**CS575: Software Design**

**Team Design Package Deliverable**

**Team Name: DirectDesign**

**Team Members:**

1.First Name Brant Last Name Stannert

Email bss54@drexel.edu

2. First Name Kevin Last Name Burdick

Email kmb474@drexel.edu

3. First Name Robert Last Name Davis

Email rjd79@drexel.edu

4. First Name Stephen Last Name Slaughter

Email sws26@drexel.edu

**Project Name: DirectElect**

**A. Introduction**

A.1 Project Overview

The DirectDesign team will develop a Service Oriented Architecture for a web-based electronic election voting system. The design will allow security policies, technologies, and changes in the voting procedures using service components which provide the functionality needed by voters and administrators.

The system will allow account management for both system administrators and voters, and voters will be able to select their ballot and vote securely using security certificates associated with the voters’ identification card. Finally, the system will allow administrators to create reports that voters will be able to view at the end of the election.

A.2 Functionality

The functional requirements of this project can be grouped under the following headings:

1. Account Management:

a. Should allow voters to register to vote with their ID-card credentials.

b. Should allow users to log into their accounts with their ID-card credentials.

c. Should allow administrators to log into the administration page.

2. Poll Management:

a. Should allow administrators to add a list of registered voters.

b. Should allow administrators to create an election page.

c. Should allow administrators to add a new ballot.

3. User Voting:

a. Should allow users to view the candidate list for their region.

b. Should allow users to select candidates on their ballot.

c. Should allow users to submit their vote.

d. Should allow users to cast additional ballots.

4. Reports System:

a. Should allow administrators to view election results, reports, and charts.

b. Should allow administrators to publish election results for voters.

c. Should allow users to view election results.

A.3 Non Functional Requirements

**Security:** The system will address two types of security: overall security of the system and specific security for the different functions associated with the polls and voting. Along with user identification cards containing authentication and signature certificates, the security of the system will be addressed using the SOA architecture that provides for new flexibility and interoperability against concerns and threats on multiple levels.

**Web Responsive Design:** DirectElect will support various browser sizes including mobile, tablet, or desktop by using CSS3 media queries and jQuery. Pages will adapt the size of content or change it when needed to allow compatibility with all screen sizes.

**Automated Attack Prevention:** CAPTCHA will be implemented to prevent automated software from logging into the system.

A.4 Off the Shelf Components and Framework

|  |  |
| --- | --- |
| **Component:** | **Rationale:** |
| Java Messaging Services (JMS) | Using the Java Messaging Services (JMS) as middleware allows the system to transmit messages using a publish and subscribe methodology which allows the system communicate. |
| GlassFish | An application server will be implemented to provide services to the application. GlassFish was chosen for this purpose because it is open-source and made for use on the Java EE platform. |
| PostgresSQL | An instantiation of a postgres database will be used to take advantage of the ACID properties. Consistency and durability of database transactions is vital to this system because we would never want to lose a vote. |

A.5 Revision

|  |  |
| --- | --- |
| **Suggestion:** | **Action:** |
| Handle voting in one state. | Admin creates election and pulls candidates and voters based on state. Voters can only vote for elections whose jurisdiction they fall in based on address (town, state, federal). |
| Defend Pub/Sub approach. | The Pub/Sub data pattern approach supports elections by allowing one mailbox per election where all registered voters can publish votes. This mailbox, called a queue in OpenMQ, is created at the start of an election and is removed when the election ends. OpenMQ uses internal queues to ensure that delivery is guaranteed, with specific recovery and retry capability. If an Election Vault service is not available for any reason or the Message Broker goes down, no votes will be lost. Due to the decoupling of publishers and subscribers, multiple instances of subscribers can be dynamically added to parallel process votes to support system peak loads during large elections without any impact to the rest of the system. All other messaging that are a request reply data pattern will be topic based in OpenMQ, where services can write to specific topics for management and statistical type requests. However, these messages are not guaranteed and therefore do not have the high Quality of Service (QoS). |
| Defend allowing ability to vote multiple times. | We based our system off of the Estonian electronic voting system which allows users to submit multiple votes until the election ends in order to limit outside influences (e.g. vote buying, intimidation) since the voter would be able to recast their vote when the influence left. <http://www.vvk.ee/public/dok/Yldkirjeldus-eng.pdf> |

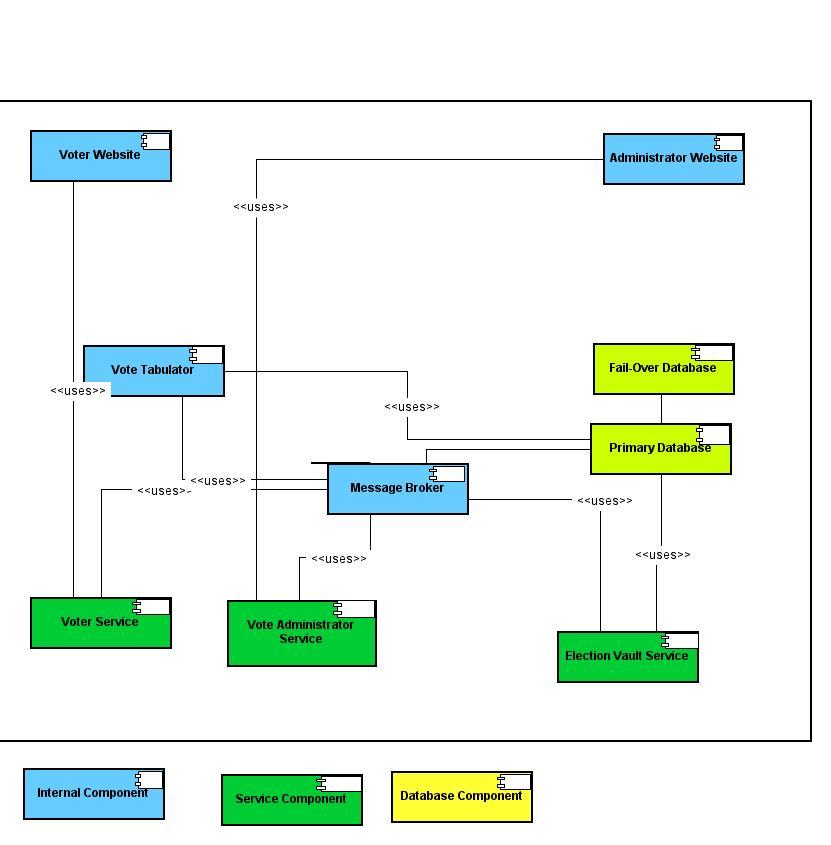
**B. Application Wide Blueprints**

This section describes the overall structure of the application and the interactions that occur between components. A deployment view of the application is also provided.

B.1 Functional Decomposition

This application uses a Service Oriented Architecture (SOA) for the following reasons:

* *Management*: given the short time period we have to develop the demo version of this application, adopting SOA enables each team member to develop a service independently with minimal communication needed between team members. Given the well defined interface of each service , each developer can work on his module alone and in parallel. This will ultimately result in faster development of the entire application.
* *Flexibility*: the loosely-coupled design of each service allows different services with the same interface to be substituted without breaking the design of the application. Also, third party services developed by external organizations can be included in the orchestration which can reduce overall programming effort.
* *Scalability*: the loosely-coupled design between the various services and the Election Vault service through the use of a message broker allow for the potential of multiple instances of Election Vault services in support of elections. This allows the capability to support peak traffic during periods when the election polls are open and elections are ongoing. Each instance can work independently and would require no additional impacts to the rest of system.



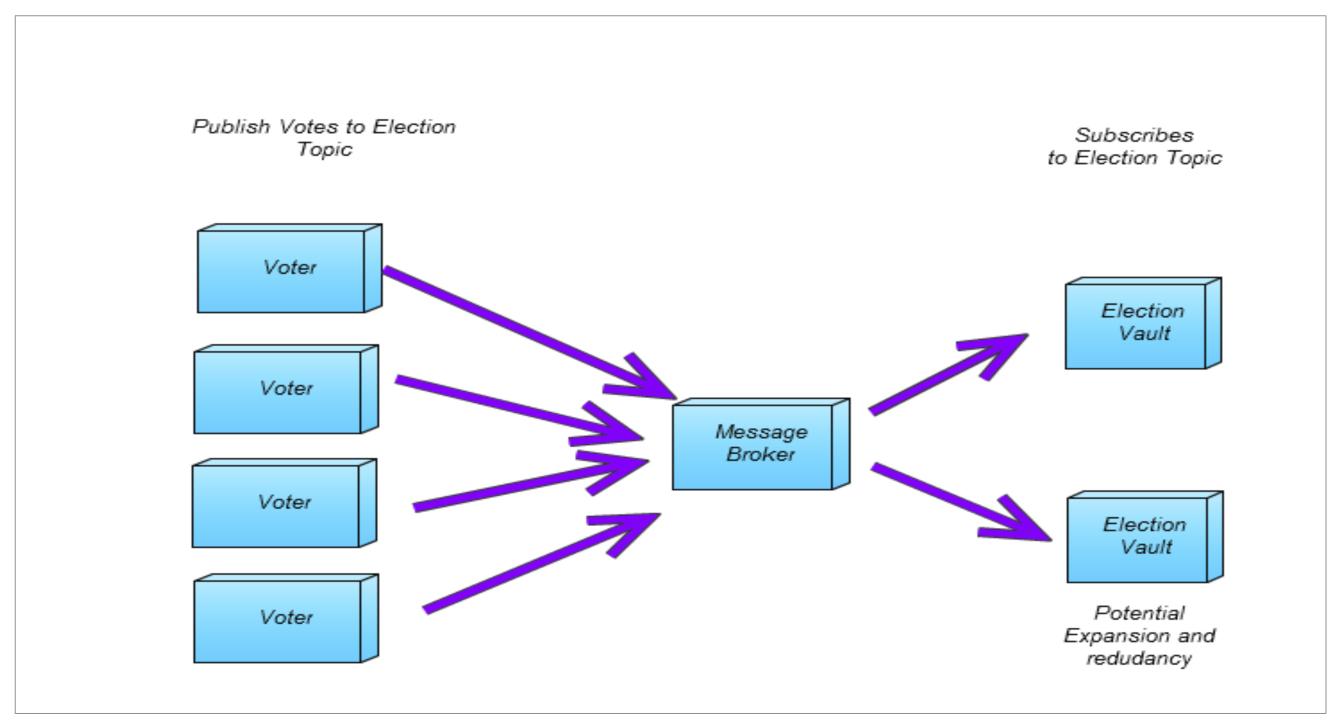
B.1 – 1 Component Diagram: Direct-Elect Web Voting System

|  |  |
| --- | --- |
| **Component:** | **Description:** |
| Online Voter Website | Voters can log into this website with their ID card credentials, view a list of candidates for their region, select candidates, view confirmation of voting, and view election results. This will be implemented using J2EE. |
| Online Vote Administrator Website | Administrators can log into this website with their credentials, create an election page, add a new ballot, add list of registered voters, register to vote, and publish election results. This will be implemented using J2EE. |
| Message Broker | A message queue that brokers the message exchange between publishers and subscribers in a topic-based system. This will be implemented using OpenMQ. |
| Voter Service | This service will allow voters to register for a specific poll, allow voters to vote multiple times as long as the poll is open, accept the voters ID credentials and pass it to the election vault for authorization, and it may support security certificates. It will publish its messages to topics queued at the message broker. This will be implemented using J2EE. |
| Administrator Service | This service shall create polls, create ballots, and create a list of valid voters. It will publish its messages to topics queued at the message broker. This will be implemented using J2EE. |
| Election Vault Service | This service shall provide information on polls, provide continuous backup on the election data, and will ensure that only one vote will count per individual voter. It will subscribe to various topics and receive topic-related messages from the message broker. This will be implemented using J2EE. |
| Primary Database | This database stores all the information about voters, administrators, polls, candidates, and votes. See database diagram B.1-2. This will be implemented using a Postgres database. |
| Fail-Over Database | This database has constant backup information from the primary database and protects the system against data loss. This will be implemented using a Postgres database. |
| Vote Tabulator | This internal component tabulates the votes in the poll. It will be stored on a laptop disconnected from the web and will calculate statistics based on votes stored in the primary database. This will be implemented using J2EE. |

Message Broker

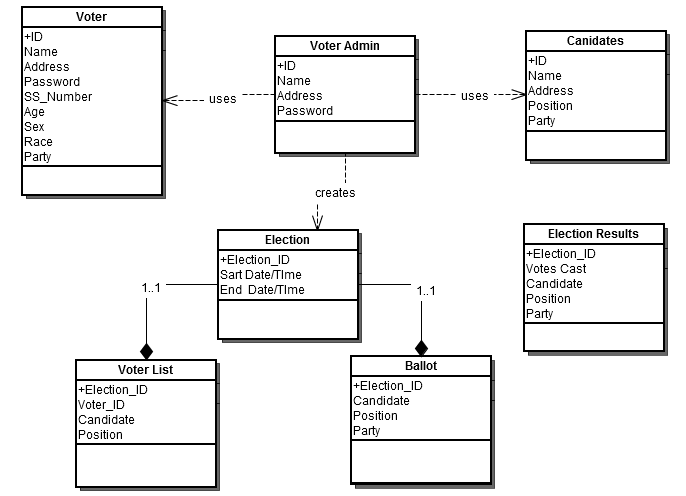
Definition

The Message Broker exists to facilitate message passing to and from various services in the system. It can support message queues when message delivery is to be guaranteed as well as publish subscribe. Request and Reply is the message data pattern that will be primarily used for Service interactions.

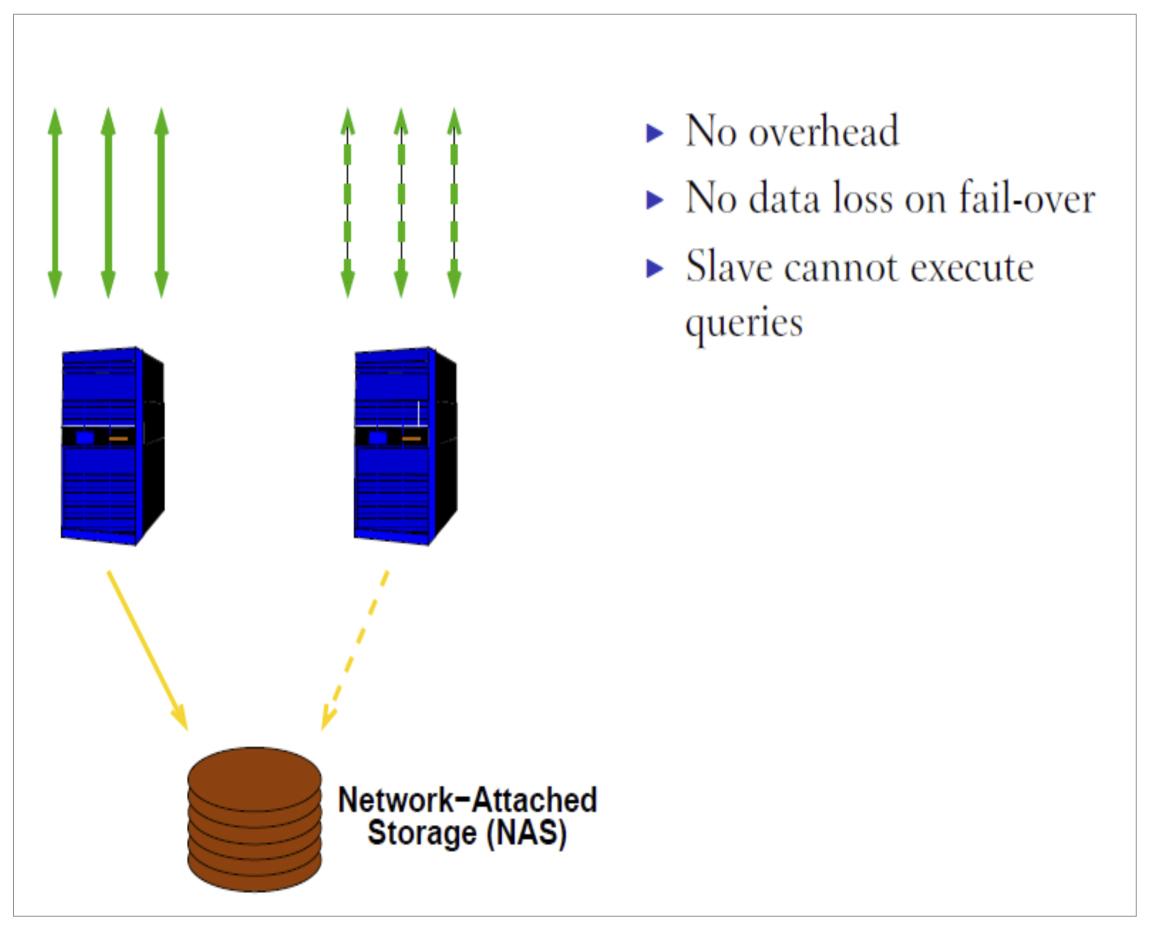


C.1-4 Database

The postgres database provides ACID properties as a guarantee a level of reliability. It also supports redundancy through the use a WAL (write ahead log file) providing switchover with no data loss.



B.1-2 Database Schema: Primary Database

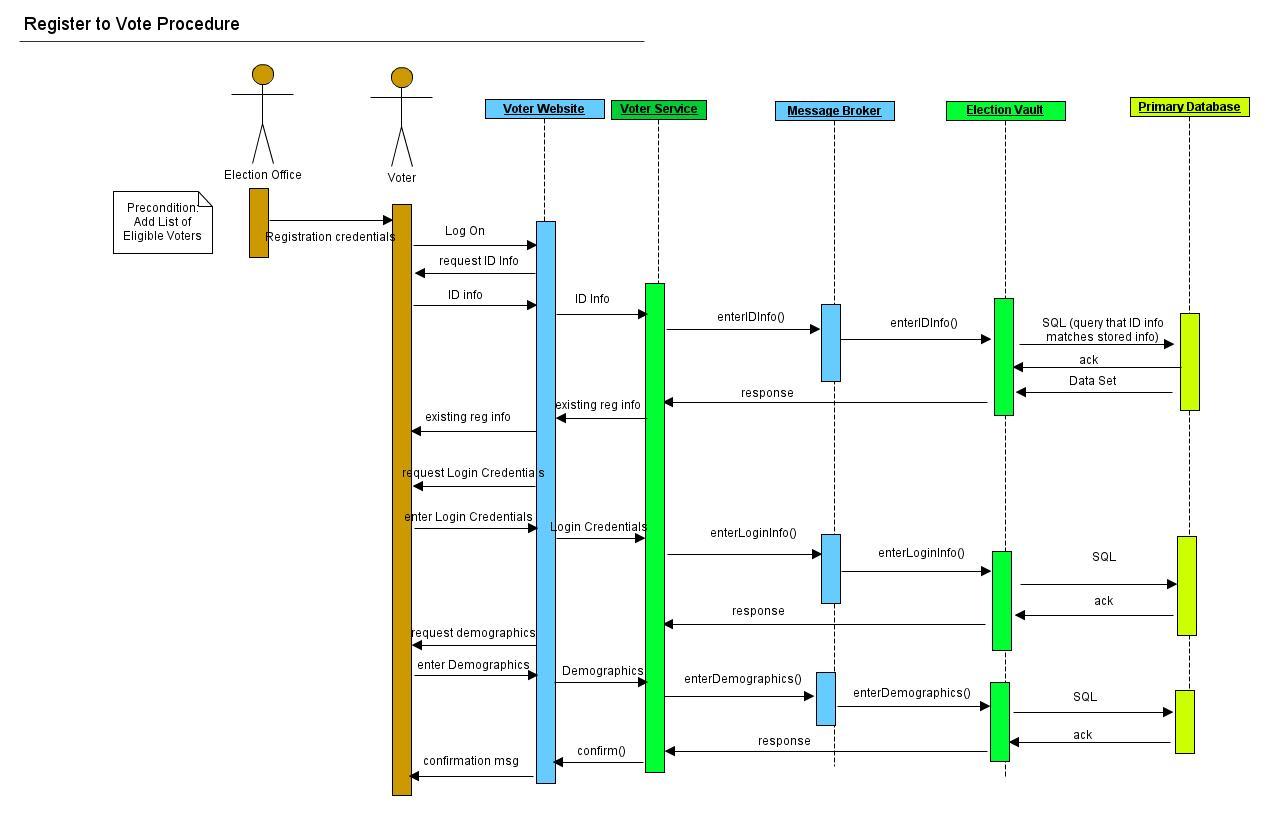


C.1-5 Database redundancy (Shared Disk Failover)

B.2 Behavioral Specification

B.2.1. Voter Registration Procedure.

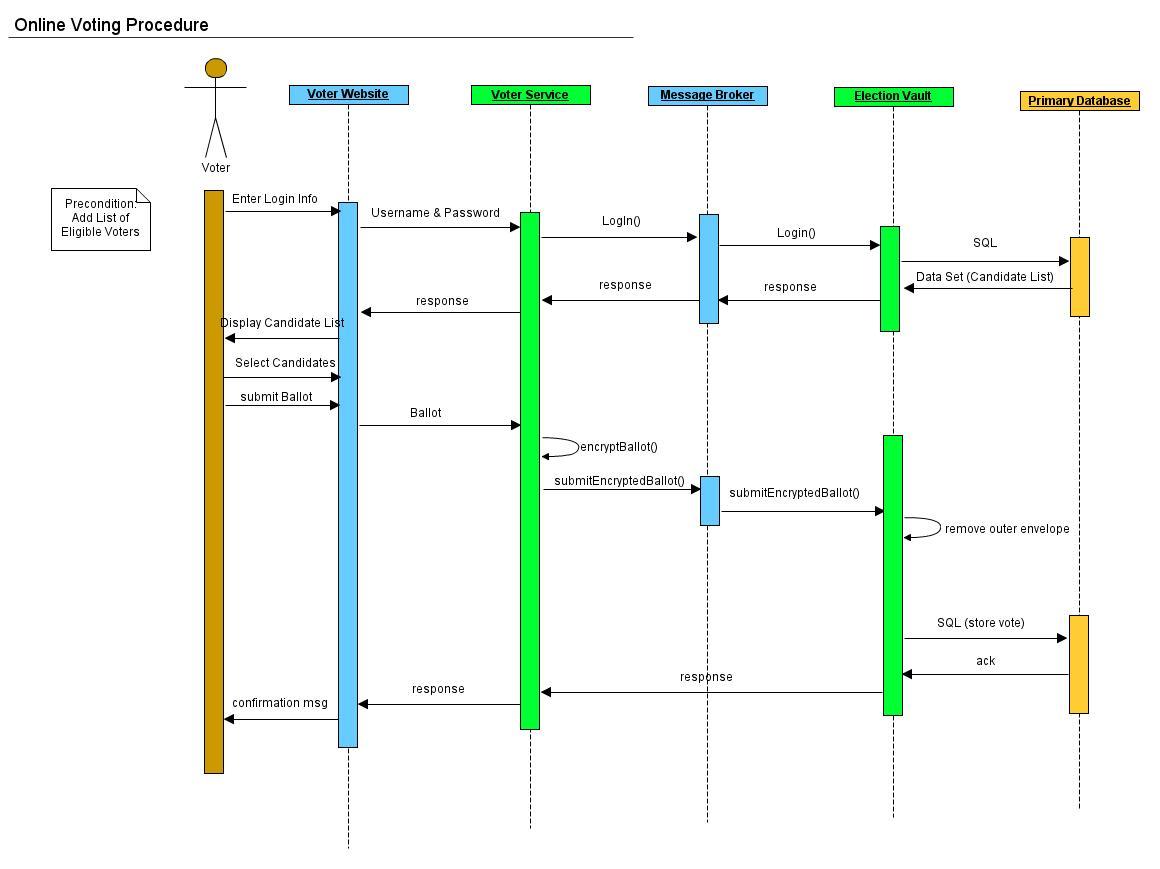
1. Election Office sends Voter registration credential by mail.
2. Logs onto voter website.
3. Enters name, social security number, and address.
4. Creates username and password.
5. Selects party affiliation.
6. Selects demographic information.
7. On successful registration, confirmation is displayed.



B.2.1-1 Sequence Diagram: Voter Registration

B.2.2. Online Voting Procedure

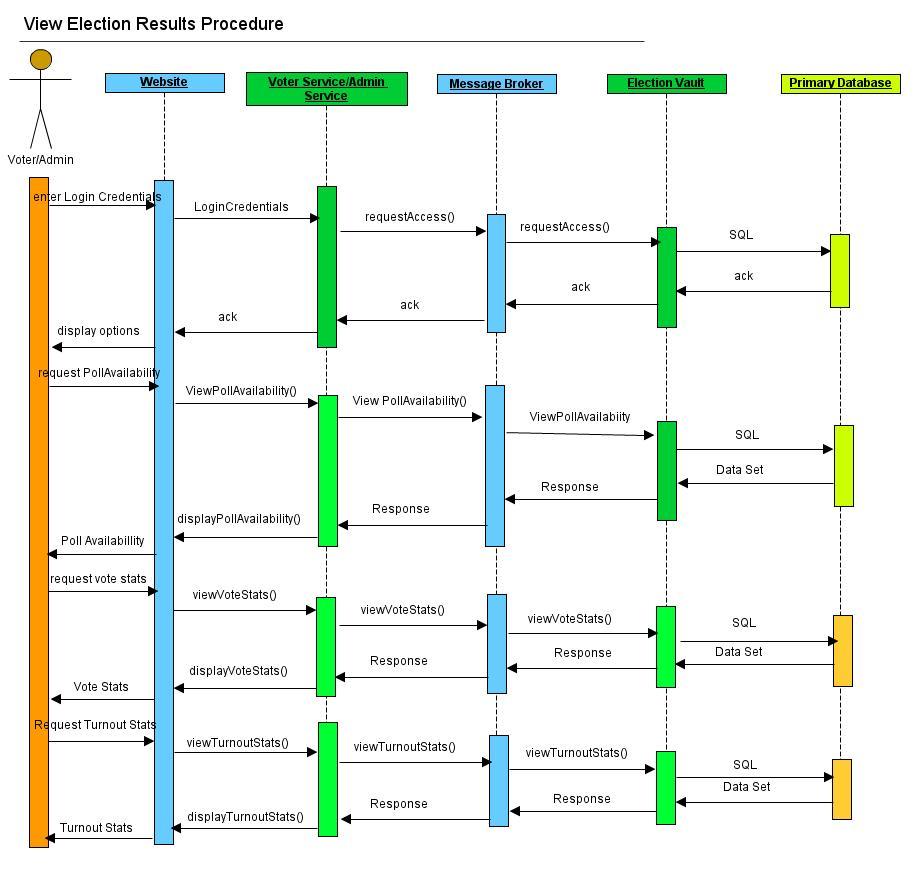
1. Voter logs onto voter website.
2. Views candidate list for Region.
3. Selects candidates.
4. Submits ballot.
5. Cast additional ballots until election end.
6. On successful vote, confirmation is displayed.

B.2.2-1 Sequence Diagram: Online Voting

B.2.3 View Election Results Procedure

1. Voter/Admin logs onto website.
2. Selects “view election results”

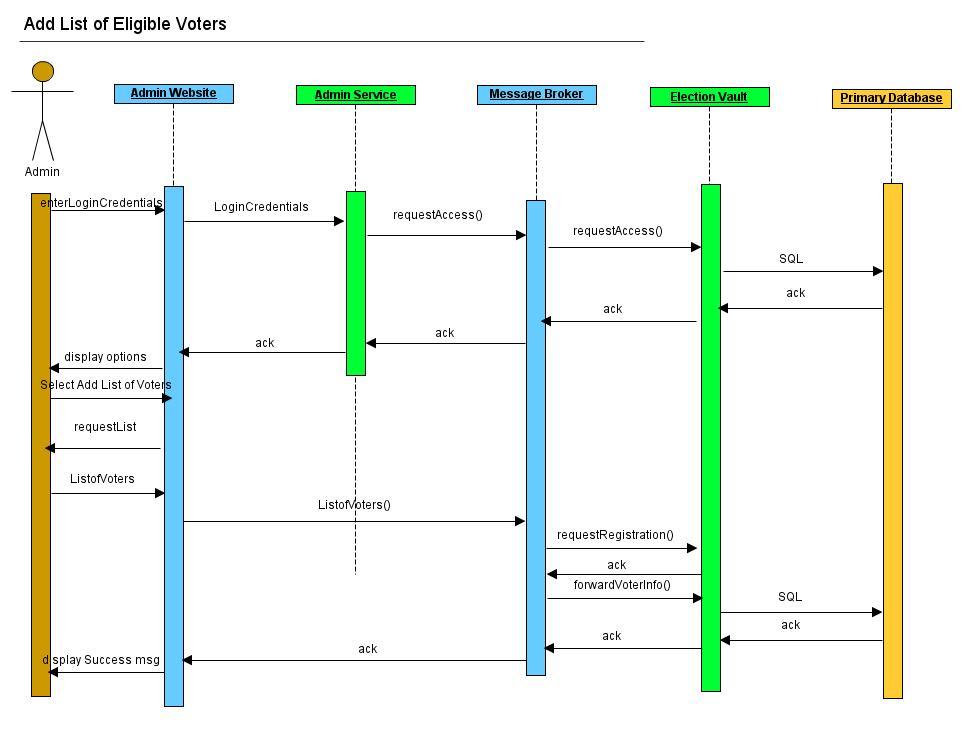
3. View report



B.2.3-1 Sequence Diagram: View Election Results

B.2.4 Add List of Voters

1. Administrator logs onto Admin website
2. Enters list of Voters and their socia.
3. On successful list addition, confirmation is displayed.



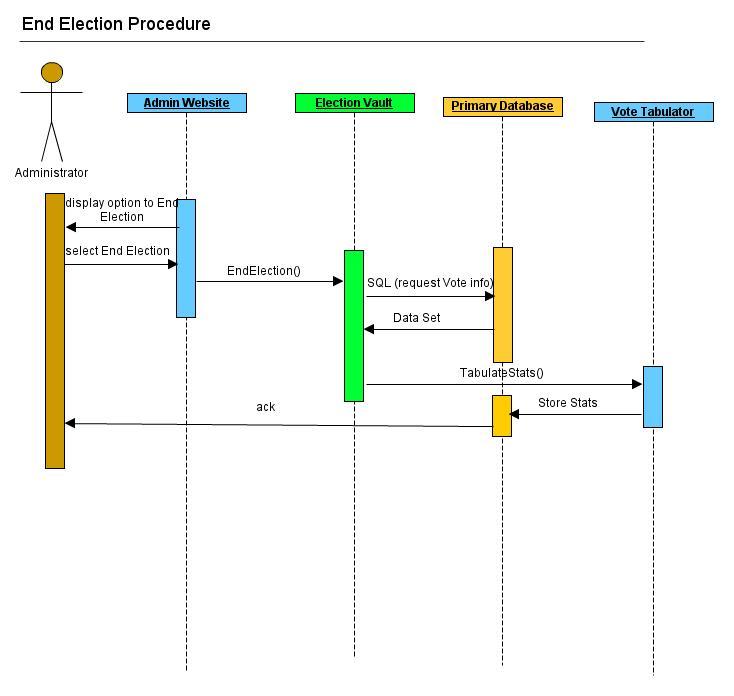
B.2.4-1 Sequence Diagram: Add list of Eligible Voters Procedure

B.2.5 End Election Procedure

1. Administrator logs onto website

2. Selects End Election

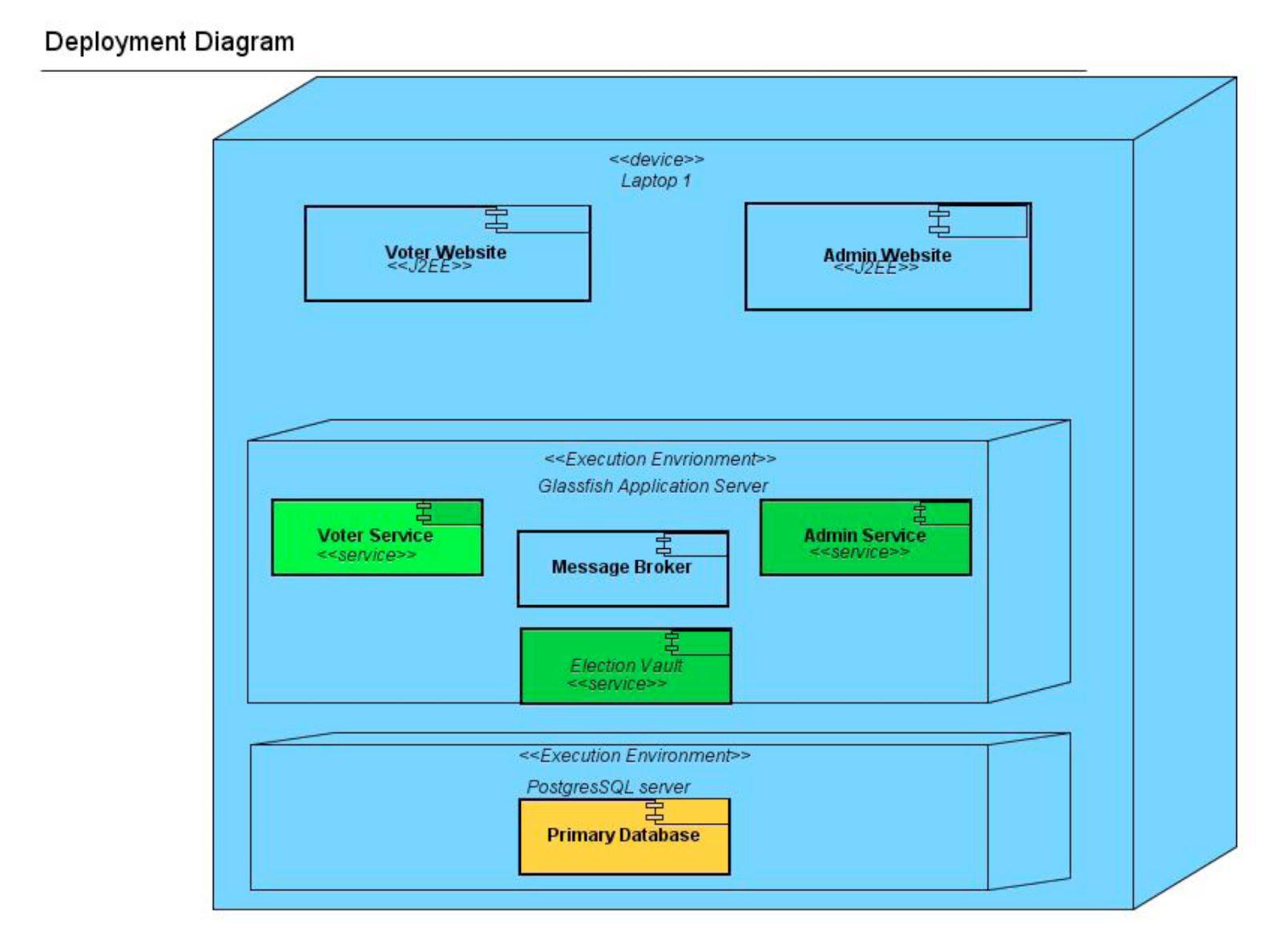
3. On Successful end of election, confirmation message is displayed.



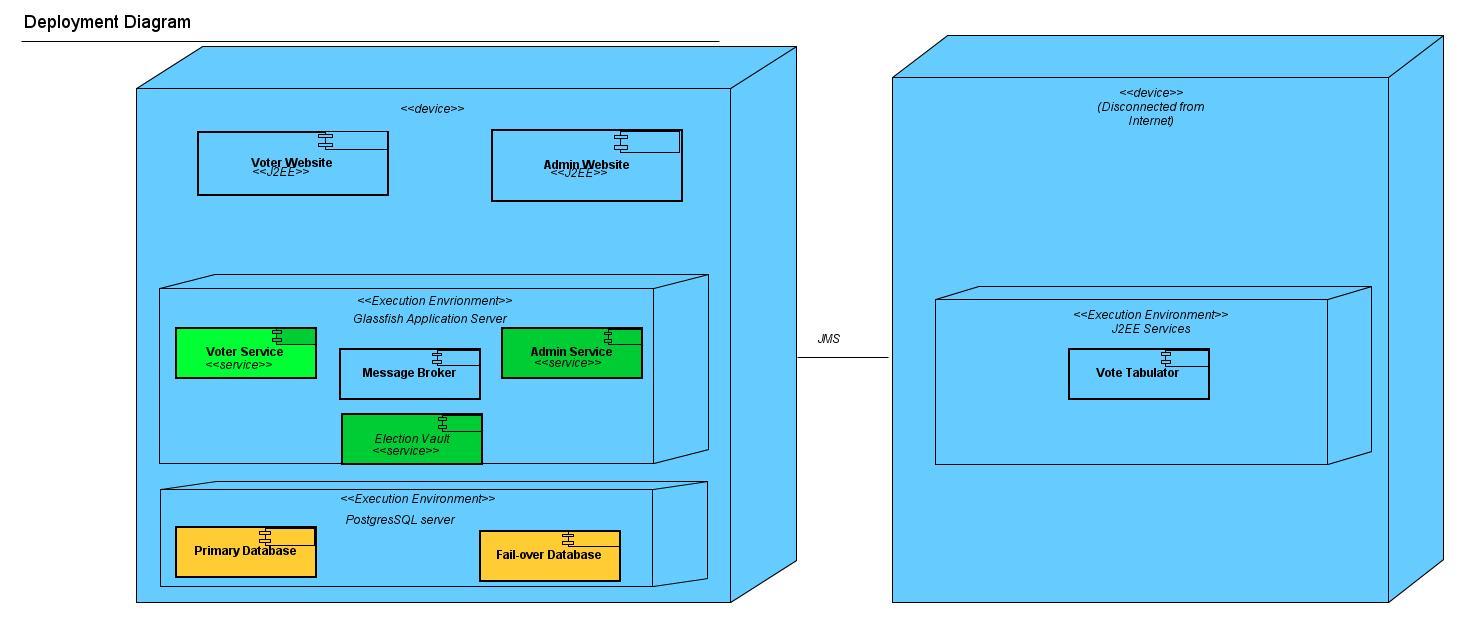
B.2.5-1 Sequence Diagram: End Election Procedure

B.3 Runtime Packaging

Diagram B.3-1 shows the deployment of our demo-run. Diagram B.3-2 shows the deployment configuration of our full-scale application. The two laptops are connected by a LAN cable. One laptop will contain the electronic voting system and services, the other will contain only the vote tabulator service in isolation. The vote tabulator is isolated from the internet and contains the private key needed to decrypt votes and run statistics on them. The purpose of this isolation is to prevent tampering with vote tabulation by intruders.



B.3-1 Deployment Diagram: Demo-run



B.3-2 Deployment Diagram: Full-Scale Version

**C. Service Blueprints**

The following is a list of Services that support the DirectElect System. For development the Contract first method will be used.

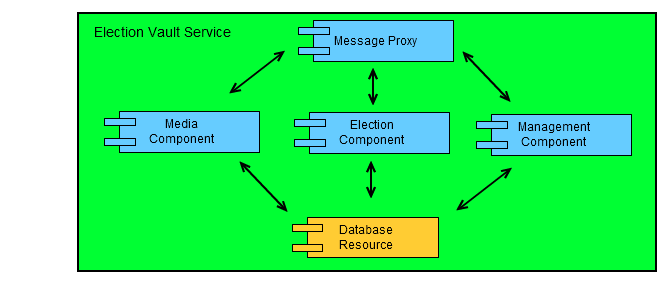
C.1 Election Vault Service

C.1-1 Definition

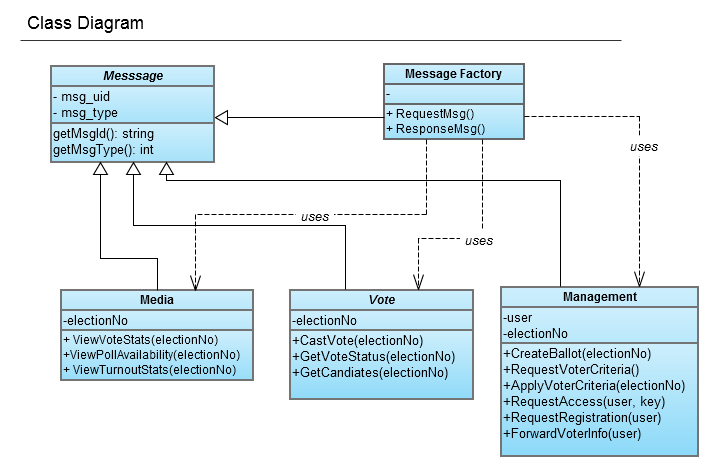
This Election Vault Service is a Component Service that is the repository for all information on active elections and support information on future elections including the lists of candidates, voters, and voter administrators. This service records votes on all the active elections. This component service type is a basic style that encapsulates the application capability in more than one component. The messaging model used by this Service will be Publish Subscribe to handle the Voting transactions during an Election and Request Reply for all other transactions.

C.1-2 Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface:** | **Direction:** | **Use Case(s):** | **Gist** |
| createBallot | Input | Add a new Ballot | Requests a new ballot be generated |
| storeBallot | Output | Add a new Ballot | Provides information on the new generated Ballot |
| submitVote | Input | Submit Ballot | Provides vote from a specific voter |
| getVoterStatus | Input | View Voter Confirmation | Requests status as to whether the voter has casted a vote and that it has been recorded in an election |
| voterStatus | Output | View Voter Confirmation | Provides status as to whether the voter has casted a vote and that it has been recorded in an election |
| getCandidates | Input | View Candidate List | Requests list of Candidates for a specific Election |
| Candidates | Output | View Candidate List | Provides list of Candidates for a specific Election |
| requestVoterCriteria | Input | Add list of Registered Voters | Request list of voter criteria for selecting list of register voters for a specific election |
| voterCriteria | Output | Add list of Registered Voters | Provides list of voter criteria for selecting list of register voters for a specific election |
| forwardVoterCriteria | Input | Add list of Registered Voters | Provides list of voter criteria for filtering list of register voters for a specific election |
| requestRegistration | Input | Register to Vote | Requests registration of the voter in the System |
| forwardVoterInfo | Input | Register to Vote | Provides Voter information to the system for registration |
| requestAccess | Input | Login using ID Card  Login Admin | Request and establishes access to Election Vault |
| Ack/nack | Output | Submit Ballot  Add list of registered voters  Register to Vote  Login Voter  Login Admin | Provides confirmation or rejection of request |



C.1-3 Internal Component Diagram



C.1-4 Class Diagram

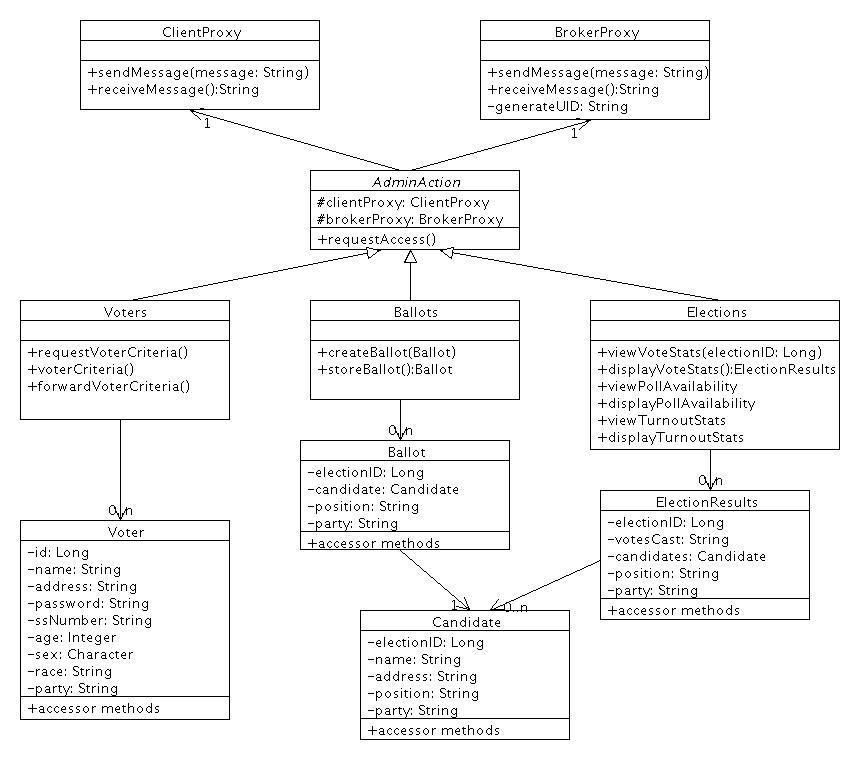
C.2 Vote Administrator Service

C.2-1 Definition

The Vote Administrator Service exists to facilitate the generation of specific Elections, including the ballots, the candidates on the ballots, when the Polls are open, and the list of valid voters. This component service type is a basic style that encapsulates the application capability in a single component.

C.2-2 Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| **Interface:** | **Direction:** | **Use Case(s):** | **Gist:** |
| createBallot | Output | Add a new Ballot | Requests a new ballot be generated |
| storeBallot | Input | Add a new Ballot | Provides information on the new generated Ballot |
| requestVoterCriteria | Output | Add list of Registered Voters | Request list of voter criteria for selecting list of register voters for a specific election |
| voterCriteria | Input | Add list of Registered Voters | Provides list of voter criteria for selecting list of register voters for a specific election |
| forwardVoterCriteria | Output | Add list of Registered Voters | Provides list of voter criteria for filtering list of register voters for a specific election |
| requestAccess | Input | Login Admin | Request and establishes access to Election Vault |
| Ack/nack | Output | Submit Vote  Add list of registered voters  Register to Vote  Login Admin | Provides confirmation or rejection of request |
| viewVoteStats | Input | View Election Results | Requests information on Voter statistics |
| displayVoteStats | Output | View Election Results | Provides information on Voter statistics |
| viewPollAvailability | Input | View Election Results | Requests information on Poll Availability |
| displayPollAvailability | Output | View Election Results | Provides information on Poll Availability |
| viewTurnoutStats | Input | View Election Results | Requests information on Voter Turnout |
| displayTurnoutStats | Output | View Election Results | Provides information on Voter Turnout |



C.2-3 Class Diagram

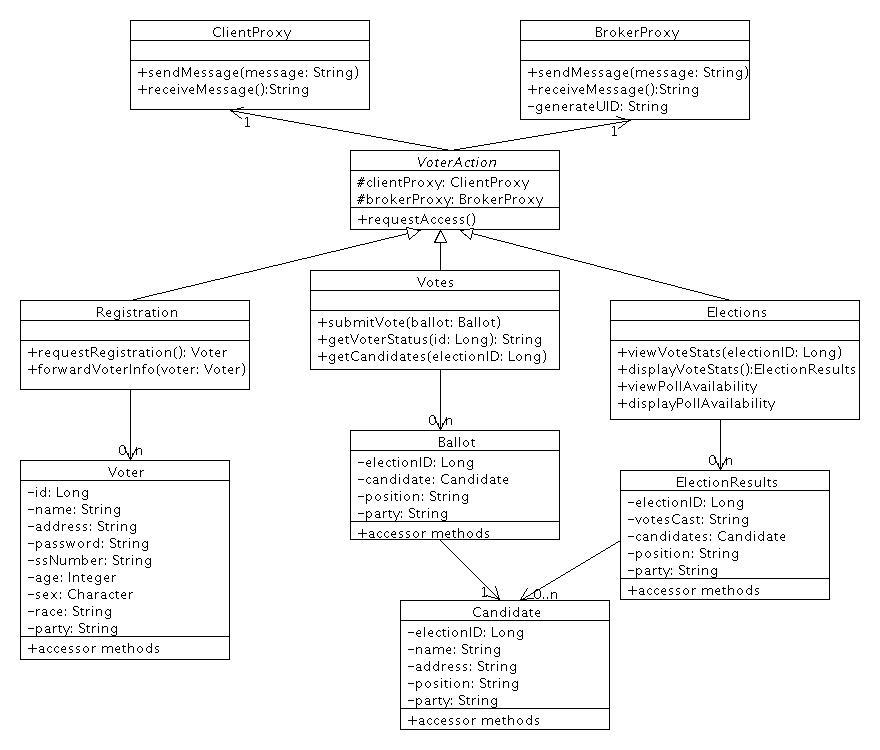
C.3 Voter Service

C.3-1 Definition

The Voter Service exists to facilitate a person the ability to register and vote in specific elections. This component service type is a basic style that encapsulates the application capability in a single component.

C.3-2 Interfaces

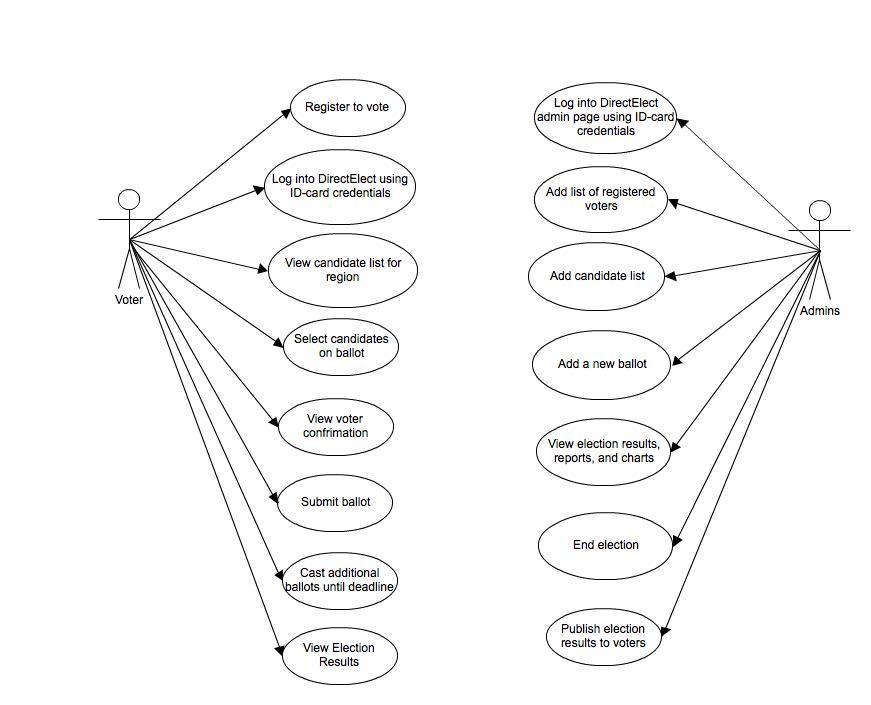
|  |  |  |  |
| --- | --- | --- | --- |
| **Interface:** | **Direction:** | **Use Case(s):** | **Gist:** |
| submitVote | Output | Submit Ballot | Provides vote from a specific voter |
| getVoterStatus | Output | View Voter Confirmation | Requests status as to whether the voter has casted a vote and that it has been recorded in an election |
| voterStatus | Input | View Voter Confirmation | Provides status as to whether the voter has casted a vote and that it has been recorded in an election |
| getCandidates | Output | View Candidate List | Requests list of Candidates for a specific Election |
| Candidates | Input | View Candidate List | Provides list of Candidates for a specific Election |
| requestRegistration | Output | Register to Vote | Requests registration of the voter in the System |
| forwardVoterInfo | Input | Register to Vote | Provides Voter information to the system for registration |
| requestAccess | Output | Login using ID Card  Login Admin | Request and establishes access to Election Vault |
| Ack/nack | Input | Submit Ballot  Register to Vote  Login Voter | Provides confirmation or rejection of request |
| viewVoteStats | Input | View Election Results | Requests information on Voter statistics |
| displayVoteStats | Output | View Election Results | Provides information on Voter statistics |
| viewPollAvailability | Input | View Election Results | Requests information on Poll Availability |
| displayPollAvailability | Output | View Election Results | Provides information on Poll Availability |
| viewTurnoutStats | Input | View Election Results | Requests information on Voter Turnout |
| displayTurnoutStats | Output | View Election Results | Provides information on Voter Turnout |



C.3-3 Class Diagram

**D. User Interface Blueprints**

D.1 Online Voting



D.1-1 Use Case: Online Voting

|  |  |
| --- | --- |
| **Actors** | Voter, Admin |
| Actor | Events |
| C. Voters | A1. The voter registers online to vote before the election starts.  A2. The voter inserts the ID-card into a card reader and opens the webpage for voting.  A3. The voter verifies him/herself by entering his/her username and password.  A4. The server checks if the voter is eligible (using the data from the register list).  A5. The voter is shown the candidate list of the appropriate electoral district.  A6. The voter selects his/her candidates.  A7. The voter submits his/her ballot which is encrypted.  A8. The voter confirms his/her choice with a digital signature (by entering the PIN-code on their ID-card).  A9. The voter receives a notice on the computer screen that the vote has been accepted.  A10. The voter can recast their ballot multiple times until the election closes.  A11. After the election closes the voter views election results. |
| D. Admins | B1. The admin inserts the ID-card into a card reader and opens the webpage for voting.  B2. The admin verifies him/herself using their username and password.  B3. Admin adds a list of eligible voters to the system.  B4. Admin adds a list of candidates to the system.  B5. Admin creates a new election limiting who can vote by specifying the office being run for, district, and/or party involved and specifying when it will end.  B7. Admin publishes election results to voters. |

D.2 Voter Interface

The voter interface is implemented as a web based J2EE application. As shown in Diagram D.2-1, it starts from Log In Page (A). (A) will require that the voter has their ID-card inserted into a card reader. If they voter has not registered yet, they will navigate to (H) and be asked to create a username and password. Once the voter is registered, they will enter their username and password and then hit submit. If the voter’s credentials match those in the database, they will be redirected to the Voter Home Page (B).

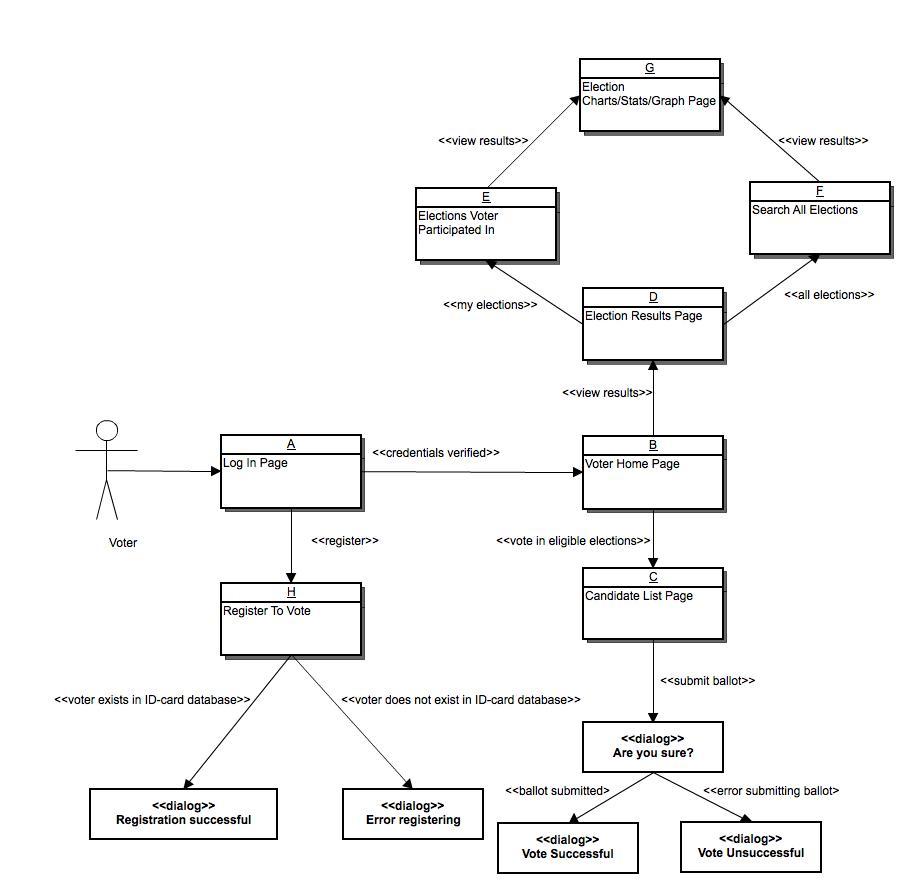
Once at (B), voters can either view results of previous elections or vote in open elections in their region.

Voters that choose to vote in open elections in their region will be directed to the Candidate List Page (C) from (B). (C) returns a list of candidates running in all open elections in the voter’s region. Once the voter makes their selections they will submit their vote and be prompted to confirm their selection by entering PIN 2 from their ID-card. The voter will receive a message saying that their vote was successful or their was an error.

Voters that choose to view previous election results will be directed to the Election Results Page (D) from (B). (D) will allow voters to either view results from elections that the Voter participated in (E) or search through all election results (F).

If the voter chooses to view their election history (E) they will be shown a list of all elections that they voted in. After selecting an item from the list, they will be redirected to the Election Charts/Graphs/Stats page (G) which will display all available information available on the election.

If the voter chooses to search through all closed elections they will be taken to a search page where they can filter results (F). After selecting an item from the list, they will be redirected to the Election Charts/Graphs/Stats page (G) which will display all available information on the election.



D.2-1 Voter UI Diagram

D.3 Admin Interface

The admin interface is implemented as a web based J2EE application. As shown in Diagram D.2-2, it starts from Log In Page (A). (A) will require that the admin has their ID-card inserted into a card reader. Admins will enter their username and password and be directed to the Admin Home Page (B) after their credentials are verified.

Once at (B) admins can add registered voters to the voter list, add candidates to the candidate list, create a new election, or search through all elections.

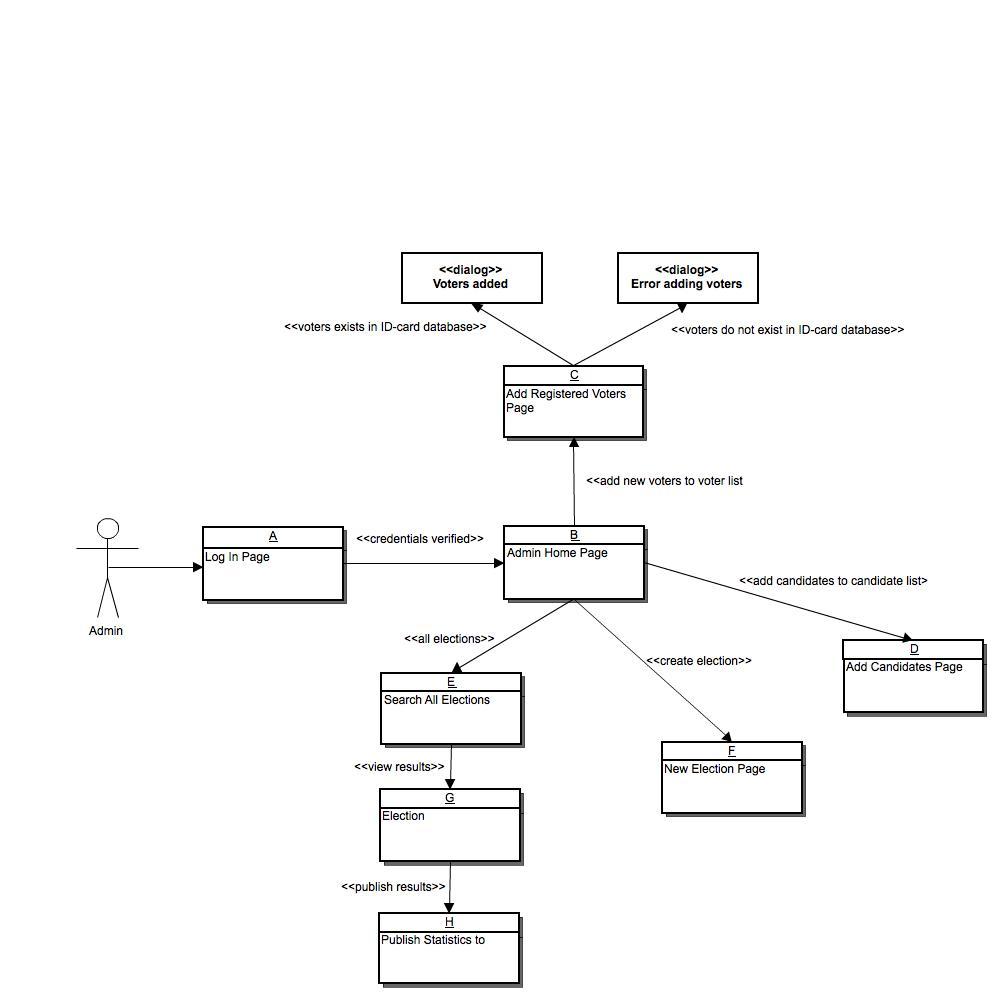
Admins that choose to add new voters will be directed to (C). Once there they will add information for the voters they are registering and submit the form. They will either receive a “Voters added” or “Error adding voters” message depending on whether the voters exist in the ID-card database.

Admins that choose to add new candidates will be directed to (D) where they can enter information about the candidates and submit it to the candidate list.

Admins that choose to create a new election will be directed to (F) where they can enter information about the election such as the office being run for, parties involved, and districts that will be eligible. Admins also need to specify when the election it will end. After the end date passes, the votes will be calculated and statistics will be available to the admin.

Admins that choose to search for elections will be directed to the Search Elections Page (E). Here admins can filter all elections based on office, district, date, and/or party involved. Once an admin selects an election they can view results by navigating to the Election Results/Charts/Graphs Page (G).

(G) will show the user information about the election if it has already ended such as results and demographics. The admin will have the option of publishing the results of an ended election to the voter by selecting a button (H).



D.3-1 Admin UI Diagram