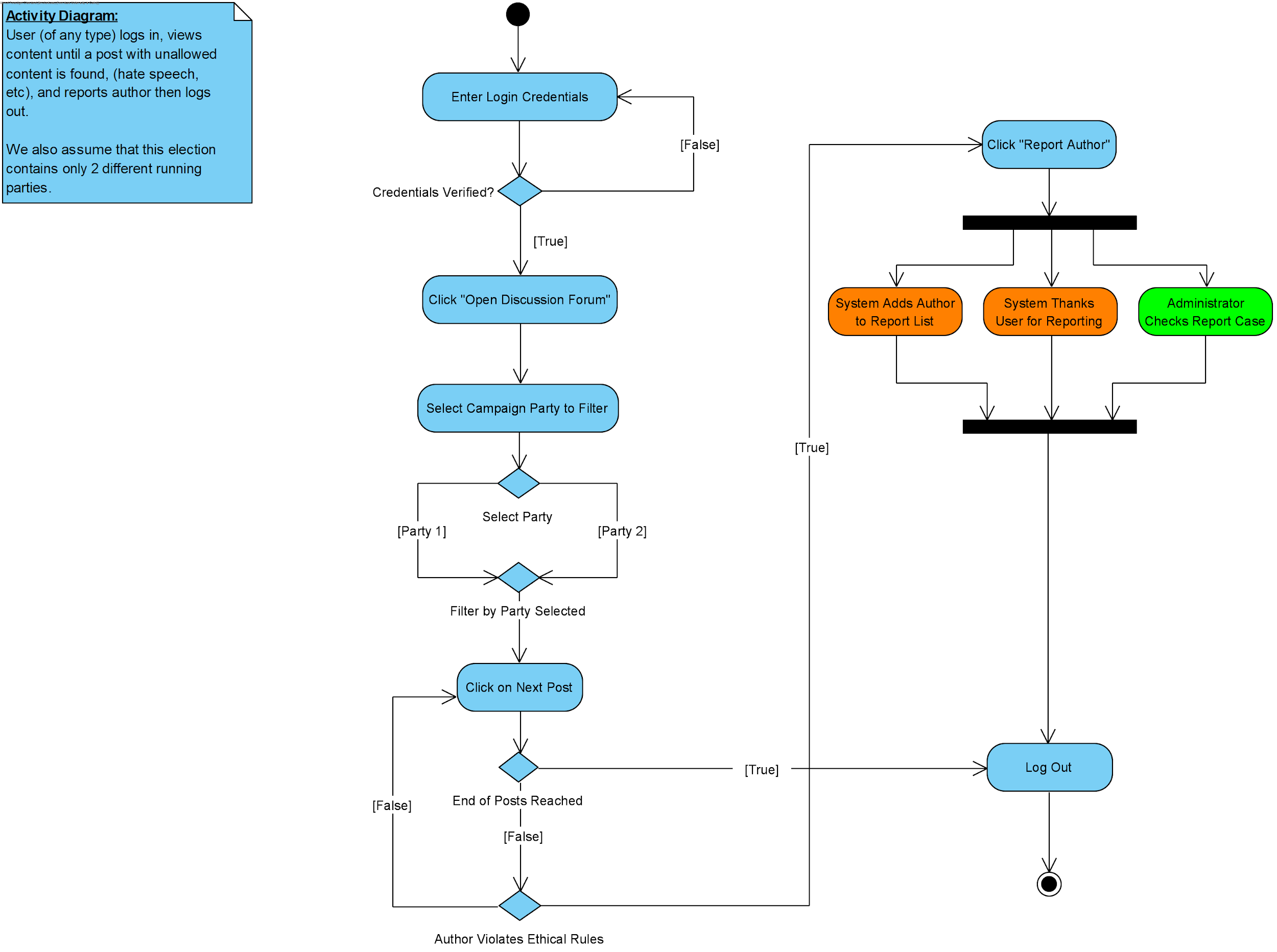
User logging in and voting for a specific candidate:



*Figure 8.3*

This figure was included to provide more context to Figure 8.2, and demonstrate the general overview of how the system behaves in response to a vote being selected.

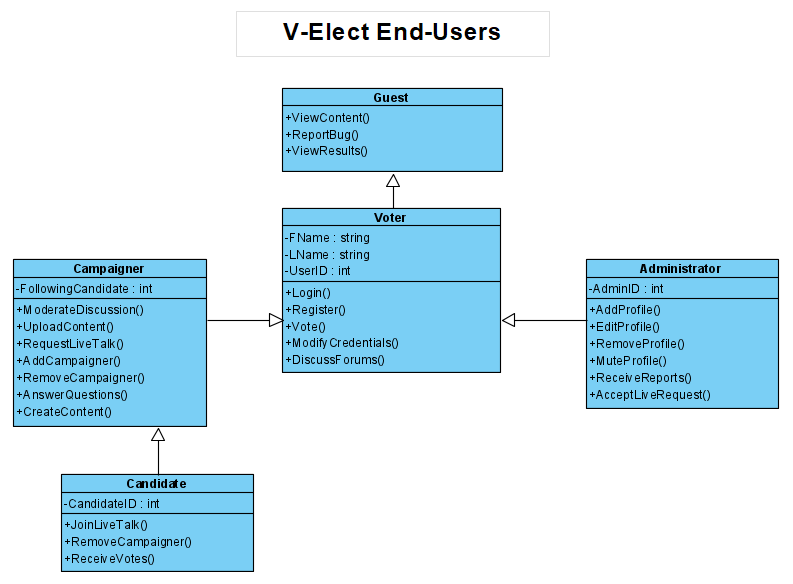
User reporting author of post:



*Figure 8.4*

This figure was included to provide context on the report author function when needed.

**Previous Class Diagram (From SRS):**



*Figure 8.5*

This figure represents a general overview of the User Classes from our SRS document. A more refined and detailed diagram can be found in section 4.