Super Voters 

**Software Quality Report**

Version 1.0

4/1/2022

David Kelly

Garhgaj Singh

Jonathan Coronado

Ya Wang

# Table Of Contents

[**1. Project Scope**](#_cj8x2kywbyoe) **2**

[**2. Improved Software Architecture Design**](#_8dtsurfgoid4) **2**

[**3. Improved UML Class Diagram**](#_tgwiff1lzpxk) **4**

[**4. Verification And Validation Results**](#_sj24fgirish3) **4**

[4.1 Unit Testing Results](#_7naqljczv2k3) 4

[4.2 Integration Testing Results](#_udancxk0xx9w) 5

[4.3 Quality Goals And Quality Metrics Results](#_uago6r2zc4ys) 5

[4.3.1 Quality Goal - 1](#_tutzn5hgswk4) 5

[4.3.2 Quality Goal - 2](#_xkkyzvmrcmfr) 7

[4.3.3 Quality Goal - 3](#_1wj1pf3bi273) 8

[4.3.4 Quality Goal - 4](#_xvuvghhoa20t) 8

[4.3.5 Quality Goal - 5](#_phgx1nev1ke5) 9

[4.3.6 Quality Goal - 6](#_7h4ntad37qyf) 9

[4.3.7 Quality Goal - 7](#_3ejno72adfqt) 10

# 

# 

# 1. Project Scope

The scope of our project was to create an application used for Presidential voting in a desktop environment. Where our desktop application allowed users to vote for the candidate they prefer in a presidential election. We were able to successfully create this application but our next step was to examine the quality metrics. To examine our quality metrics we focused on the unit and integration testing for this program. We felt unit testing is a really good choice because team members will get to test individual components of the software and see if certain methods are working.

# 2. Improved Software Architecture Design

The architectural pattern the application has applied is Multi-layered. This pattern contains several parallel layers and each layer performs certain functions for the application.

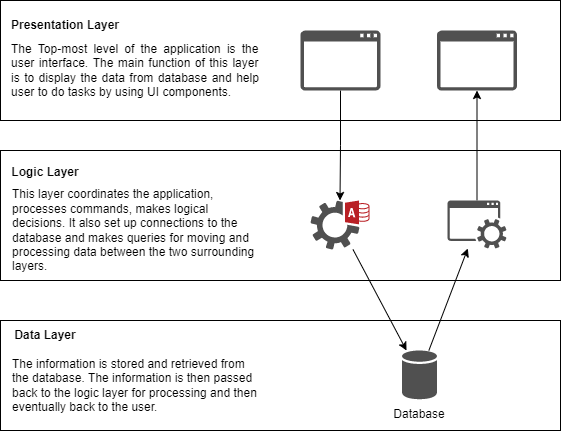


Figure 2.1 Multi-layered Architecture Diagram

# 3. Improved UML Class Diagram

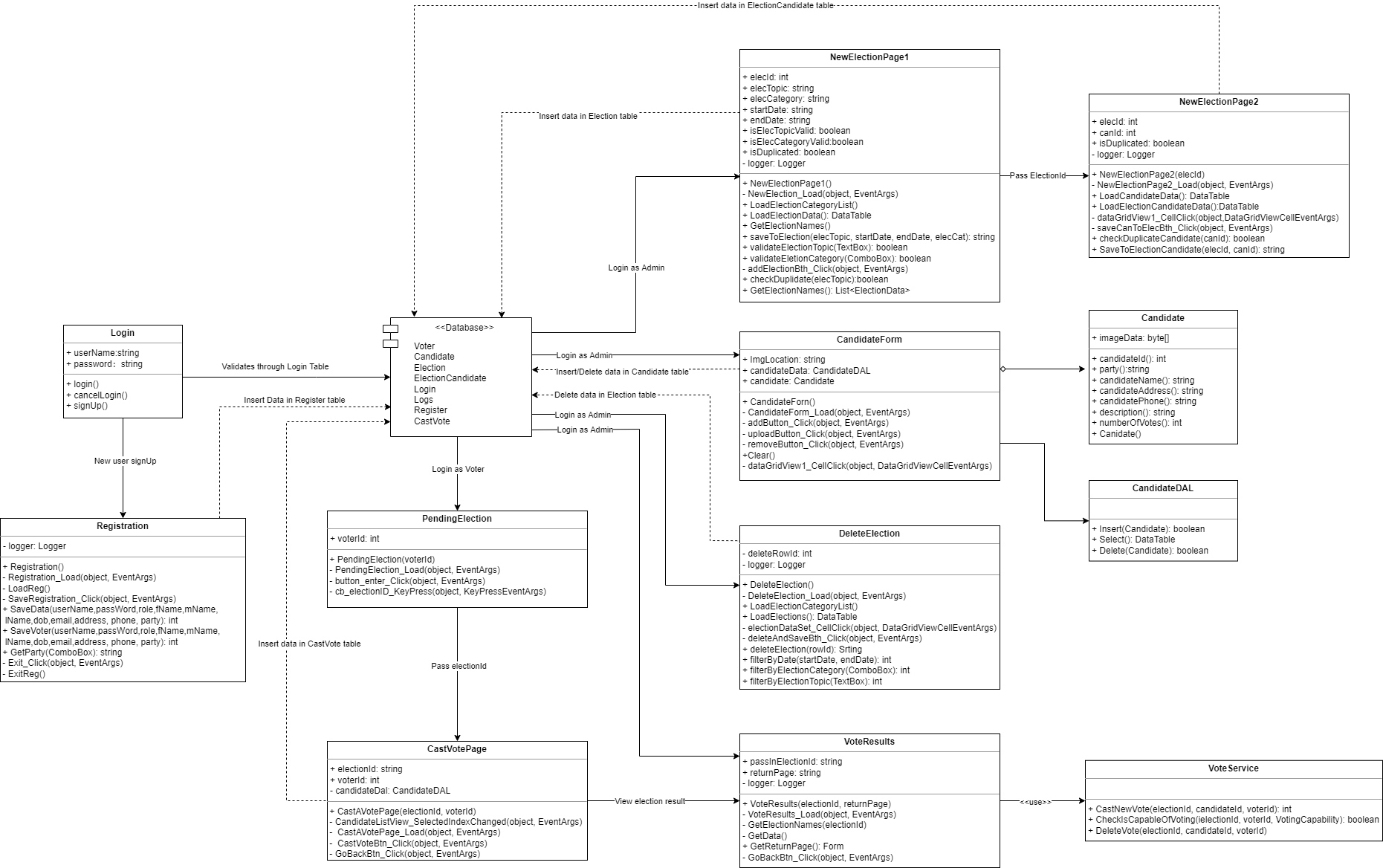


Figure 3.1 Updated UML Class Diagram

# 4. Verification And Validation Results

## 4.1 Unit Testing Results

Our code is separated into many classes and each class has important functions which help to connect our logic together. We went forward to test our classes to validate if components were running properly and to do this we first created a folder in our code project and named it “SuperVoterUnitTests”.The main goal of using unit tests was to verify if our code was working the way we wanted it and if it was matching our design documents. We added test cases to pages such as registration to verify if the user was correctly getting registered properly and if it was returning the correct value or not. We also added test cases for our cast to vote page to see if users were capable of voting and if they were easily able to add or delete the votes they made. We also tested our voter and candidate page to see if the user was able to add/ delete and load information as voters and candidates. Furthermore, we created tests to see if our program could load our election data and if the user was able to select the election category or not. After testing our functions we made sure we were passing all test cases used and if they weren’t, we tried to modify our code in order to get the tests to pass. So after testing our code we were able to pass all of our test cases. We currently have 32 unit test cases and all test cases were passed. The unit test cases can be found here,

<https://github.com/leiawang89/SuperVoters/tree/sprint2/SuperVotersUnitTests>

## 4.2 Integration Testing Results

The aim of integration testing is to test the interactions between two different classes or modules. It focuses on determining the correctness of the interface so as to confirm that the values are passed correctly between different classes. We currently have 5 integration test cases and all test cases were passed. The integration test cases can be found here,

<https://github.com/leiawang89/SuperVoters/tree/sprint2/SuperVotersIntegrationTests>

## 4.3 Quality Goals And Quality Metrics Results

All quality goals and quality metrics discussed in this section are from the [SuperVoter Sofware Quality Plan](https://docs.google.com/document/d/14VdRtQf-3DyTcUuyLWyRHpX_Z82HhH_VpKIAieZ4zp8/edit#) on page 4.

### 4.3.1 Quality Goal - 1

| **Quality Goal** | The application should be user-friendly and a new user should be able to understand how to use it the first time. |
| --- | --- |
| **Quality Metrics** | The application tracks how often the help system is loaded. Based on the usage, we can determine how user-friendly the application is. |
| **Satisfaction Status** | Satisfied |

We made some UI modifications in this milestone in order to meet the user-friendly quality goal. Figure 4.1 is the previous design for the New Election page. However, this design is not satisfied by users while doing the validation test. An administrator should follow two steps in order to successfully add a new election. The first step is to add a new election with a valid election topic, election category, and date range. The second step is to add candidates to the newly added election. This design combined the two stages onto a single page, leading to considerable misunderstanding. Without reading the user instruction book, a new user is unsure where to begin. Furthermore, the previous design contains too many tables, increasing the time it takes for the page to load. If the database grows larger, loading the data into the data grid view will take a lot longer in the future.

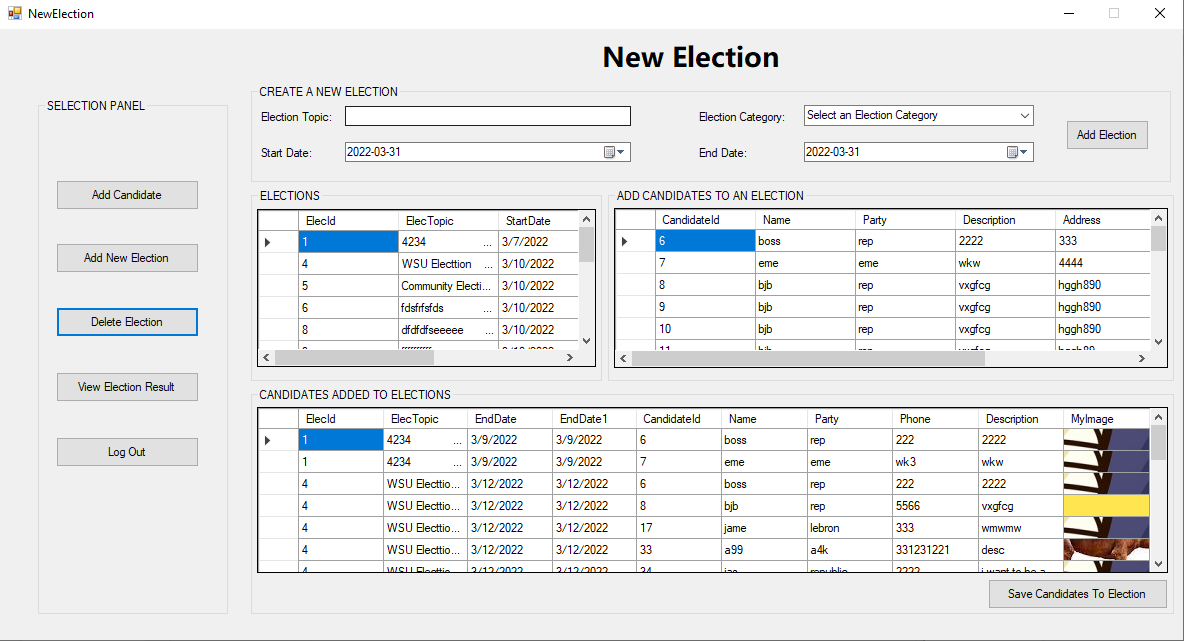


Figure 4.1 NewElection Page Design In Milestone1

Thus, we made some changes to the New Election page. The New Election page was split into two pages. The first page allows users to add a new election. Then the user can select the newly added Election from the drop-down menu and click confirm button for the next step.

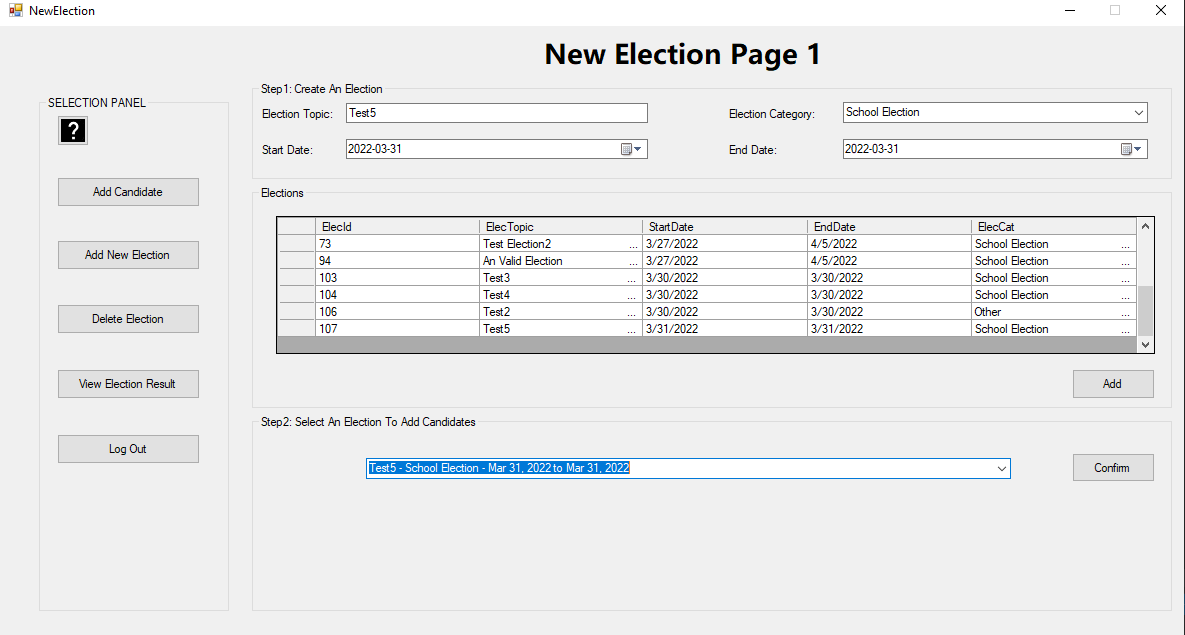


Figure 4.2 Updated NewElection Design(New Election Page1)

The next page is for adding candidates to the currently selected election. Then click Save Candidates To Election button to finish. The new design makes the process of adding elections more straightforward and simple to follow. It also allows users to add candidates to the previously added elections by selecting the election name from the dropdown menu.

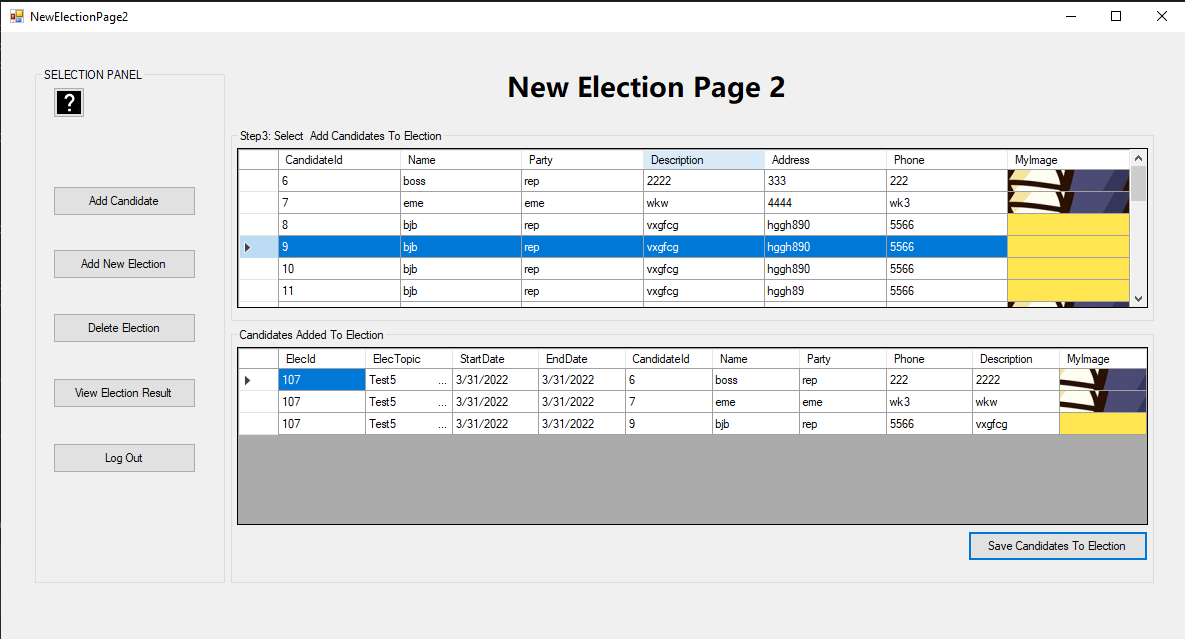


Figure 4.3 Updated NewElection Design(New Election Page2)

### 4.3.2 Quality Goal - 2

| **Quality Goal** | The application should be reliable by being available 99% of the time. |
| --- | --- |
| **Quality Metrics** | The application tracks the overall error count to make sure it is reliable and available. |
| **Satisfaction Status** | Satisfied |

To meet this goal, we have error handling in all the major functions that read or write to the database. Once an error is caught, it will write to our Logging system up in Azure Cloud SQL Server. Also, if a user gets one error when casting a vote or when viewing voting results, an email is sent to the Administrator. The following figure is a screenshot of the Logs table. It contains all the log messages including the user’s login status, error messages, warning messages, etc. An administrator can track the locations of the error by looking at the StackTrack column.

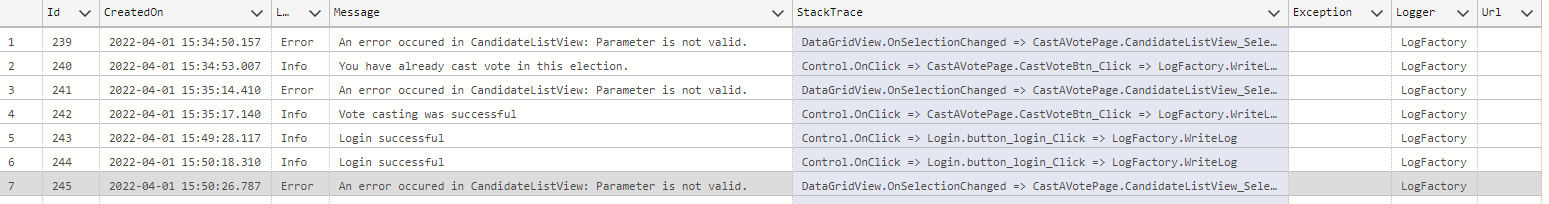


Figure 4.4 Screenshot of Logs Table In Database

### 

### 4.3.3 Quality Goal - 3

| **Quality Goal** | The application should be efficient by handling at least 10 simultaneous submissions at a time. |
| --- | --- |
| **Quality Metrics** | The application tracks when the user submits their vote. This tracks the successful count for voting. |
| **Satisfaction Status** | Satisfied |

The Admin will be able to check this goal by running a query in our database to see if there are any errors related to simultaneous submissions. Each error has a timestamp and can be counted up.

### 4.3.4 Quality Goal - 4

| **Quality Goal** | The voting submission process should take no more than 5 seconds to return a response. |
| --- | --- |
| **Quality Metrics** | The application tracks when the user submits their vote and records the start time and end time. |
| **Satisfaction Status** | Satisfied |

We are trying to handle this goal by making sure the user is able to cast their votes as soon as possible without any lags. We will also record when a user submits their vote to ensure quality. For example, from the Logs table, we can see that a casting vote took 0.46 s to complete, which satisfies goal 4.

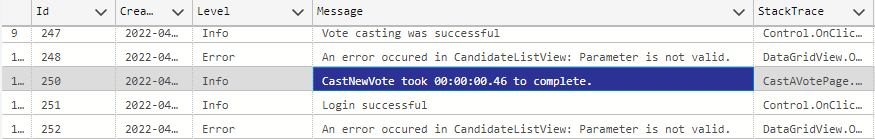


Figure 4.4 An Example Of Recorded Time For Casting A Vote

### 

### 4.3.5 Quality Goal - 5

| **Quality Goal** | The vote results should be shown no more than 5 seconds after a user submits their vote. |
| --- | --- |
| **Quality Metrics** | The application tracks how long it takes for the vote results to be shown to the user. |
| **Satisfaction Status** | Satisfied |

The vote results have a stopwatch setup to start when the query starts and stops when it is finished. The elapsed time is recorded in the Logging system. For example, from the Logs table, we can see that a user took 0.05 ms to get the vote result, which meets goal 5.

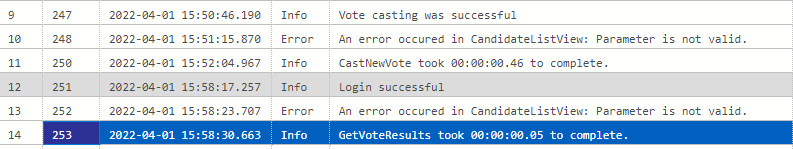


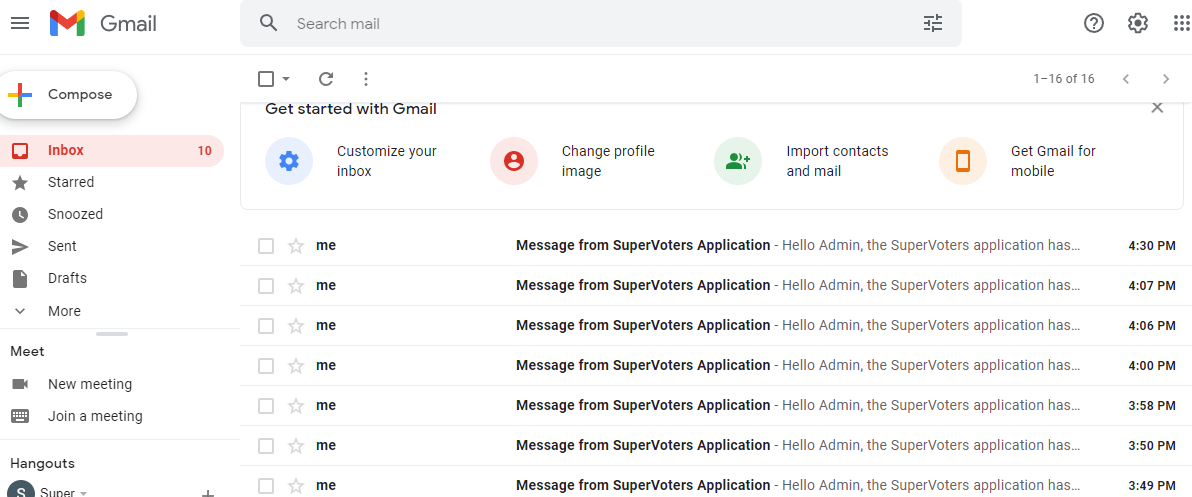
Figure 4.5 An Example Of Recorded Time For Getting A Vote Result

### 4.3.6 Quality Goal - 6

| **Quality Goal** | The system is robust and if it fails, a message is sent to the support staff. It must recover within 5 - 30 minutes depending on the outage. |
| --- | --- |
| **Quality Metrics** | 1) The application tracks when a user cannot login due to failure  2) The application tracks when an error has occurred during submission or showing results. This helps in seeing if our system is robust. |
| **Satisfaction Status** | Satisfied |

We will have an Administrator on call to monitor the email account and the Logger System. The agreement is to have a Service Level Agreement (SLA) of 5-30 minutes. Generally, a Windows machine can be rebooted if the problem exists on one machine. If it’s a system-wide issue such as Azure Cloud is down, it might be > 5 minutes.

Email:



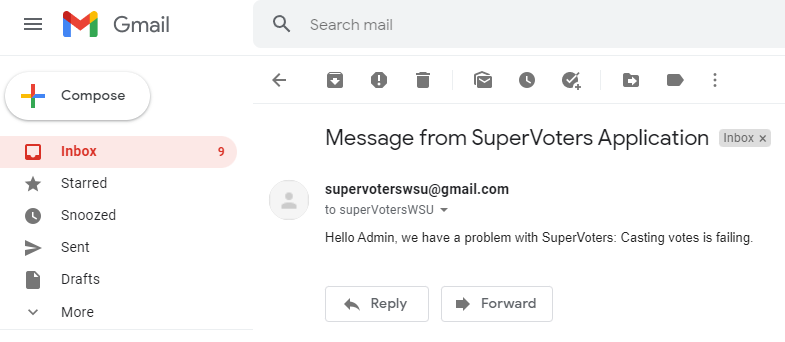


Figure 4.6 A Screenshot of superVotersWSU@gmail.com

### 4.3.7 Quality Goal - 7

| **Quality Goal** | The login process should authenticate the user within 5 seconds. |
| --- | --- |
| **Quality Metrics** | The application tracks when a user cannot get authenticated within 5 seconds. |
| **Satisfaction Status** | Satisfied |

This goal will help the user get authenticated as soon as possible in order to use the application. We will track if there is a problem with authentication or if the user is not able to login. That way the admin is able to help soon as possible to track any issues the user is facing and take immediate actions.

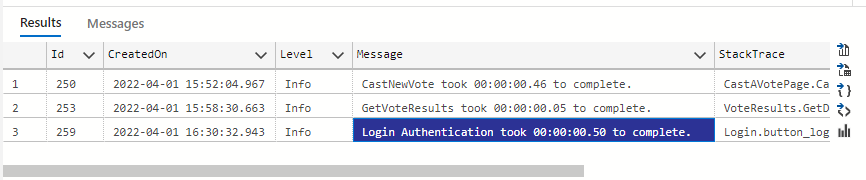


Figure 4.7 An Example Of Recorded Time For Login