

G04/11 PC Voting System

Tianrun Gu, Collin Dreher, James Bamberger, Casper Cheng

(I) Software Plan

1. Scope (Product overview and summary. Evolves into Sec. 1 of [requirements spec](#))

This product mainly provide service (voting system) for smart phones to send vote message to a specific smart phone which runs the voting software A. This system will count votes from text messages and enter the data into a Tally Table. Phone numbered will be saved to avoid duplicated votes.

1.1 Functions (What can this product do? Evolves into Sec. 3 of requirements spec)

This product's main functions include:

1. Voter can send short text message from his/her smartphone to a specific smartphone running voting softwareA (Android version).
2. Upon receipt of the short text message containing the ID number of the best poster (only one should be selected), the said ID number will be entered in a TallyTable with attributes CandidateID, count.
3. The phone number of the voter will be remembered in a VoterTable with attributes VoterPhoneNo, CandidateID, so that no voter can use the same phone to vote twice.
4. The administrator at the smartphone running voting softwareA can issue a special command, which will terminate the voting. The voting softwareA will display the TallyTable in decreasing number of votes so that the first winner will be displayed first and so on. At the end the VoterTable will be eliminated.

1.2 Performance (How well can this product do? Evolves into Sec. 4 of requirements spec.)

This voting system could handle at least two hundred concurrent users, five hundred thousand user profiles and be able to process a job placement inquiry in less than one second.

1.3 Limitations (What this product cannot do? Included in Sec. 8 of requirements spec.)

1. This voting system couldn't handle over-performance behavior like too many voters or too much messages need to process in a certain short time.
2. This voting system could only work between smartphones but not on any other platforms.

2.0 Tasks (From developer's viewpoint, what are the tasks? For

example tasks may include user interface, database manager, data mining, profile management, and so on. Cost analysis is based on tasks.)

Tasks include:

1. Design and develop User Interface
2. Build the database for votes (TallyTable) and profiles (VoterTable)
3. Implement the interface of the server
4. Implement functions for certain behaviors like calculating the votes or check if a voter votes more than once.

3.0 Resources

3.1 Hardware (for development and for deployment)

1. Personal computers

3.2 Software (what do you need for the development environment?)

1. UML tools for UML design
2. IDE for coding

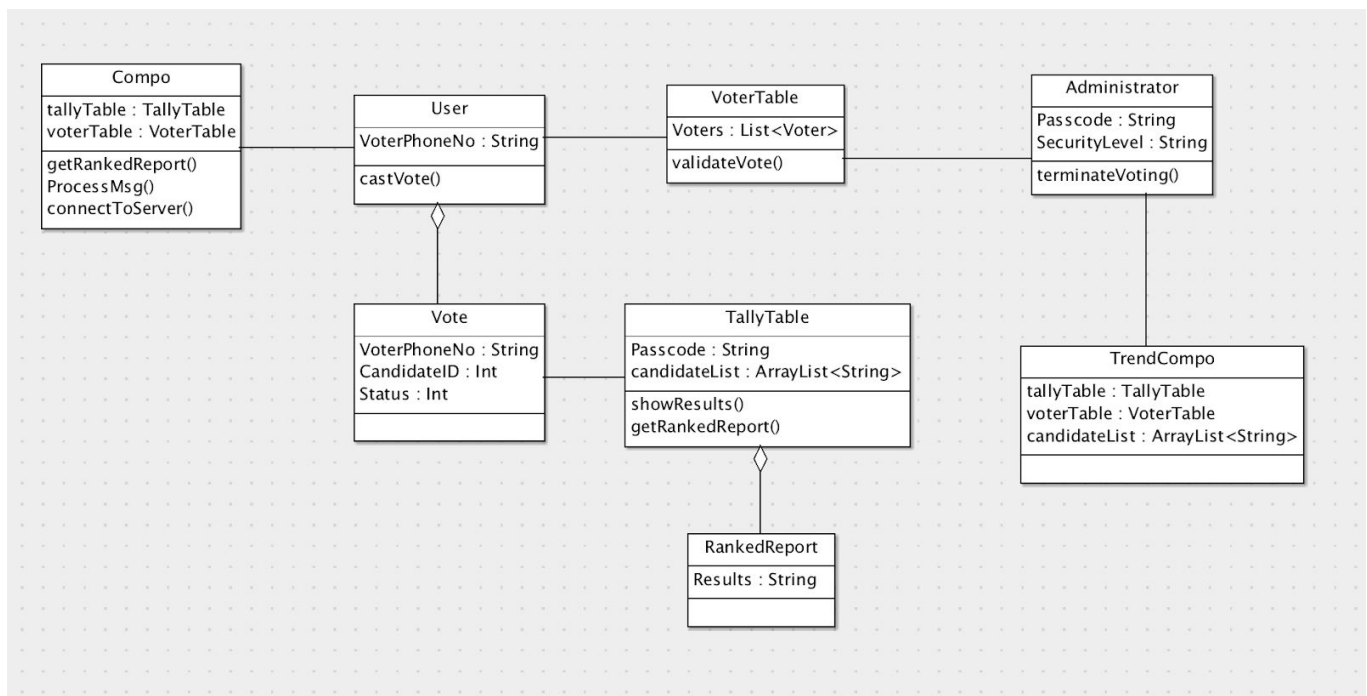
3.3 People (introduction of personnel and their roles)

1. Software developers
2. Software testers
3. Project manager

(II) UML Diagrams

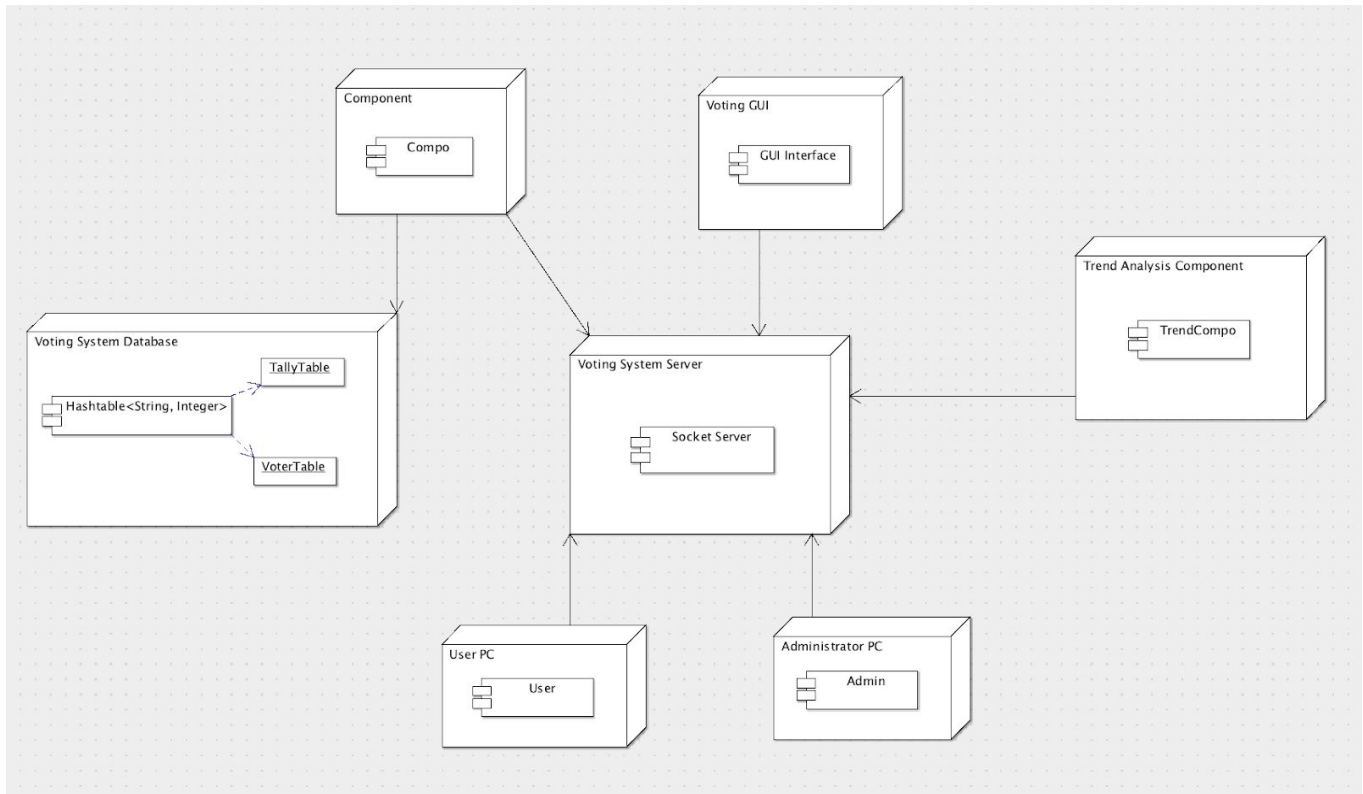
Class Diagram

The class diagram displays the structure of our system. The main object that performs the most of the tasks for our system is `Compo`. In this component we have users that are able to cast votes. The votes are stored in the `TallyTable` class, and also in `VoterTable`. These objects only exist because of the `Compo` object. Additionally, the `RankedReport` object is only ever created using the `TallyTable`. Lastly, we have an `Administrator` who has the authority to terminate voting - creating our trends analysis component, `TrendCompo`.



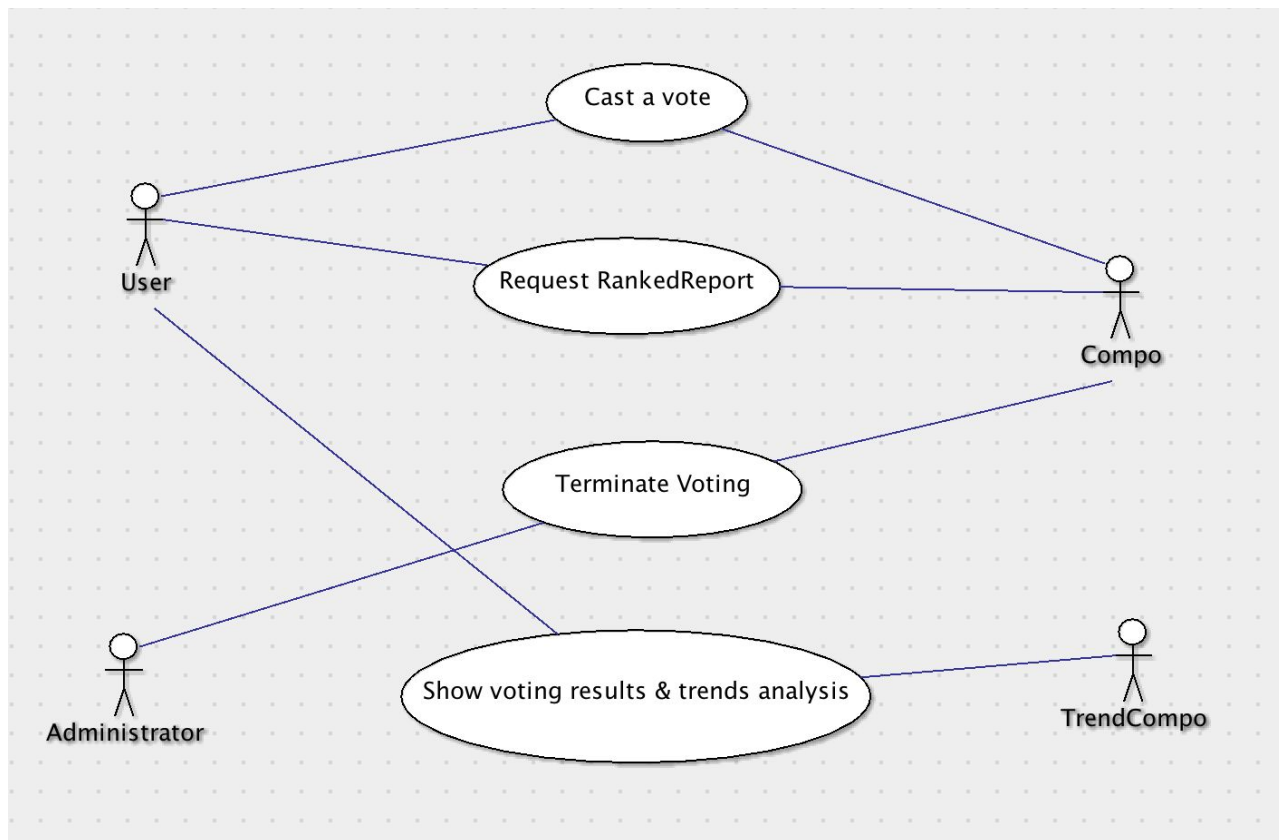
Deployment Diagram

The deployment diagram is meant to show what specifically our system uses from a hardware/software level. We basically have three main components: `Compo`, `TrendCompo`, and the `VoteGUI`. In order for the voting system to work at all, it is required to have a `Socket server` that is always running. This server connects all other aspects of the system. The three above components are connected to the server. As well as the `User` and `Admin PC`'s. Lastly, we have a 'database' for the voting system. This is not a standard SQL or external database. We simply use two `Hashtable` objects to create the `TallyTable` and `VoterTable`.



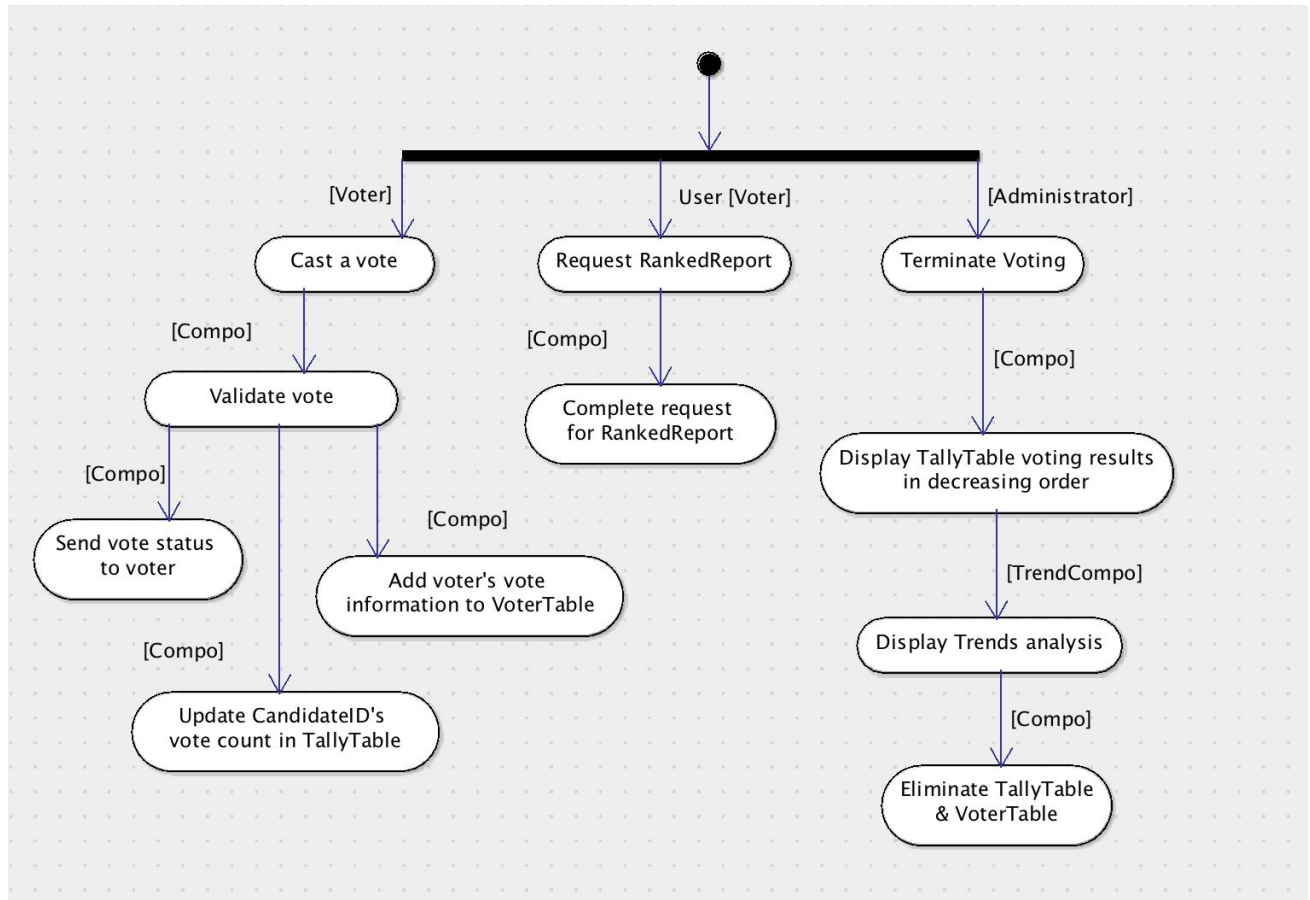
Use Case Diagram

There are two main users for the system: users and the administrator. The users main use case when interacting with the system is to cast a vote for a specific candidate or poster. The administrator has the ability to both start and end a vote. The voting component is the part of the system which will tally the votes casted and store them to be displayed when the vote is terminated. The Trend Component returns a trend analysis and the final voting results whenever a vote is terminated.



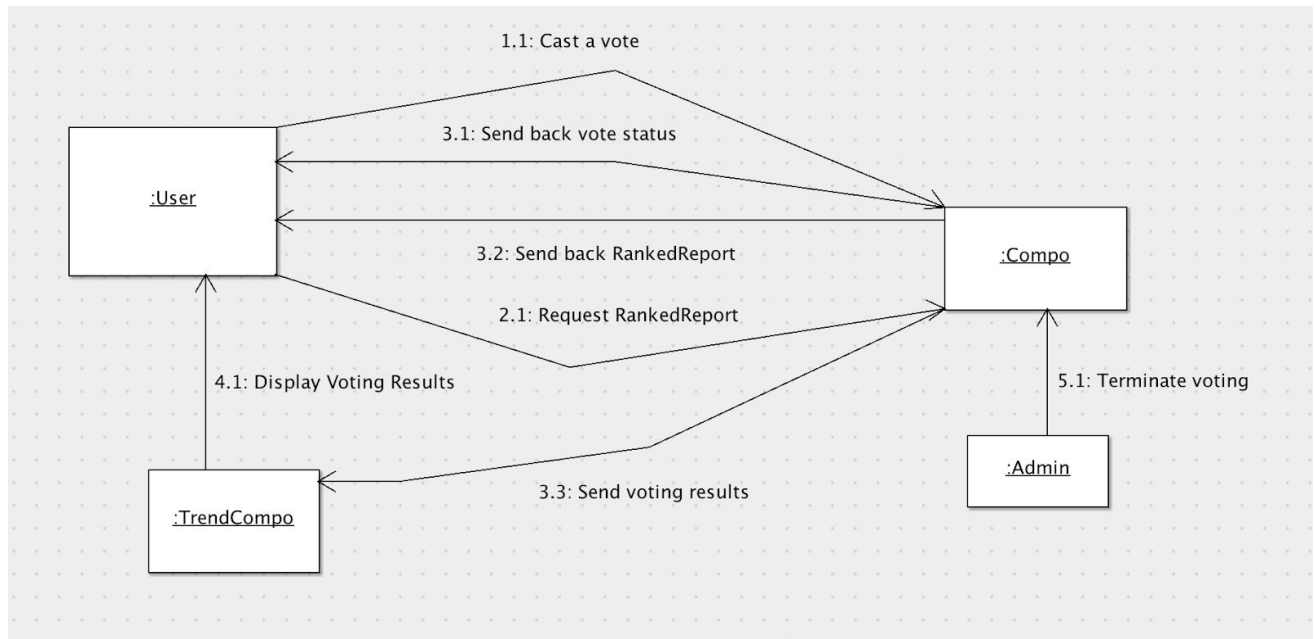
Activity Diagram

This activity diagram essentially shows the how the flow is from the Voter, User, and Administrator standpoint. For the Voter, he/she casts a vote which is validated. If it is validated, the component sends a message telling the voter the vote is casted successfully, Voter's information is logged, and CandidateID is added. Users can request a ranked report and shows the trend of the voting results. Administrator is responsible for terminating the vote when Voters are done and the TrendCompo uses the knowledge-database to display the TallyTable and the Trend analysis of the voting session. Both TallyTable and VoterTable are terminated at the end.



Collaboration Diagram

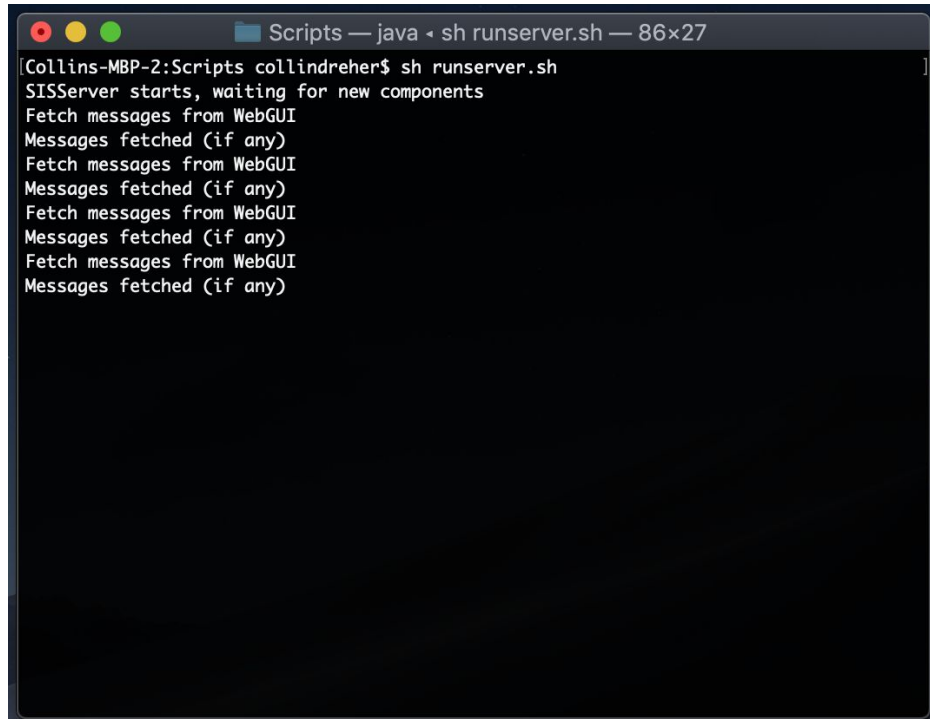
The Collaboration Diagram depicts how the objects interact with one another in the system. The User casts their vote to the voting component, which then returns if the vote was valid or not. Once the admin of the system terminates the vote, the voting component sends the vote information to the trend component so it may perform its trend analysis and display the voting results along with the trend information.



(III) Scenario Walkthrough

Scenario 1: Set up all components

1. In order to get all components connected to one another, you must first run the server. We have set up scripts in order to easily compile and run our components.

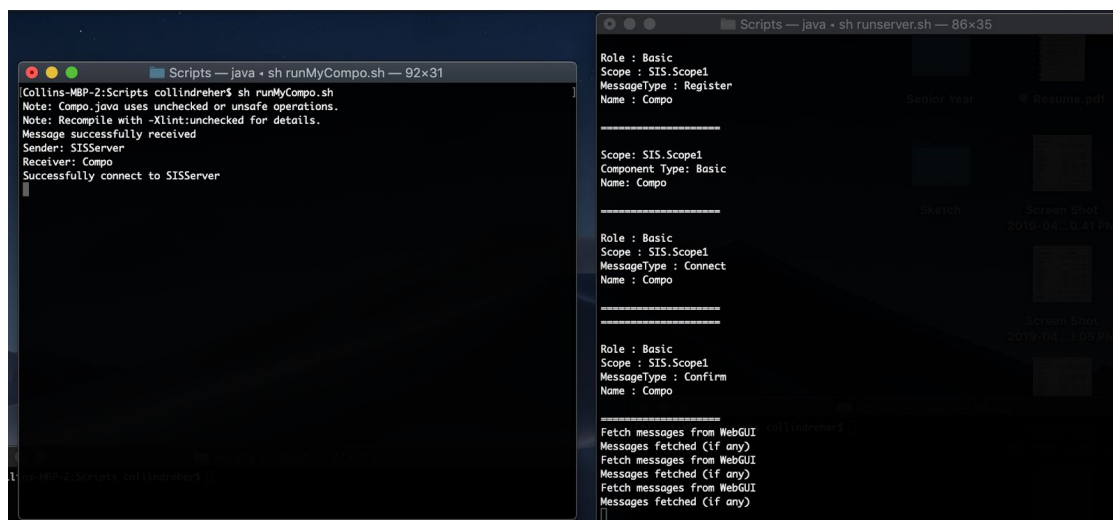


```

Scripts — java • sh runserver.sh — 86x27
[Collins-MBP-2:Scripts collindreher$ sh runserver.sh
SISServer starts, waiting for new components
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)

```

2. Connect the `Compo` component to the server. This is done by compiling and running the component, thus establishing a connection with our running server. The server receives and displays validation upon receiving external connection.



```

Scripts — java • sh runMyCompo.sh — 92x31
[Collins-MBP-2:Scripts collindreher$ sh runMyCompo.sh
Note: Compo.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Message successfully received
Sender: SISServer
Receiver: Compo
Successfully connect to SISServer

```

```

Scripts — java • sh runserver.sh — 86x35
Role : Basic
Scope : SIS.Scope1
MessageType : Register
Name : Compo

-----

Scope : SIS.Scope1
Component Type: Basic
Name : Compo

-----

Role : Basic
Scope : SIS.Scope1
MessageType : Connect
Name : Compo

-----

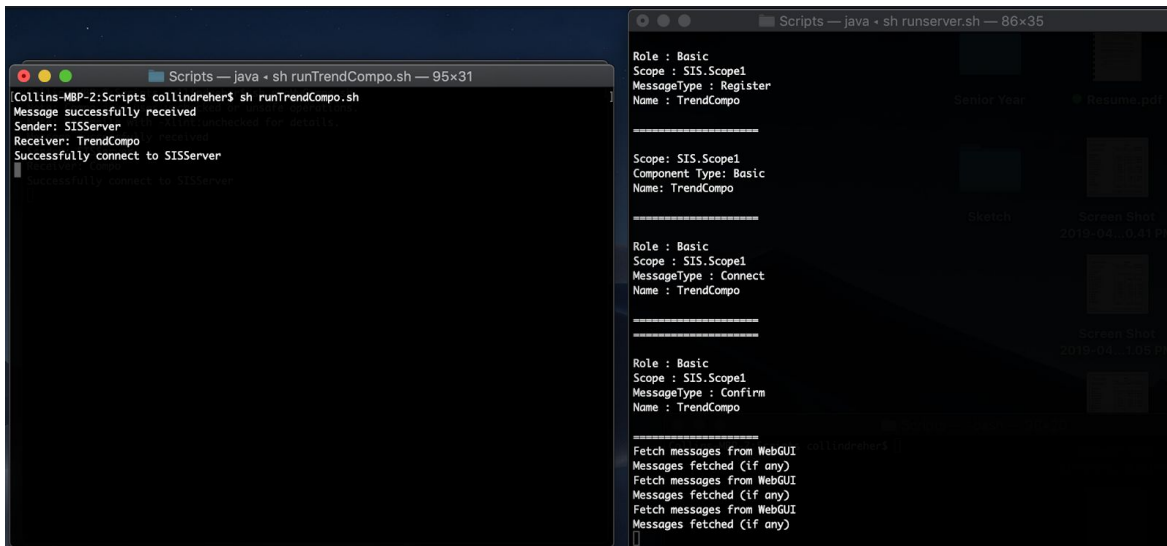
Role : Basic
Scope : SIS.Scope1
MessageType : Confirm
Name : Compo

-----

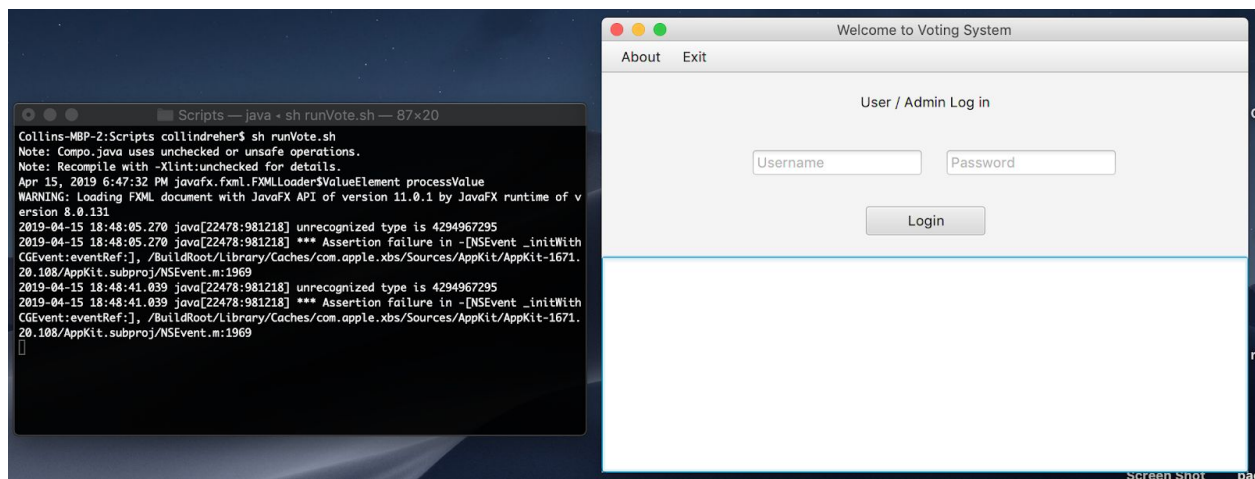
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)
Fetch messages from WebGUI
Messages fetched (if any)

```


3. Connect the Trend Analysis component, `TrendCompo`, to the server. Like before, a script is run that compiles and executes the java file, thus establishing a connection to the server. Again, the server receives the connection.

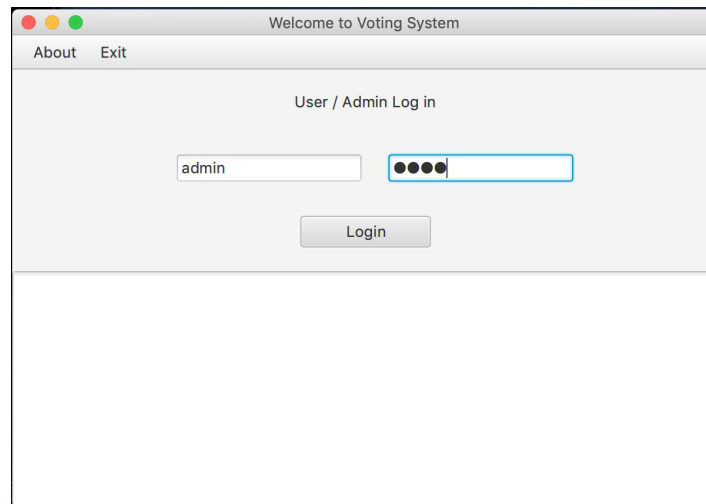


4. Run the Voter GUI. This GUI is a component created to replace the `PrjRemote`. This GUI is where the user can perform the rest of the project scenario.



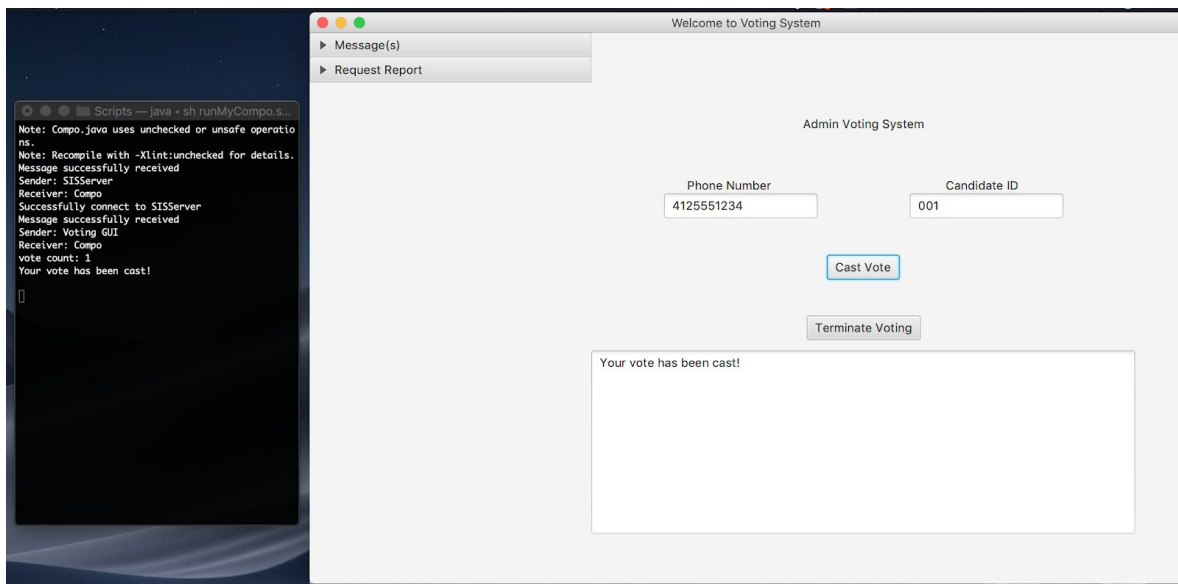
Scenario 2: Login

1. The user is able to log in to the GUI. The GUI will authenticate the user. If successful, the user will be sent to the next page, where they can continue performing tasks.

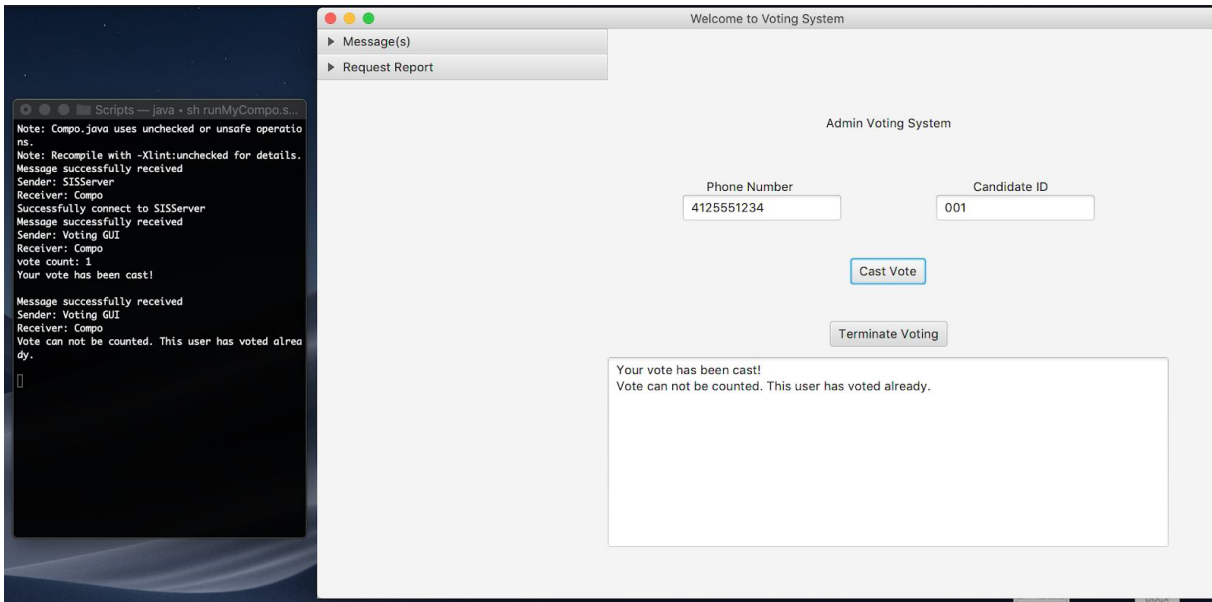


Scenario 3: Voting

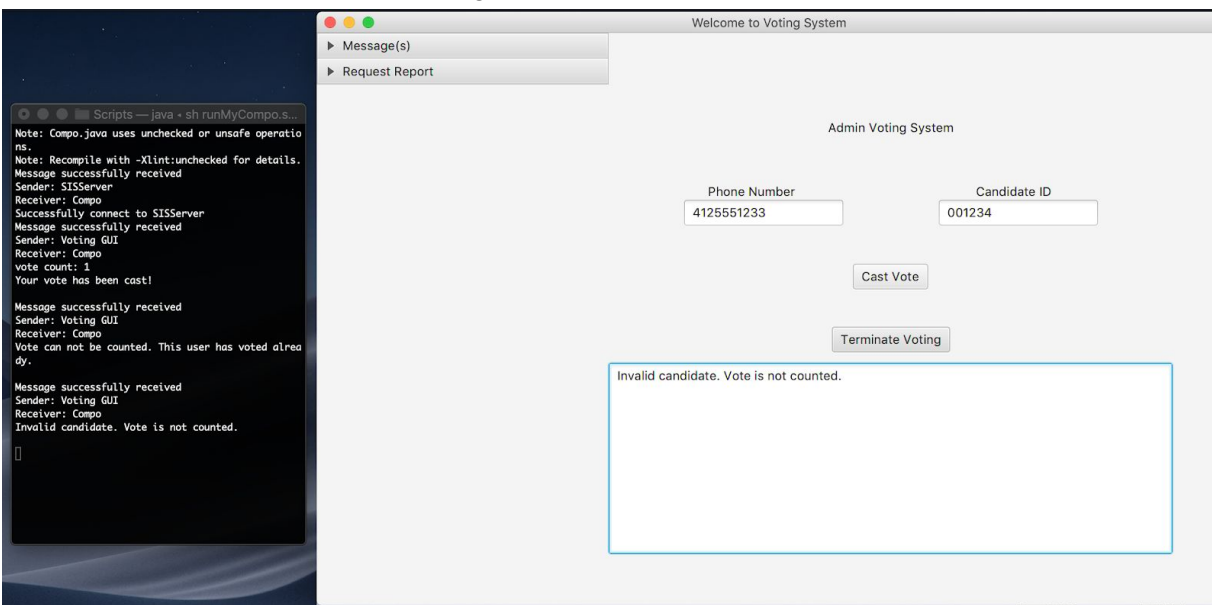
1. The user is able to cast a vote. The user must enter their phone number, as well as a candidate ID for the person they would like to vote for. Once finished, selecting the Cast Vote button will lock in their vote by sending it to the server. `Compo` will then grab the data and store in the `VoterTable` and `TallyTable`.



2. The user may attempt to vote again. However, this is not permitted. If the user attempts to vote again with the same phone number, their vote is not counted and they are returned appropriate output to the GUI.

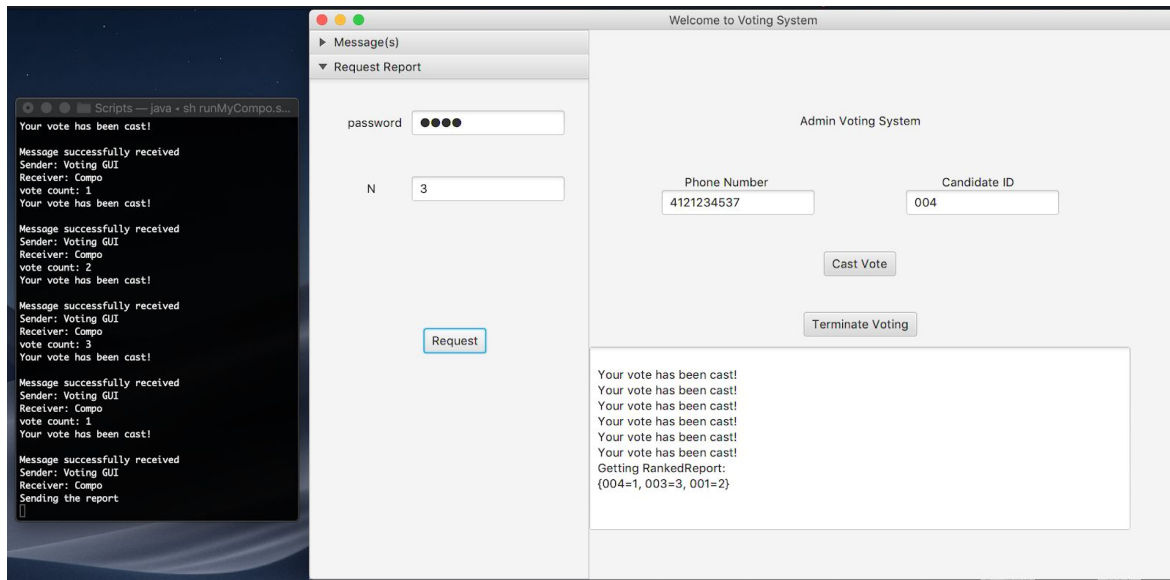


3. The user may attempt to vote for an invalid candidate. If the Candidate ID does not exist, the user is prompted to vote again.



Scenario 4: Request RankedReport

1. The user is able to request a report of the current tallies. A `RankedReport` is requested by the user by inputting the number of top candidates they would like to see. A password is also requested for authentication. Output is displayed accordingly.



Scenario 5: Terminate Voting

1. If the user is an administrator, they have the ability to click the Terminate Voting button in the GUI. By doing so, `Compo` is terminated, `VoterTable` and `TallyTable` are destroyed, and the results are sent to the `TrendCompo` for analysis. The analyzed data and predictions are then sent from the component to the terminal for display.

```

Message successfully received
Sender: SISServer
Receiver: TrendCompo
Successfully connect to SISServer
Message successfully received
Sender: Compo
Receiver: TrendCompo

***Poster IDs received***

Assumption that the Voting items are divided into the following categories:
AI
Literature
Math
Music
Philosophy
Science

TREND ANALYSIS RESULTS

Trends observed from this year's voting
|
| Votes for Category - AI :12.77%
| Votes for Category - Literature :19.18%
| Votes for Category - Math :8.51%
| Votes for Category - Music :6.38%
| Votes for Category - Philosophy :38.3%
| Votes for Category - Science :14.89%
|

Prediction of next year's VOTING TREND based off previous years data (Sophisticated Prediction)
|
| Votes for Category - AI :14.71%
| Votes for Category - Literature :10.78%
| Votes for Category - Math :4.9%
| Votes for Category - Music :14.71%
| Votes for Category - Philosophy :25.49%
| Votes for Category - Science :29.41%
|

```

```

Prediction of next year's VOTING TREND based off previous years data (Sophisticated Prediction)
|
| Votes for Category - AI :14.71%
| Votes for Category - Literature :10.78%
| Votes for Category - Math :4.9%
| Votes for Category - Music :14.71%
| Votes for Category - Philosophy :25.49%
| Votes for Category - Science :29.41%
|

Trends Observed
-----
Category AI is predicted to have a UPWARDS trend.
Category Literature is predicted to have a DOWNWARDS trend.
Category Math is predicted to have a DOWNWARDS trend.
Category Music is predicted to have a UPWARDS trend.
Category Philosophy is predicted to have a DOWNWARDS trend.
Category Science is predicted to have a UPWARDS trend.

Votes for Category AI are 85.09% from Computer Science backgrounds
Votes for Category AI are 14.91% from General Science backgrounds

Votes for Category Literature are 38.91% from Computer Science backgrounds
Votes for Category Literature are 61.09% from General Science backgrounds

Votes for Category Math are 43.22% from Computer Science backgrounds
Votes for Category Math are 56.78% from General Science backgrounds

Votes for Category Music are 12.92% from Computer Science backgrounds
Votes for Category Music are 87.08% from General Science backgrounds

Votes for Category Philosophy are 35.46% from Computer Science backgrounds
Votes for Category Philosophy are 64.54% from General Science backgrounds

Votes for Category Science are 73.69% from Computer Science backgrounds
Votes for Category Science are 26.31% from General Science backgrounds

```

The number of votes and the candidate ID are passed into a Ontological knowledge-base, where the counts for categories are later tallied up in order to compute this year's voting analysis. From this year's voting data, we can observe the trend of this year's voting, as we can see which category gets what percentage % of votes.

Extra credit components (2 Extra points total)

Ontological knowledge-base (1 Extra Point)

Below shows the ontological knowledge-base when visualized

```
Can_Id, category, year, num_of_votes
001, AI, 2019, 6
002, Literature, 2019, 9
003, Math, 2019, 4
004, Music, 2019, 3
005, Philosophy, 2019, 18
006, Science, 2019, 7
```

Sophisticated Prediction for NEXT year (1 Extra Point)

With this year's voting data, we read in two years of past data in order to generate a prediction of what the data will be like in 2020.

```
-----
| Prediction of next year's VOTING TREND based off previous years data (Sophisticated Prediction)
|
| Votes for Category - AI           :14.71%
| Votes for Category - Literature :10.78%
| Votes for Category - Math        :4.9%
| Votes for Category - Music       :14.71%
| Votes for Category - Philosophy :25.49%
| Votes for Category - Science     :29.41%
|
|-----
```

From the predicted voting results for different categories for 2020, we can also observe whether the predicted trend is going up/down compared to year 2019 as shown below.

Trends Observed

```
-----
Category AI is predicted to have a UPWARDS trend.
Category Literature is predicted to have a DOWNWARDS trend.
Category Math is predicted to have a DOWNWARDS trend.
Category Music is predicted to have a UPWARDS trend.
Category Philosophy is predicted to have a DOWNWARDS trend.
Category Science is predicted to have a UPWARDS trend.
```

Based on the ontological knowledge base, we also assigned backgrounds for different categories on people so we can further analyze the trend as we know the breakdown of people who votes for different categories. For example, we can see that 85.09% of voters that come from Computer Science backgrounds voted for AI while only 14.91% of voters who come from General Science backgrounds voted for AI. From this further analysis and breakdown, we can see how Computer Science people are more interested in the AI category and maybe next year we can direct more of our attention to General Science background voters to get the rest of their votes.

```
Votes for Category AI are 85.09% from Computer Science backgrounds
Votes for Category AI are 14.91% from General Science backgrounds

Votes for Category Literature are 38.91% from Computer Science backgrounds
Votes for Category Literature are 61.09% from General Science backgrounds

Votes for Category Math are 43.22% from Computer Science backgrounds
Votes for Category Math are 56.78% from General Science backgrounds

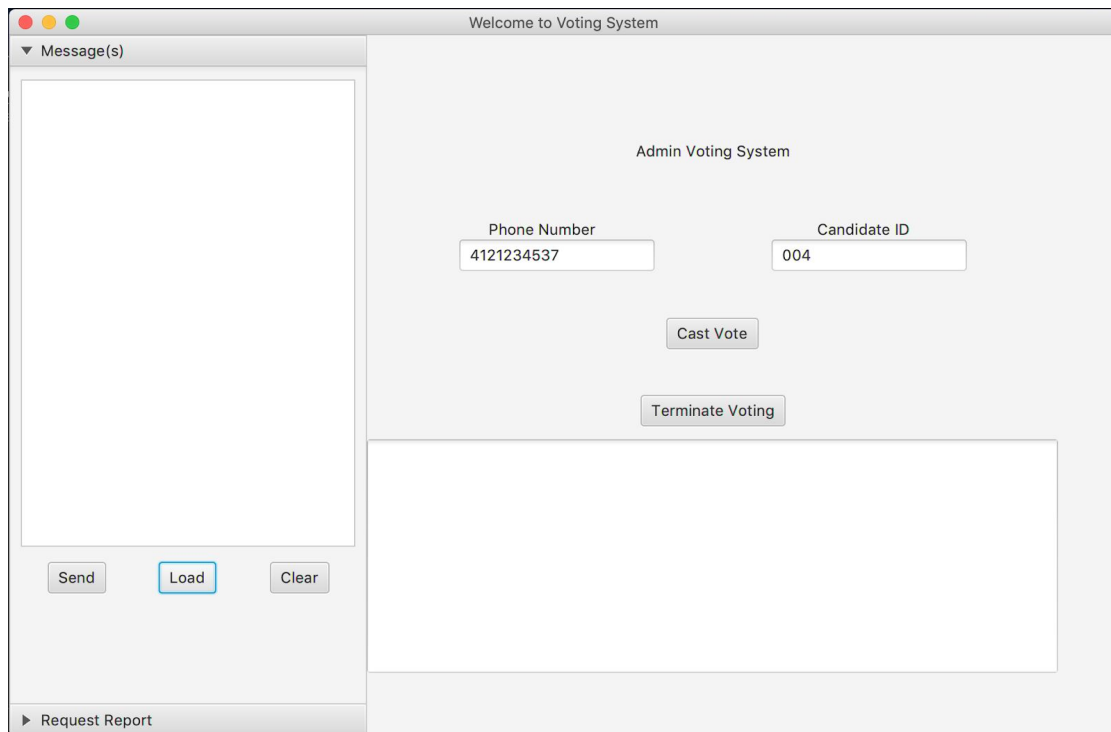
Votes for Category Music are 12.92% from Computer Science backgrounds
Votes for Category Music are 87.08% from General Science backgrounds

Votes for Category Philosophy are 35.46% from Computer Science backgrounds
Votes for Category Philosophy are 64.54% from General Science backgrounds

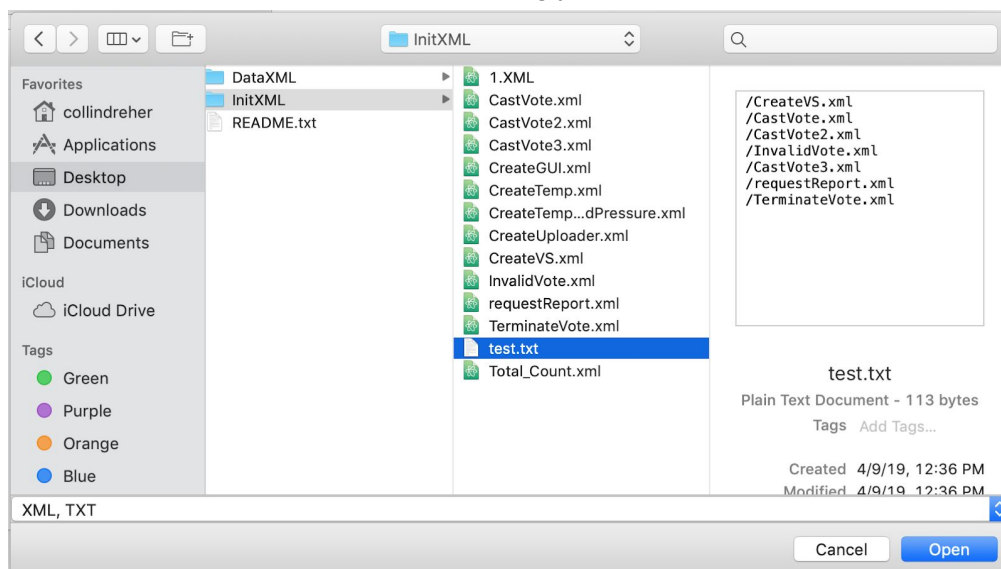
Votes for Category Science are 73.69% from Computer Science backgrounds
Votes for Category Science are 26.31% from General Science backgrounds
```

Scenario 6: Testing

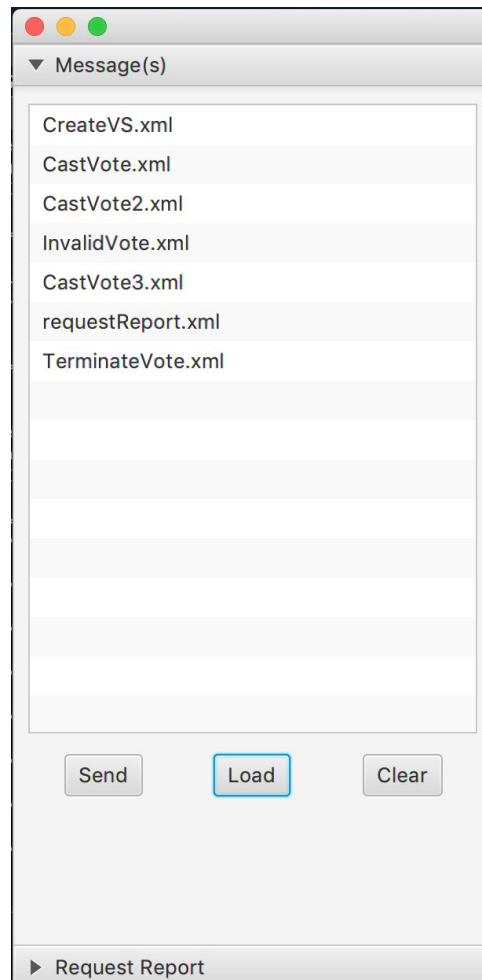
1. The user is able to test the program with a basic test script. An accordion is able to be opened, displaying several buttons and a text area. A test script file can be loaded by clicking the Load button.



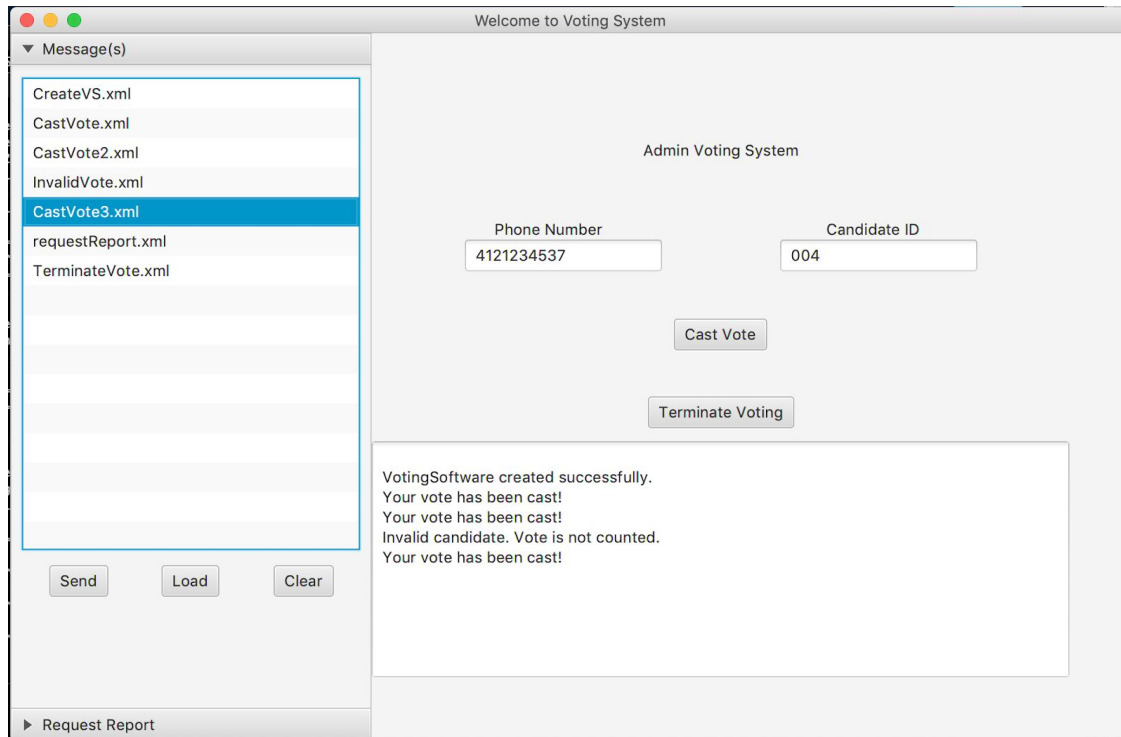
2. Upon doing so, a `FileChooser` object is opened. This allows the user to load in any file with .XML or .TXT extensions. If a .TXT file is loaded, the GUI will parse through each line and open the .XML files accordingly.



3. Once the test script file is loaded, each .XML file is displayed in the order that is was written on the .TXT file. The user is then able to select any file they would like, and execute that test file.



4. Once selecting a file to test, and hitting the Send button, the test is executed. Just like any other user task performed above, the test case is executed using the multiple components and their established connection to the server. The example below shows a test script that creates the voting software, followed by casting several valid and invalid votes. Not yet executed are the test files for requesting a `RankedReport` and terminating the voting.



(IV) Source Code

1. ../SISv5/Components/M3/Compo.java
2. ../SISv5/Components/M3/Controller.java
3. ../SISv5/Components/M3/TrendCompo.java
4. ../SISv5/Components/M3/voteGUI.java
5. ../SISv5/Components/M3/Vote.fxml
6. ../SISv5/Components/M3/Admin.fxml
7. ../SISv5/Components/M3/login.fxml
8. ../SISv5/xml/InitXML/test.txt
9. ../SISv5/xml/InitXML/CreateVS.xml
10. ../SISv5/xml/InitXML/CastVote.xml
11. ../SISv5/xml/InitXML/CastVote2.xml
12. ../SISv5/xml/InitXML/CastVote3.xml
13. ../SISv5/xml/InitXML/InvalidVote.xml
14. ../SISv5/xml/InitXML/requestReport.xml
15. ../SISv5/xml/InitXML/TerminateVote.xml
16. ../SISv5/NewSISServer/SISTask.java