Paradigms of Programming Coursework Term 1 - Year 1

Table of Contents

Concise List of Bugs And Weaknesses

Bugs

- GUI Glitch (Singular Vote)
    - A glitch identified during testing in the hackathon that causes a graphical defect. If the student opts to only vote for a single candidate, selecting “N/A” for the second position, two dropdowns remain at the bottom right of the screen after voting. Functionality works flawlessly, graphical bug identified but unable to be rectified in time.
- SQL Data Caching
    - A bug has been noticed during our generation of dummy data to show the customer. While the SQL queries should be dynamic and get the data from the remote MySQL database in real time, the calculation of results appears to cache within the application. All other queries are functional, supporting data changes in the database to immediately be present in the application (such as a new candidate), however, the result calculation requires a program restart to be detected. If the change is performed from the application, rather than just remotely (such as a new vote), then they are instantly shown.
    - Possible fix would be to force a schema refresh on page load in the event it has changed remotely.
    - NB: This is a very low priority bug that has been identified and does not interfere with any of functionality as per the project specifications. The bug has only been noticed during the testing of additional functionality that has been made possible due to the method of data storage which would enable multiple voting stations to operate at once with a centralised data storage location.

Weaknesses

- New Election Creation Validation
    - There is currently not a failsafe or any form of validation in place to ensure that there are not two elections created that overlap in which case would cause considerable disruption to the application. The only validation in place is to ensure the end date is after the start.
- Past Election Results
    - While not requested in the client specification at any time, this is still a limitation regardless. The current code bases the results from the current election, with some time and modification this can be adjusted to take input from the user to select past elections as well.
- Graph Display Clarity
    - One weakness is when displaying the results, we decided to create a graph for visual representation, but when looking at the graph it is difficult to decide which bar is representative of each candidate and also where first votes finishes, and second votes starts. To overcome this problem, we attempted to create a blank bar in between the first and second votes, however this didn’t work, because technically a candidate could have zero votes which makes it more complicated.
- Winner Calculation
    - STV is used at all times. Due to the method in which the program has been designed, the initial calculation of the winner is based off the first preference vote

count only, then uses the STV for all other preferences. It was noted within A.5.2 that unless there was a tie, the total number of votes should be taken used to determine the winner which contradicts the spec that implies a tie in only the first preference. In this instance, we were unable to contact the customer for clarification so it must be presumed that the total votes in this case should include all preferences.

Classes and OOP Features

Classes Used

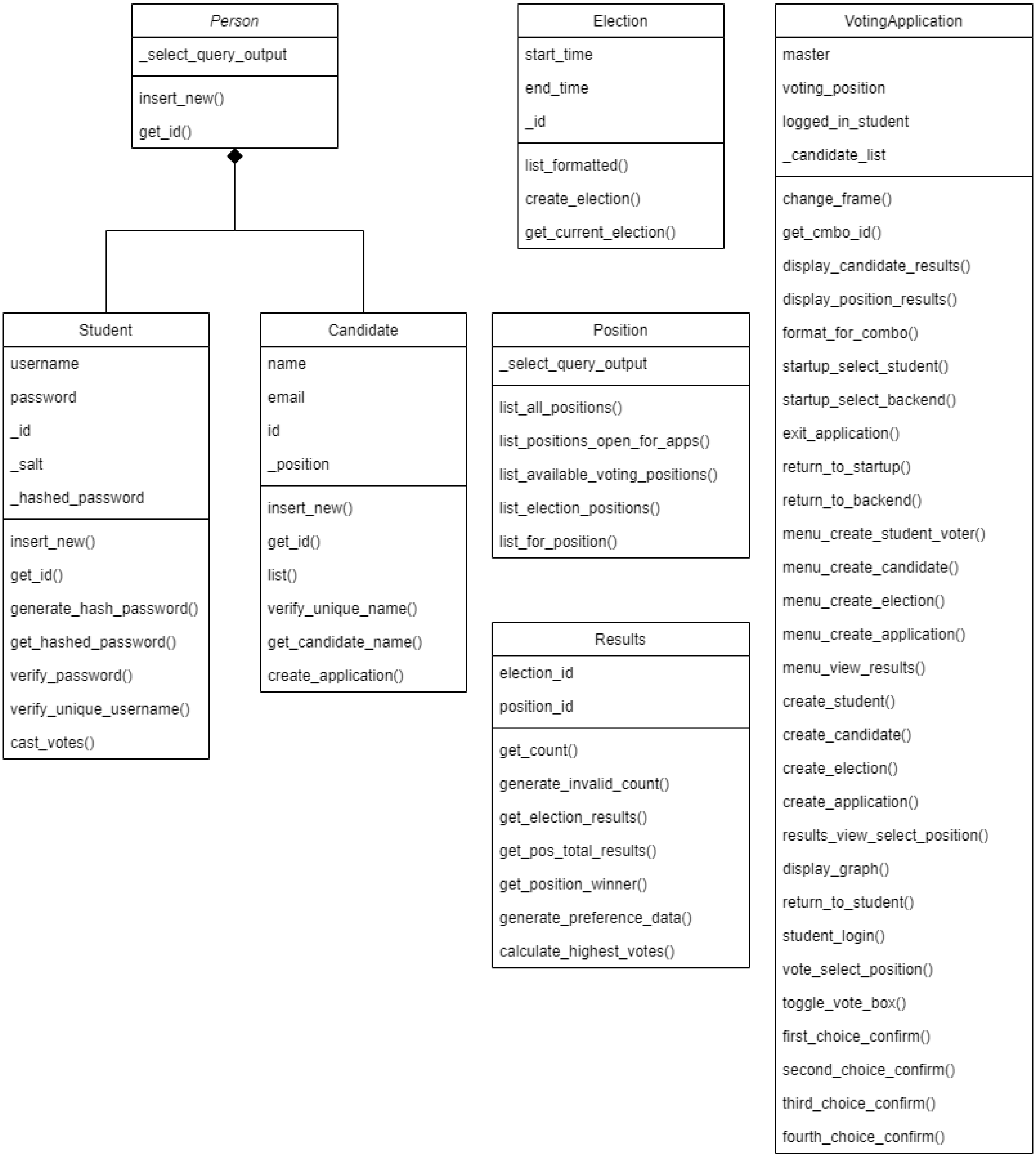
1. Person Class
2. Student Class
3. Candidate Class
4. Election Class
5. Position Class
6. Results Class
7. Voting Application (UI)

Explanation of Class Diagram & OOP Features Used

We broke the project down into six main classes.

- The 'Person' class allows abstract methods for different classes such as student and candidate, this ensures that two methods are implemented and that a private variable is available.
- The 'Student' class handles actions related to the student, including large functions like login authentication and password hashing. It will also provide the functionality to enable the system to insert a new student into the database, with the additional feature set of verifying that the username is unique before insert. This class will also handle the submission of votes that the student makes.
- The 'Candidate' class will allow you to insert a new candidate into the system, as long as the name is unique. It will also handle the insertion of candidate applications for a position, leveraging other class structures to do that more efficiently.
- The 'Election' class will allow the software to list the current elections as well as create elections for the future.
- The 'Position' class will list all positions with an array of filters based on them, ranging from including positions that have candidates in an election and positions which a student has not voted for. The functionality is also present to ensure that positions are only votable if four candidates have applied to them.
- The 'Results' class deals with counting results and determination of who the winner is for each position in the election. The winner will be returned so that it can be announced to the students that voted. First, second, third and fourth votes will also be processed and returned in this class, in case there needs to be a move to second preference votes.
- As for the ‘VotingApplication’ class, displaying and controlling of user interface actions is done via this class that includes a large number of different functions. The reasoning for a single class to be used in this case is to improve inter-communication with functions that target interface elements. This class handles the output of the system/election occurs, so, for example, the winner, the number of votes they achieved (first votes) and the total amount of votes are needed.

Class Diagram



Description of The Features Implemented

Feature 1 - Interactive GUI

The GUI was developed throughout the project to ensure that only required backend features were created that tied into frontend functionality. The GUI was implemented window by window in a logical way. We used Pygubu to make it easier to create the UI, as it allowed us to see different windows and what they would look like without running the program and without having to type multiple lines of code. It also helped us to lay out each window easier, meaning we could make our program look nicer and easier to use with less time and resources being allocated to it.

Feature 2 - Login with UID and Password

We developed the student login page in the standard mode page; this is the section of the program where students can go and vote for the positions that are running in this election, as well as viewing results.

Before we can log into to the application a student voter has to be created, this is done by going into the backend admin section of the program and selecting create a student voter; this will take you to a form that you can fill out which requires you to type in a unique username and password, with both inputs being validated before submission. Once the student voter has been created, the details will be stored on our database in the student voter database. When you log into the student voter page, the program does a MySQL query to search for the login and password, and if they matched with the record stored in the database, then the program will allow the student to login to the application and start casting votes. We also implemented password hashing into our login page to improve security and bring the overall project up to a more professional standard, as it is intended that the application will be distributed and used within the university (in the scenario). We decided to store all our information regarding the logins within a database because it is a centralised place for the information to be kept meaning that any student could log into their account from any computer and cast their votes.

We implemented the login page by using the student class. This class reads the password and user name that the student types in when they first create the account and stores uses the insert_new function to store the information into the database. The “verify_password” function searches for the username and password in the database and checks whether the details that have been typed in are correct or not, it uses the stored salt to regenerate the inputted data to check it matches the hash stored. If the details are correct, then the user will be taken to the next page where they are allowed to votes, and if the input is incorrect, then the user will be prompted to retype in their username and password.

Feature 3 - Upload Candidates by GSU Position

When creating a candidate and assigning them to a position, we again used a database for this. Creating a candidate and assigning them to a place is done by two steps on our application. Firstly, you have to create the candidate; this is done by going into the admin section of the program and selecting the create a candidate option, this will take you to a form to fill in where you have to fill in the candidate's name and email address. Once the candidate is created, this is stored in the candidate database and is given an ID (e.g. 1, 2, 3…).

To assign the candidate to a position you have to again go into the admin backend section of the program and this time select create candidate application, this will take you to a form containing drop-down menus where you select the election you would like to run for, this candidate name and then finally the position. The drop-down menu's only shows positions that are open (when there are less than four people that have already applied) and candidates that are eligible to run for the position (ones that have not yet applied for a post). The database then takes the primary key of the candidate and the primary key of the position and puts them in a new database called candidate applications. The primary keys are put together as foreign keys in the database; the app gets given its own primary key. This is also the table which determines whether the position is still available or filled due to too many candidates running.

We chose to use databases to implement this feature because databases allow us to create primary and foreign keys which can link tables together, making it a lot easier to use, then using text files and spreadsheets. This makes our validation very good for this section of the application and stops people running for multiple positions and also closing positions as soon as they get full up.

This feature using the candidate class to insert the new candidate and creates the application. To generate the candidate, the program uses the "insert_new" function to inset the candidate into the database. Once the candidate is in the database, then you can assign the candidate to a position. To attach a candidate to a position the program uses the "create_application" function, this function executes a MySQL query to insert the data selected in the drop-down menus on the create application page and then stores the data into the "gsuCandidateApplication" and then commits the changes.

Feature 4 - Cast votes

When the student is logged into the application, they are able to cast a vote for a particular position that is running on the current election. The student first must select a position from the drop-down menu; if a student has already voted for a particular position, then the voter will be unable to vote for the position again, and as a result, the option for that specific position will be omitted from the list.

Once a position has been selected, the student can now cast their votes in order of preference. Once the votes have been cast the ballots get saved in the election votes database ready for the votes to be counted and displayed. Each vote gets given its own ID as well as the foreign key for the voter, the position they have voted for and the candidate ID the voted for each preference. We decided that implementing this section with drop-down menus to allow the student to cast their votes would be easier as they don't have to remember the name of the candidates as well as retaining the exact name of the position of they were voting for, this makes it easier and minimises mistakes as the voter has to make the selection from the drop-down menu. If the candidate wishes to only vote for one candidate, this is also handled elegantly via the use of the drop-downs.

We also decided to store all our votes in a database again because it makes it easier to read the information with a few simple MySQL queries compared to a for loop to look through a text document. Having a database also makes everything central, so all votes get stored in a central location, making it easier to count votes and allowing more machines to be used for voting.

To cast a vote, the program users the student class and the position class. Firstly the position class is used to display the positions available to vote for. The "list_available_voting_positions" function from the position class I the class that is responsible for displaying all the places that the students is currently available to vote for, this function is also used to validate what the student can vote for so they don't vote twice for the same position. Once the position has been selected, the "list_for_position" function will be used to display all the candidates that are currently running for the position. This is where the student can choose what candidate they want to vote for in order of preference. Once the choices have been chosen, the student class then uses the "cast_votes" function to read the options that the student has selected and then executes a MySQL to insert the choices made into the "gsuElectionVotes" database.

Feature 5 - Count Votes

When counting votes, we had to take the votes that had been cast in the previous section, which meant we had to read how many times that a particular candidate appeared in the first votes (or second, third, fourth) and count them up. Then we had to compare these numbers to each other for each candidate on a particular position. Implementing this wasn't too difficult because we used a database, meaning we didn't need to use any variable or loops. The functions we created during this process could be reused later in another feature to display the results. By storing all the information in a database, we prevented ourselves from having to handle complex loops or wasting CPU cycles, instead, we could call a single query to get all votes for a specific candidate in the election.

The total number of votes are displayed next to the candidate on the position results screen. Unfortunately, the winner of each position is generated solely on the Single Transfer Vote method, creating the winner from the 1st preference, before checking the 2nd, 3rd and 4th preference if there is a tie between any candidates. While this method makes more sense and was included/implied within the spec, the feature task directly contradicts this by stating that the winner is based merely on the total number of votes.

Feature 6 - Visualise Results

To implement the visualisation of results, we added a button at the bottom of the page where students can vote for positions so they can view the results, as well as on the backend admin menu.When this button is pressed, they will be taken to a page where a position drop-down is available to display per-position results, or they can view all the election results. If the student chooses a position to view, the results for they will be shown a table with a break down of votes that each candidate received in order of preference and the number of people voted in the election for that position. If the student clicks on the view election results button, they get shown a list of all the positions in the ballot along with the winners for each position; the page also gives the user the option to save the results in a text file for printing.

This feature was pretty easy to implement because the information displayed on these pages is easily generated and retrieved from the database, making use of the results class.

The results class is full of functions that work together to display the information on the screen. Once a user has selected a position they wish to view the results for the "generate_preference_data" function returns the preference votes from the database for all the candidates within that function and displays them on the screen. The "get_position_winner" function returns the winner for the given position and the "get_pos_total_results" function gets all the results for the given position and returns them. Should the user wishes to see all the results for all positions, the program uses the "get_election_results" which gets all the results for every position within an election.

Testing

Blue = Integration testing with other modules.

Case Num	Test Case	Actions performed	Inputs	Expected	Outcome	Status	Corrective Action
Startup Menu							
1	Admin Mode Btn	Mouse Click	N/A	Page changes and user is navigated to admin menu.	Page changed and user is navigated to admin menu.	**Pass**	N/A
2	Standard Mode Btn	Mouse Click	N/A	Page changes and user is navigated to user menu.	Page changed and user is navigated to user menu.	**Pass**	N/A
3	Quit Btn	Mouse Click	N/A	Window closes and exits gracefully.	Window closes and exits gracefully.	**Pass**	N/A
Admin Menu							
4	Create Student Btn	Mouse Click	N/A	Page changes and user is navigated to create student window.	Page changes and user is navigated to create student window.	**Pass**	N/A
5	Create Cand Btn	Mouse Click	N/A	Page changes and user is navigated to create candidate window.	Page changes and user is navigated to create candidate window.	**Pass**	N/A
6	Create Election Btn	Mouse Click	N/A	Page changes and user is navigated to create election window.	Page changes and user is navigated to create election window.	**Pass**	N/A
7	Create Cand App Btn	Mouse Click	N/A	Page changes and user is navigated to create candidate app window.	Page changes and user is navigated to create candidate app window.	**Pass**	N/A
8	View Results Btn	Mouse Click	N/A	Page changes and user is navigated to view results window.	Page changes and user is navigated to view results window.	**Pass**	N/A
9	Main Menu Btn	Mouse Click	N/A	Page changes and user is navigated back to main menu.	Page changes and user is navigated back to main menu.	**Pass**	N/A
Create Student Voter Page							
10	Create Student Voter	Entered data, submitted form.	User: TestingT Pass: Test27	User should be created, added to database.	User is added to database, created successfully.	**Pass**	N/A
11	Create Student Voter (Blank Username)	Entered password, left username blank, submitted form.	Pass: Test27	Program should not insert user, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
12	Create Student Voter (Blank Password)	Entered username, left password blank, submitted form.	User: TestingT	Program should not insert user, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
13	Create Student Voter (Blank Username + Pass)	Left both username and password blank,	N/A	Program should not insert user, should return error.	Error returned to user, passed successfully.	**Pass**	N/A

		submitted form.					
14	Create Student Voter (Nums In Username)	Entered data, submitted form.	User: Test27 Pass: Test27	User should be created, added to database.	User is added to database, created successfully.	**Pass**	N/A
15	Create Student Voter (Special Chars in Pass)	Entered data, submitted form.	User: TestingT Pass: Test27!>#	User should be created, added to database.	User is added to database, created successfully.	**Pass**	N/A
16	Cancel Button	Mouse Click	N/A	Page changes and user is navigated back to main admin menu.	Page changes and user is navigated back to main admin menu.	**Pass**	N/A
Create Candidate Page							
17	Create Candidate	Entered data, submitted form.	Name: Graeme Email: g@gmail.com	Cand should be created, added to database.	Cand is added to database, created successfully.	**Pass**	N/A
18	Create Candidate (Blank Name)	Entered email, left name blank, submitted form.	Email: g@gmail.com	Program should not insert cand, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
19	Create Candidate (Blank Email)	Entered name, left email blank, submitted form.	Name: Graeme1	Program should not insert cand, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
20	Create Candidate (Blank Name + Email)	Left both name and email blank, submitted form.	N/A	Program should not insert cand, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
21	Cancel Button	Mouse Click	N/A	Page changes and user is navigated back to main admin menu.	Page changes and user is navigated back to main admin menu.	**Pass**	N/A
Create Election Page							
22	Create Election	Entered data, submitted form.	Start: 00:00 20/01/2020 End: 00:00 20/01/2021	Create a new election and add it to the database	Election is added to database, created successfully.	**Pass**	N/A
23	Create Election (Blank start date)	Entered end date, left start date blank, submitted form.	Start: 00:00 20/01/2020 End:	Program should not insert election, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
24	Create Election (Blank finish date)	Entered start date, left end date blank, submitted form.	Start: End: 00:00 20/01/2021	Program should not insert election, should return error.	Error returned to user, passed successfully.	**Pass**	N/A
15	Create Election (Wrong Format)	Entered start and end date in wrong format, submitted form.	Start: 20-01/2020 3:10am End: 02/10/2020 15:00	Program should not insert election, should return error.	Election successfully doesn't insert election, error is not displayed to user.	**Partial Fail**	Handle ValueError more gracefully.
26	Create Election (Blank)	Left start and end date blank, submitted form.	N/A	Program should not insert election, should return error.	Error returned to user, passed successfully.	**Pass**	N/A

27	Cancel Button	Mouse Click	N/A	Page changes and user is navigated back to main admin menu.	Page changes and user is navigated back to main admin menu.	**Pass**	N/A
Candidate Application							
25	Submit an Application	Submit an Application	Election Date and Time: 1 Position: 19 - Faculty of Engineering and Science Officer Candidate: 495 - Isaiah Waistall	Create new application and add it to the database.	Application is added to database, created successfully.	**Pass**	N/A
26	Submit an Application (Empty)	Submit an empty candidate application	N/A	Do not proceed, display error.	Displayed error.	**Pass**	N/A
27	Submit an Application (Without Candidate)	Submit an application without candidate selected	Election Date and Time: 1 Position: 19 - Faculty of Engineering and Science Officer	Do not proceed, display error.	Displayed error.	**Pass**	N/A
31	Submit an Application (Position already at max)	Try to add more than 4 candidates to a position	Try to select position for full position.	The position shouldn't be displayed in the drop-down menu	Position is not listed.	**Pass**	N/A
32	Submit an Application (Single candidate applying for multiple positions)	Try to make a single candidate apply to multiple positions.	Try to get candidate to apply for another position	Candidate should not be in the list of candidates as they have already applied for a different position	Candidate did not appear in the dropdown.	**Pass**	N/A
33	Cancel Button	Mouse Click	N/A	Page changes and user is navigated back to main admin menu.	Page changes and user is navigated back to main admin menu.	**Pass**	N/A
View results							
34	View Results	View results for selected position.	Election: 1 Position: 1 - President	Results displayed correctly.	The results are displayed correctly.	**Pass**	N/A
35	View Results (Blank Dropdowns)	Left both dropdowns blank.	N/A	No results displayed or page changes.	No results displayed or page changes.	**Pass**	N/A
36	View Results (Blank Election)	Selection the position but not the election	Position: 1 - President	No results displayed or page changes.	No results displayed or page changes.	**Pass**	N/A
37	View Results (Blank Position)	Selection the election but not the position	Election: 1	No results displayed or page changes.	No results displayed or page changes.	**Pass**	N/A
39	View Results (Change on new votes)	Add more votes and re-check results.	Position: 4 - Tertiary GSU Officer	The results should change.	Results change.	**Pass**	N/A
40	Calculation Check (Total Votes)	N/A	N/A	Four candidates, 5 votes should equal 20 votes.	Vote count correct.	**Pass**	N/A

41	Calculation Check (Winner Change)	Change the number of votes so that the winner changes from Yovonnda Waring to Stephi Quernel	Add more votes for Stephi Quernel	Winner changes to Stephi.	The winner changes correctly.	**Pass**	N/A
42	**Graph View**	**Mouse Click**	**N/A**	**Graph displayed correctly.**	**Graph displayed correctly.**	**Pass**	**N/A**
43	**Graph View (Graph Change)**	**Mouse Click**	**New votes.**	**Values change on graph.**	**Values changed correctly on graph.**	**Pass**	**N/A**
44	Back Button	Mouse Click	N/A	Page changes and user is navigated back to main admin menu.	Page changes and user is navigated back to main admin menu.	**Pass**	N/A
Student Login Page							
45	Login (Correct Details)	Login with correct details.	User: Test Password: Test	Login should succeed, navigate to vote page.	Login succeeded, navigated to vote page.	**Pass**	N/A
46	Login (Incorrect Details)	Login with incorrect details.	User: Test Password: Password	Error should be displayed to user.	Error was displayed to user.	**Pass**	N/A
47	Login (Blank Details)	Attempt login with blank fields.	N/A	Error should be displayed to user.	Error was displayed to user.	**Pass**	N/A
Select Position Page							
49	Vote Prep (Blank Election)	Leave election empty.	N/A	Display an error to user.	Displayed an error to the user.	**Pass**	N/A
50	Vote Prep (Select Position)	Select from dropdown.	Position: 4 - Tertiary GSU Officer	Navigate to voting page for position.	Navigated to correct voting page.	**Pass**	N/A
51	Vote Prep (Leave Blank Position)	Leave position empty.	N/A	Display an error to user.	Displayed an error to the user.	**Pass**	N/A
52	Vote Prep (Vote Twice For Same Position)	Ensure student can't vote for same position twice.	N/A	Position 4 should not appear in dropdown.	Position did not appear in dropdown.	**Pass**	N/A
53	Back Button	Mouse Click	N/A	Navigate back to login page.	Navigate back to login page.	**Pass**	N/A
54	View Results Button	Mouse Click	N/A	Navigate to results page.	Navigated to results page.	**Pass**	N/A
Voting							
55	Vote (Select 1st Choice)	Select first choice	Select first choice, mouse click confirm.	Choice saved, proceeds to next choice.	Results saved, proceeds on.	**Pass**	N/A
56	Vote (1st Choice Empty)	Leave first choice empty	Leave first choice empty, mouse click confirm.	Do not proceed to next choice.	Does not proceed to next choice.	**Pass**	N/A
57	Vote (Select 2nd Choice)	Select second choice	Select second choice, mouse click confirm.	Choice saved, proceeds to next choice.	Results saved, proceeds on.	**Pass**	N/A
58	Vote (2nd Choice Empty)	Leave second choice empty	Leave second choice empty, mouse click confirm.	Do not proceed to next choice.	Does not proceed to next choice.	**Pass**	N/a

59	Vote (Select 3rd Choice)	Select third choice	Select third choice, mouse click confirm.	Choice saved, proceeds to next choice.	Results saved, proceeds on.	**Pass**	N/A
60	Vote (3rd Choice Empty)	Leave third choice empty	Leave third choice empty, mouse click confirm.	Do not proceed to next choice.	Does not proceed to next choice.	**Pass**	N/A
61	Vote (Select 4th Choice)	Select fourth choice	Select fourth choice, mouse click confirm.	Choice should save and submit results.	Results submitted.	**Pass**	N/A
62	Vote (4th Choice Empty)	Leave fourth choice empty	Leave fourth empty, mouse click confirm.	Should not submit results.	The confirm button didn't cast any votes	**Pass**	N/A
63	Vote (Only First Preference)	Set second preference to N//A	First preference only.	Choice saved, proceeds to next choice.	Results saved, proceeds on. Graphical bug detected.	**Partial Fail**	Correct graphical bug.
Student Results View / Student View Position Results							
64	Student Position Results (Blank Position)	Leave position blank, select button.	N/A	Do not proceed, no error.	No results were shown, page did not proceed.	**Pass**	N/A
65	Student Position Results (Match admin, include correct percentage)	Check results are as expected.	N/A	Results displayed should be correctly.	Results displayed are correct.	**Pass**	N/A
66	Back button	Mouse Click	N/A	Navigate back to login page.	Navigate back to login page.	**Pass**	N/A
Student View Election Results							
67	View Election Results (All Positions Listed)	Check results are as expected.	N/A	Results displayed should be correctly.	Results displayed are correct.	**Pass**	N/A
68	View Election Results (Correct Users)	Check results are as expected.	N/A	Results displayed should be correctly.	Results displayed are correct.	**Pass**	N/A
69	Back button	Mouse Click	N/A	Navigate back to login page.	Navigate back to login page.	**Pass**	N/A
70	Vote Export	Mouse Click	N/A	Results should be exported to text file.	Results exported to text file.	**Pass**	N/A

Test 15 - Corrective Fix Applied

Corrective Code Applied To Fix Test:

```
try:
      start_date_time = datetime.strptime((self.ui_builder.get_object(
          'bkend_create_election_start_txtbx'
      ).get()), "%H:%M %d/%m/%Y")
      end_date_time = datetime.strptime((self.ui_builder.get_object(
          'bkend_create_election_end_txbx'
      ).get()), "%H:%M %d/%m/%Y")
    except ValueError:
      # If not valid, alert user.
      self.ui_builder.get_object(
          'bkend_create_election_error_lbl'
      ).configure(text="Error: Invalid date input.")
```

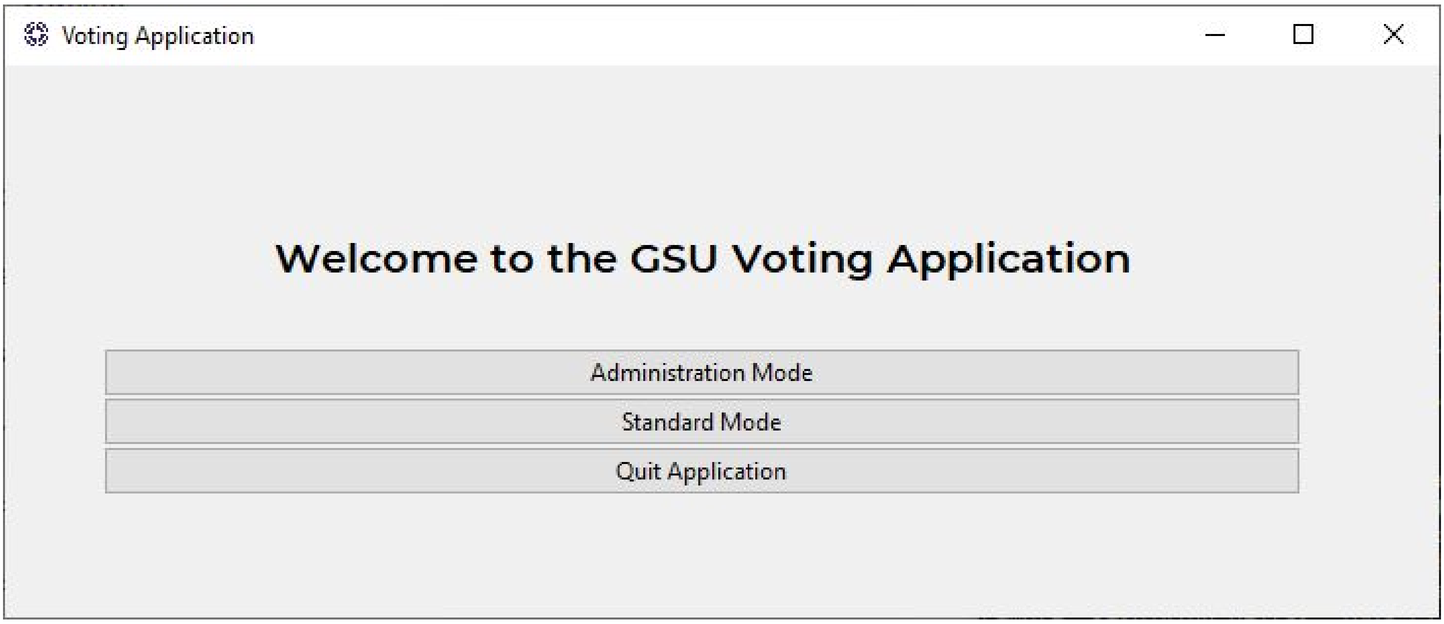
Test 63

Unfortunately, this test was identified too late within the development process. While a fix was attempted, the root cause of the duplicated dropdown boxes has yet to be identified.

Screenshots Demonstrating Implementation

Feature A.1

Startup Menu

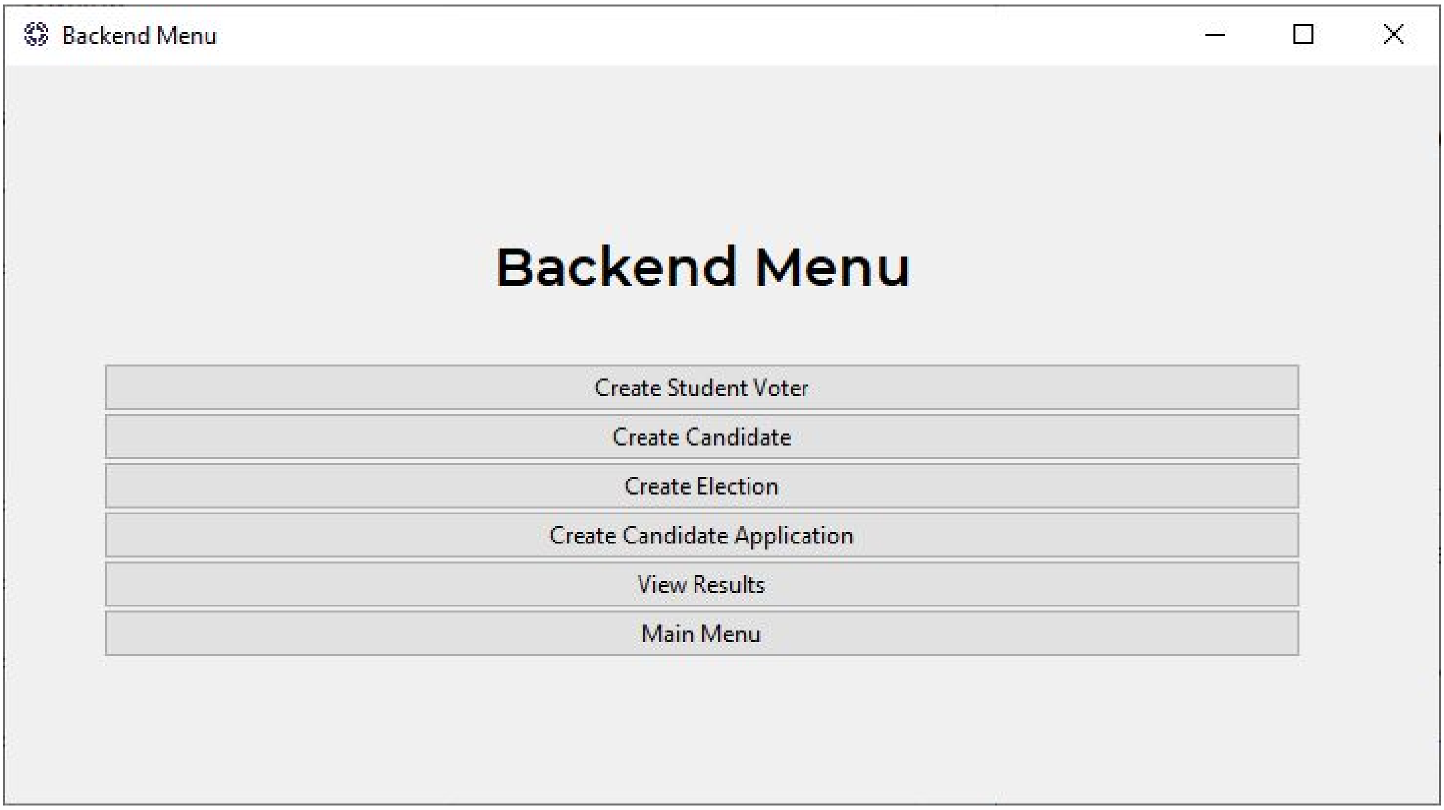


Voting Application

Welcome to the GSU Voting Application

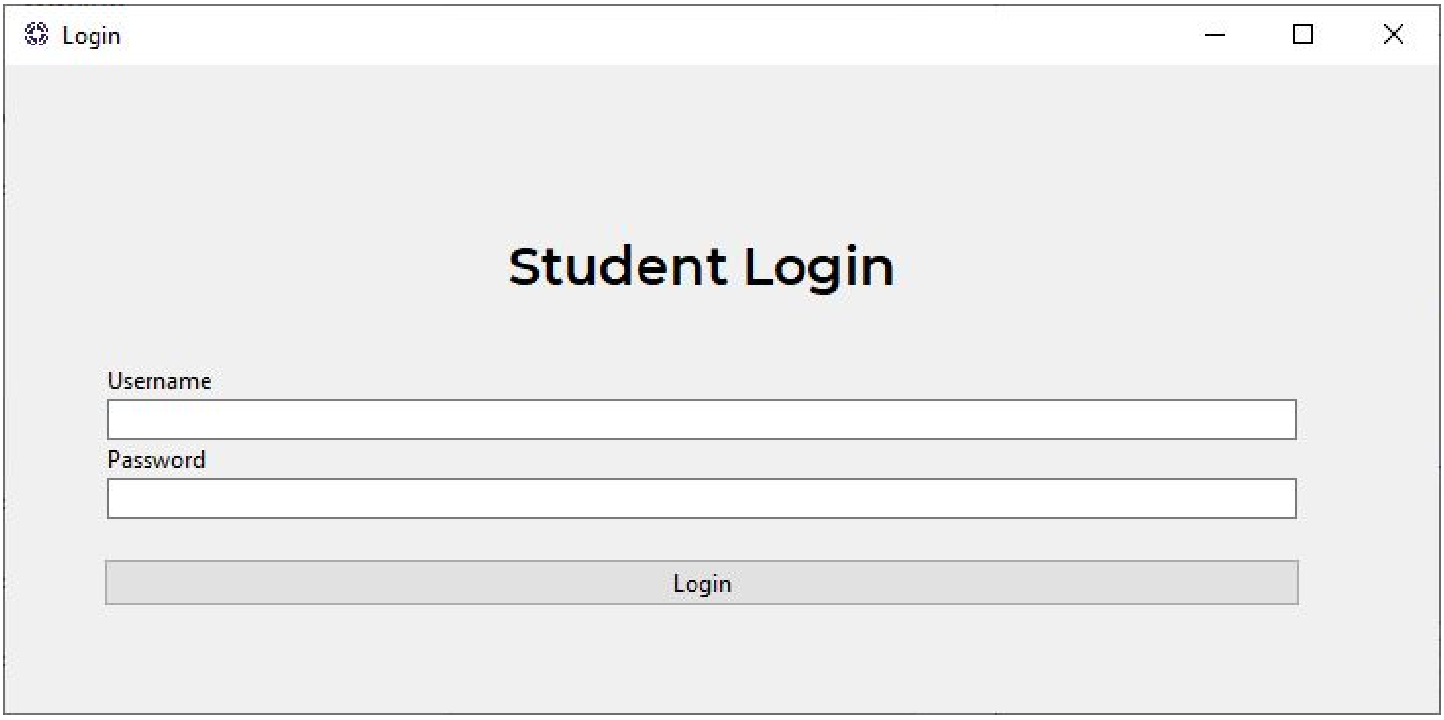
In our application there are three main options, Administrator (backend functionality), Standard (Student login) and quit application. Clicking Quit application will terminate the application

Backend Menu (Admin Menu)



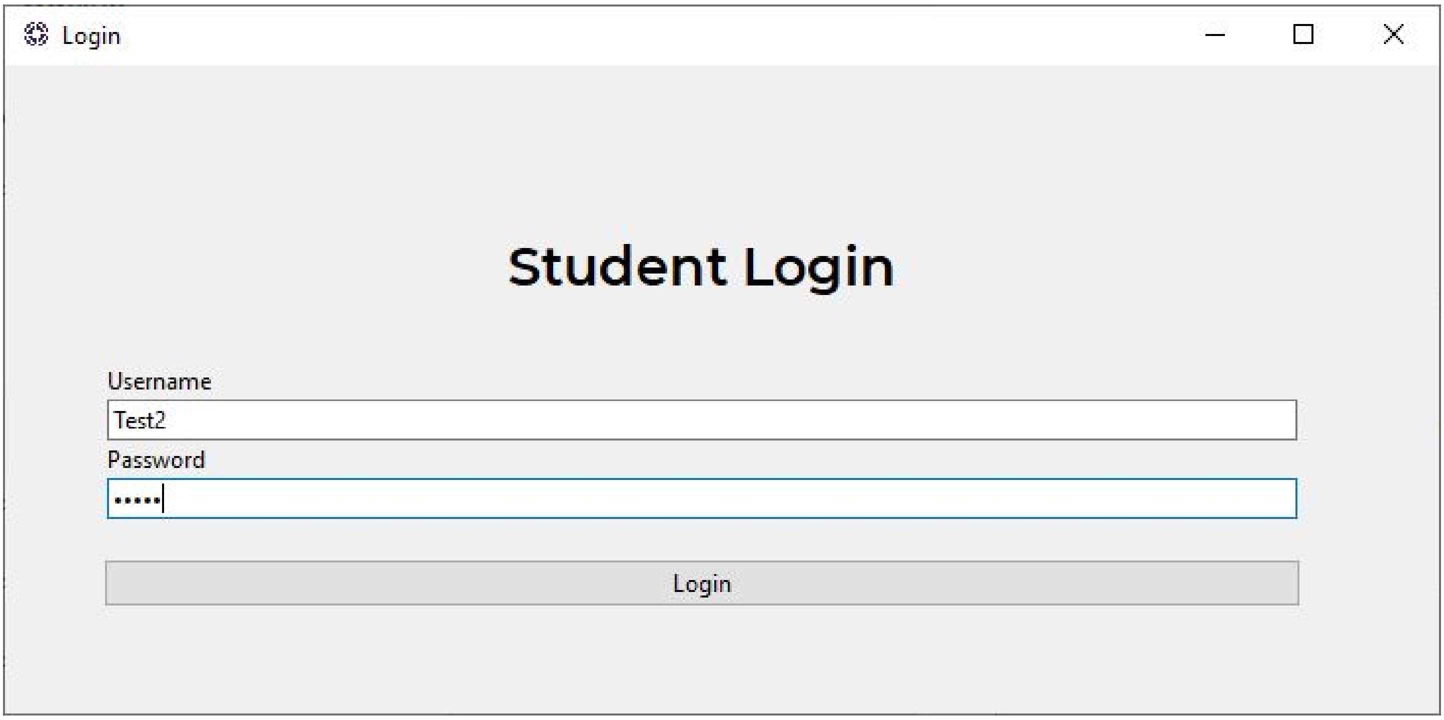
This is the backend or administrator menu.

Student Login Menu



Clicking standard mode will prompt you to enter your student login.

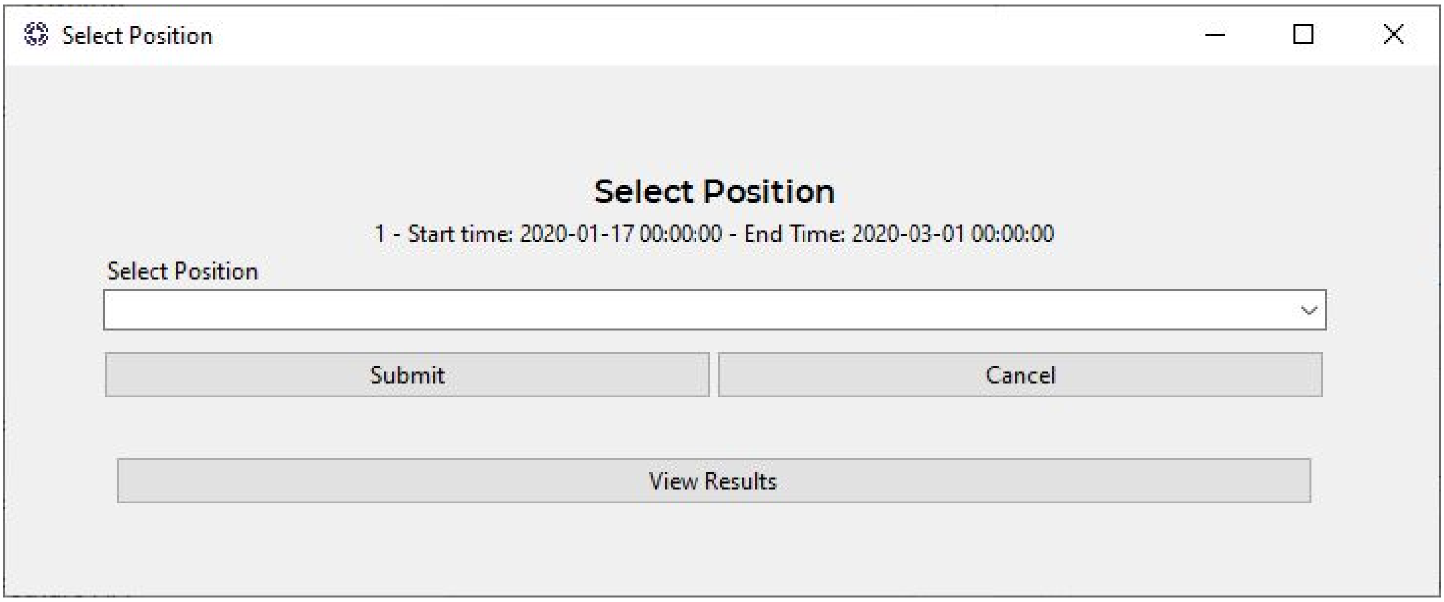
Feature A.2



Student Login

Username: Test2

Password: •••••

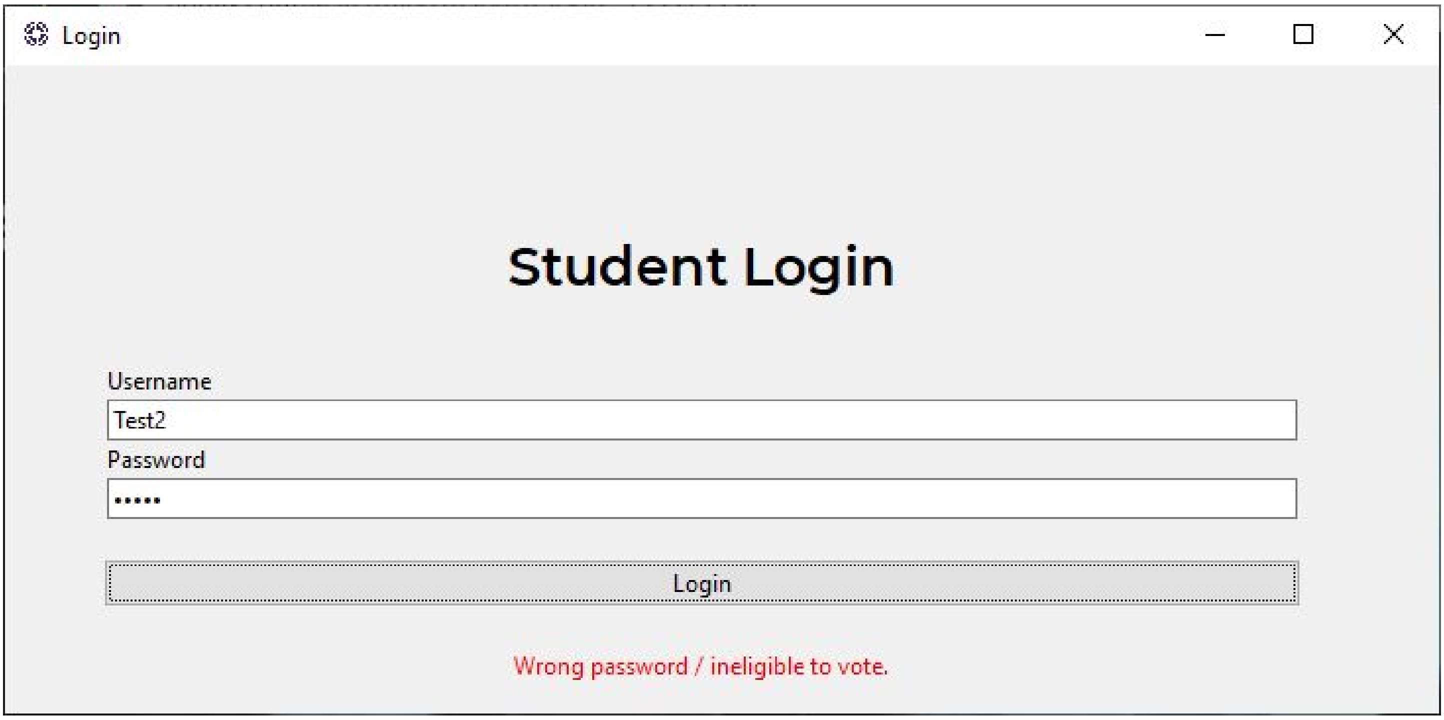


Select Position

1 - Start time: 2020-01-17 00:00:00 - End Time: 2020-03-01 00:00:00

Select Position:

Entering Test2 in both username and password will allow a student to log in and then select a position to vote for, as long as that position has not already been voted for by that student and there are four candidates for that particular position. If you type a wrong password or an unknown (ineligible) username then an error will be shown, and you can remove your username and password and try again. Shown in the screenshot below.

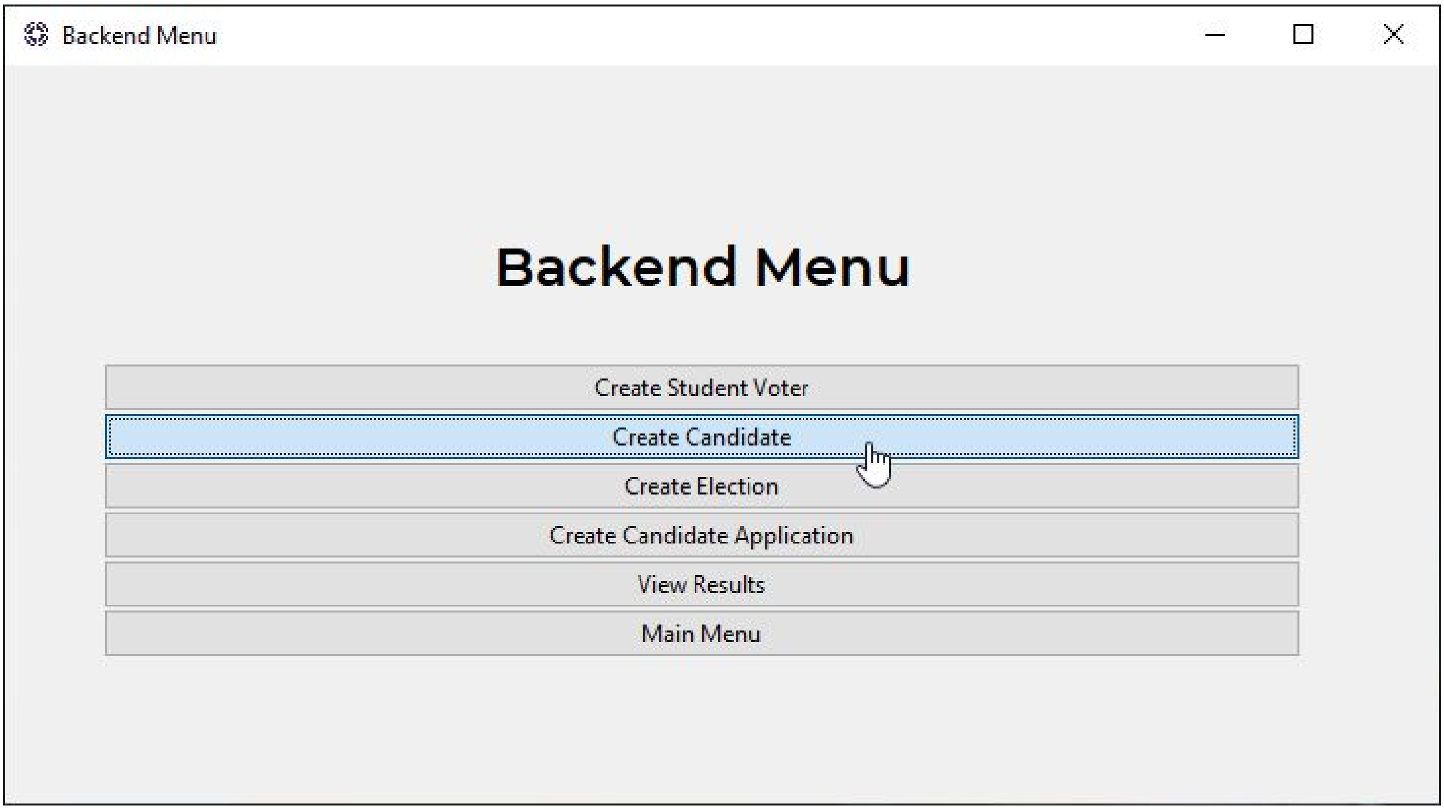


Test2

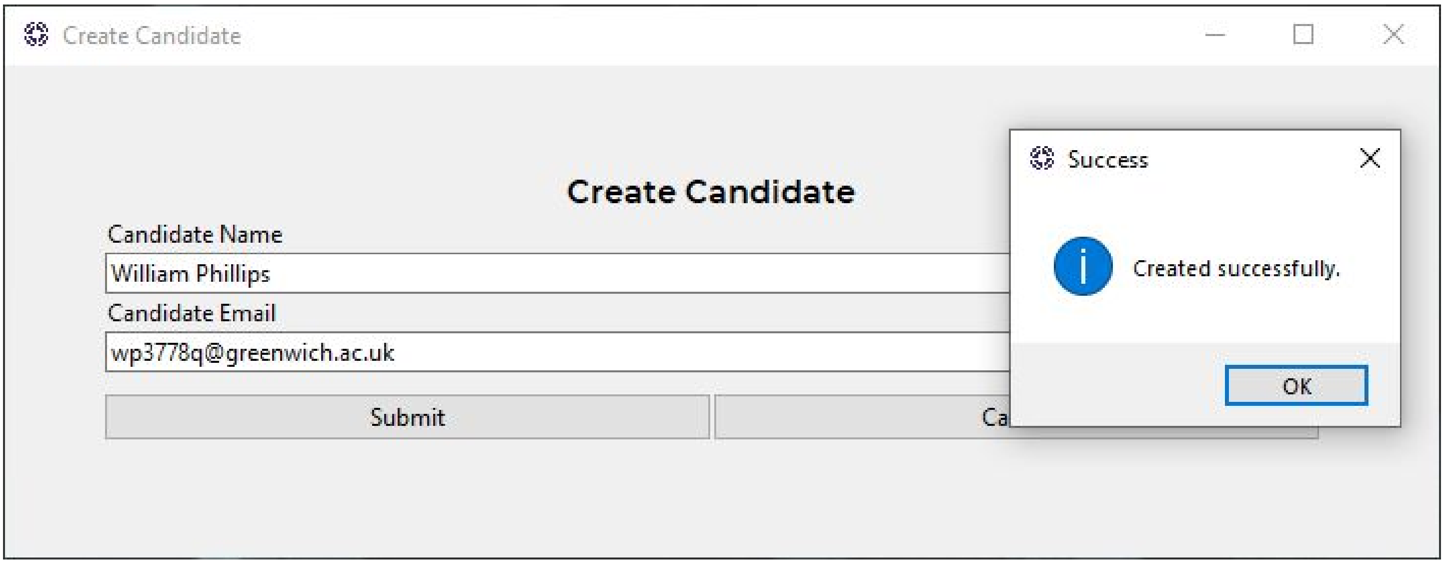
•••••

Wrong password / ineligible to vote.

Feature A.3



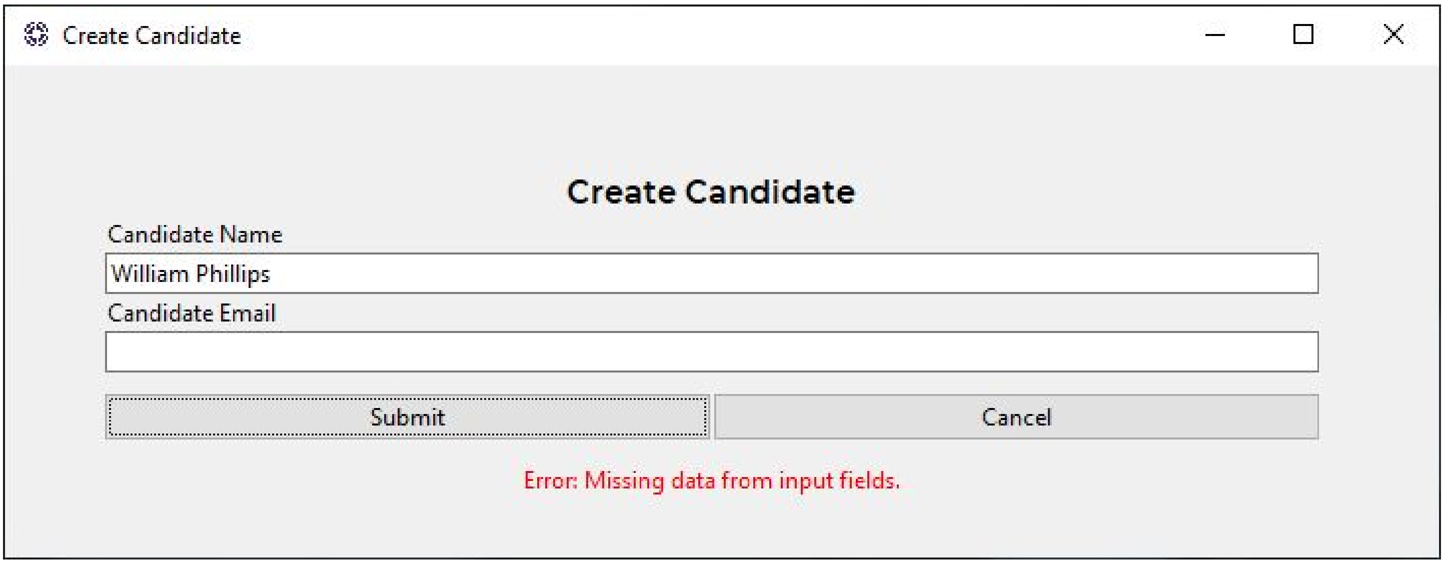
Click on create candidate



Candidate Name
William Phillips
Candidate Email
wp3778q@greenwich.ac.uk

Created successfully.

Create a candidate and a popup will tell you that its been successful (if its unique)



Create Candidate

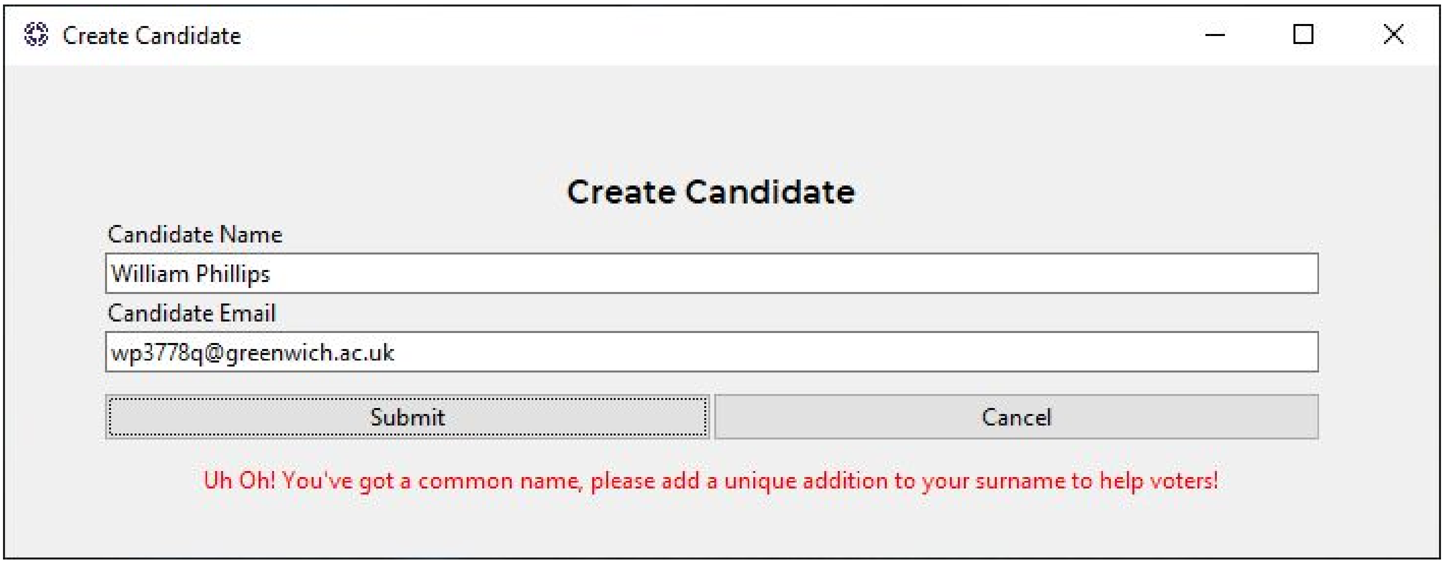
Candidate Name

William Phillips

Candidate Email

Error: Missing data from input fields.

An error message will be displayed if missing data.



Create Candidate

Candidate Name

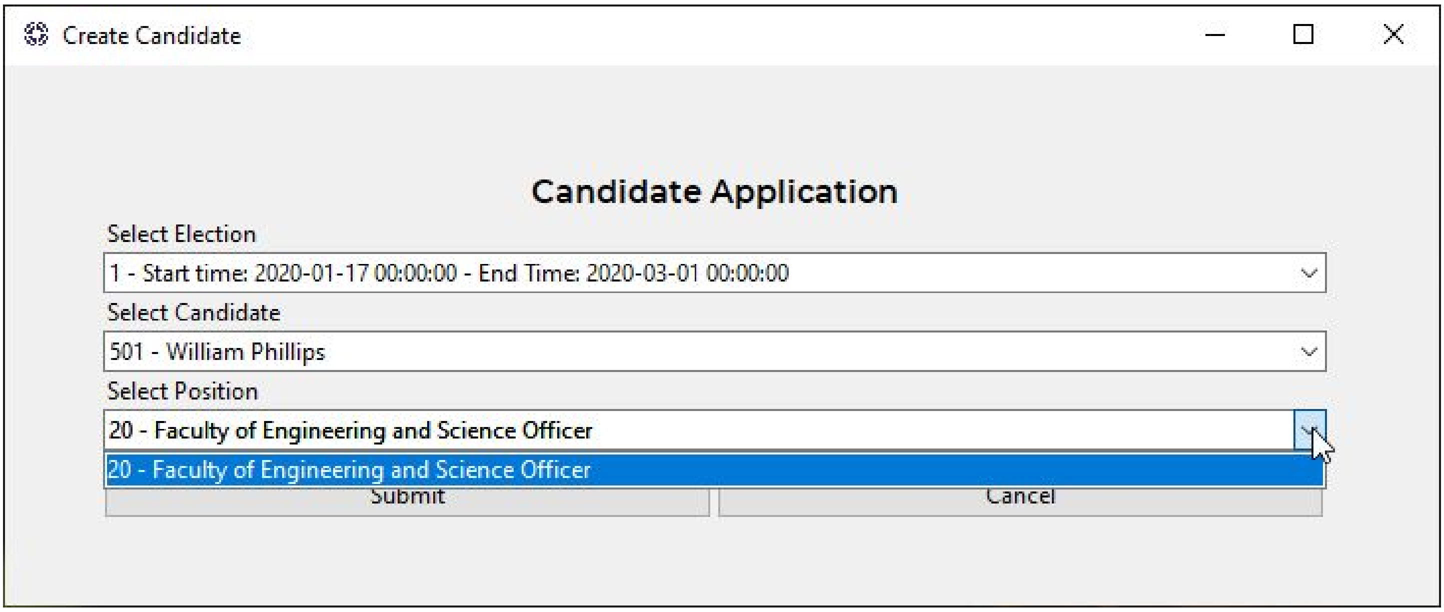
William Phillips

Candidate Email

wp3778q@greenwich.ac.uk

Uh Oh! You've got a common name, please add a unique addition to your surname to help voters!

An error message will be displayed if there is a duplicate name.



Candidate Application

Select Election: 1 - Start time: 2020-01-17 00:00:00 - End Time: 2020-03-01 00:00:00

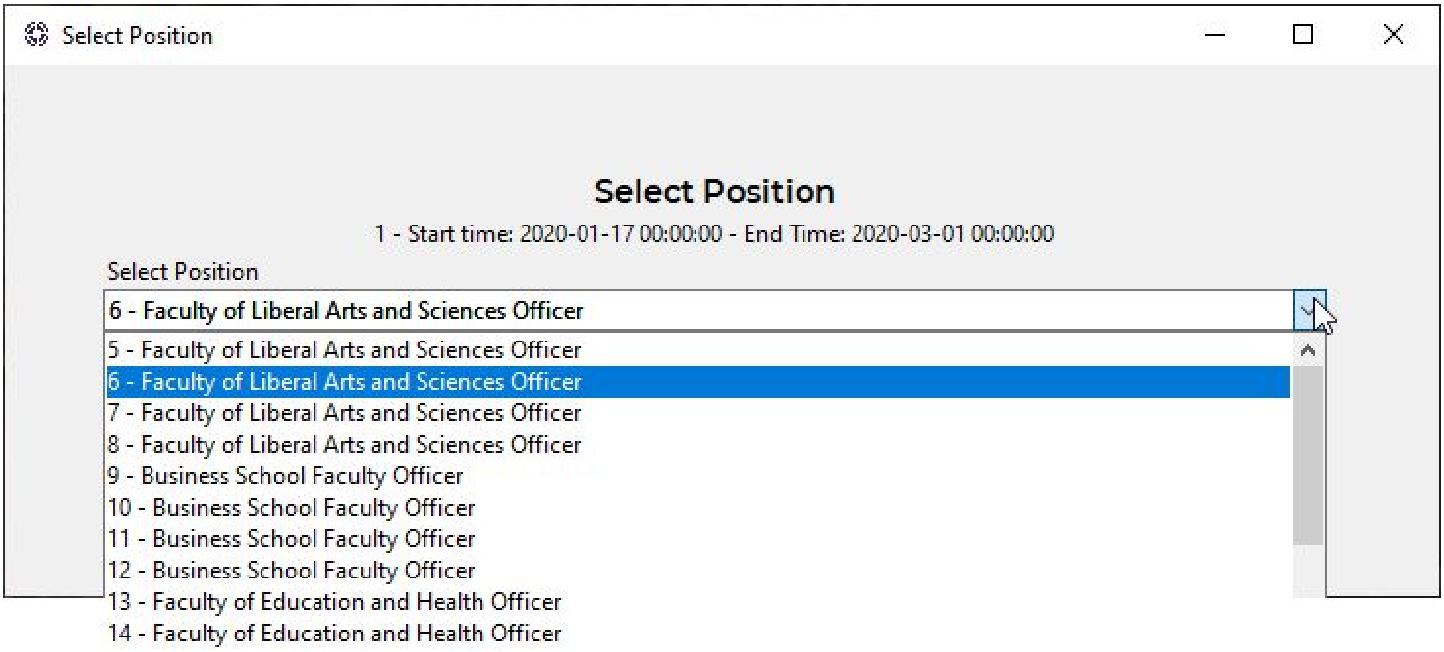
Select Candidate: 501 - William Phillips

Select Position: 20 - Faculty of Engineering and Science Officer

20 - Faculty of Engineering and Science Officer

This candidate can then pick what position for what election they want to enter into. After this has been submitted then students can vote for them under that position (as long as there are 4 candidates for that position. A candidate can only apply for a single position in the election and can only apply for positions that don’t already have 4 candidates.

Feature A.4



Select Position

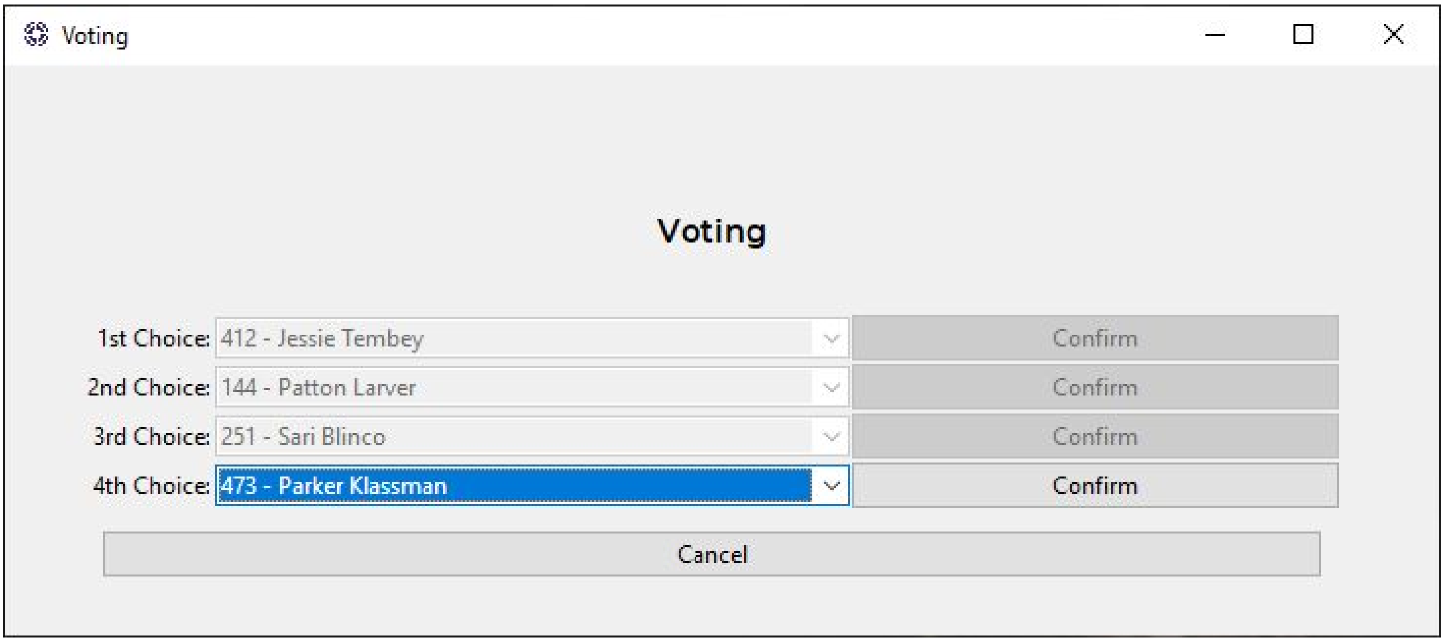
1 - Start time: 2020-01-17 00:00:00 - End Time: 2020-03-01 00:00:00

Select Position

6 - Faculty of Liberal Arts and Sciences Officer

5 - Faculty of Liberal Arts and Sciences Officer
6 - Faculty of Liberal Arts and Sciences Officer
7 - Faculty of Liberal Arts and Sciences Officer
8 - Faculty of Liberal Arts and Sciences Officer
9 - Business School Faculty Officer
10 - Business School Faculty Officer
11 - Business School Faculty Officer
12 - Business School Faculty Officer
13 - Faculty of Education and Health Officer
14 - Faculty of Education and Health Officer

First you have to select the position you are going to vote for. The dropdown is restricted to prevent candidates from applying for positions that don't have 4 candidates available for the position, and also prevents them from being able to apply for positions they've previously voted for. This is why in the screenshot they're unable to vote for the first 4 positions.



Voting

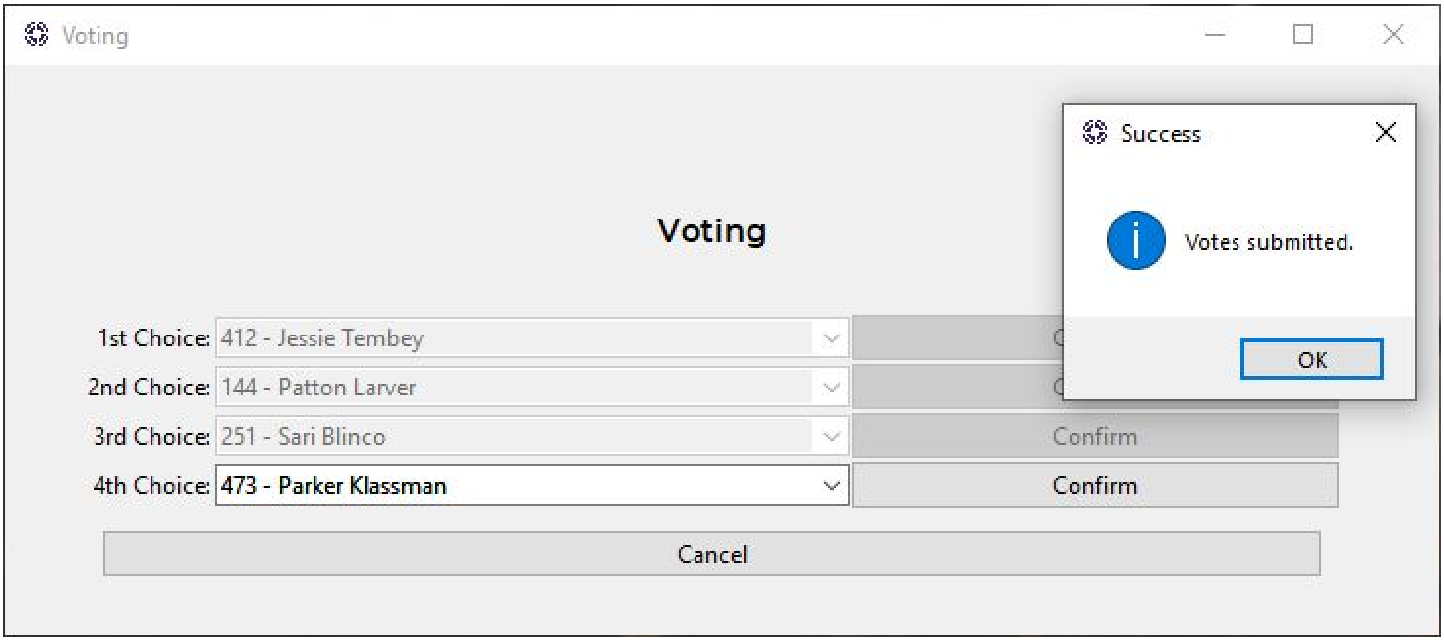
1st Choice: 412 - Jessie Tembey

2nd Choice: 144 - Patton Larver

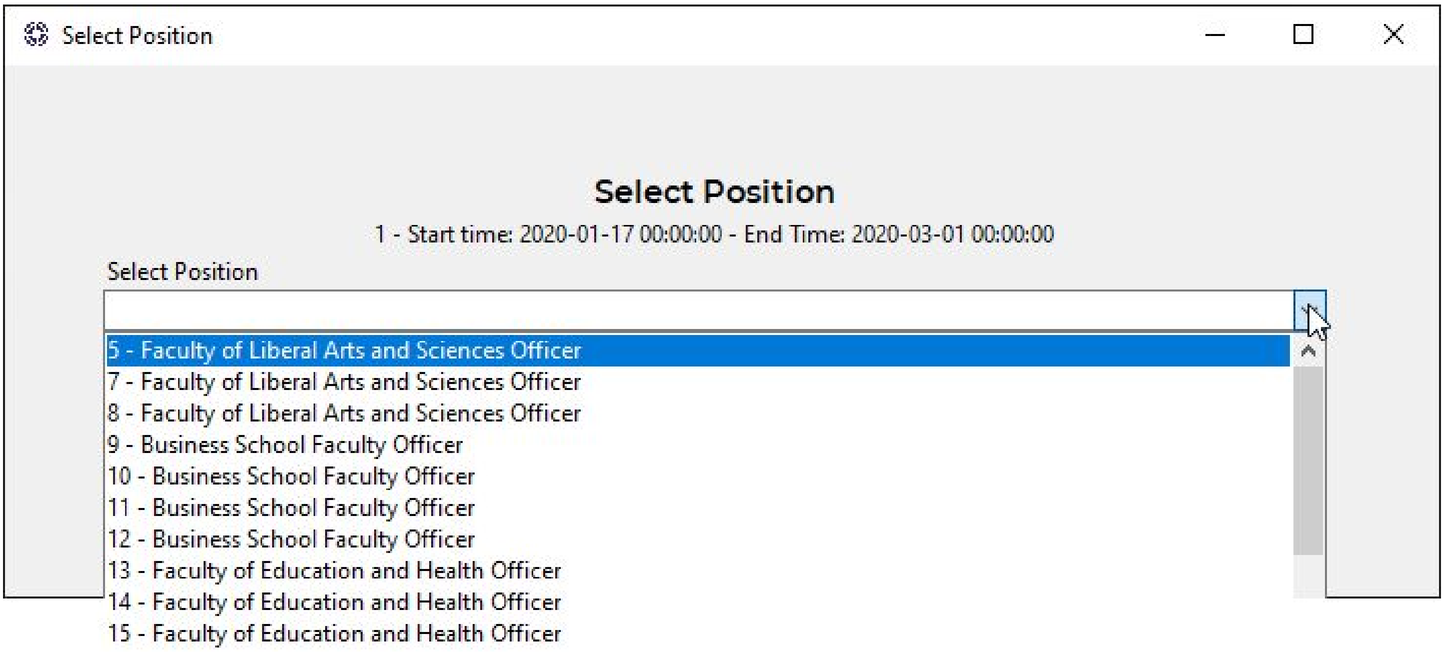
3rd Choice: 251 - Sari Blinco

4th Choice: 473 - Parker Klassman

Then you must select one of the four candidates for each position, if you don't want to vote for someone then you can have a vote of N/A. After you confirm the last candidate it will submit the votes for them to be counted. Clicking ok on the pop up box will return you to the login screen.



Voting
Voting
1st Choice: 412 - Jessie Tembey
2nd Choice: 144 - Patton Larver
3rd Choice: 251 - Sari Blinco
4th Choice: 473 - Parker Klassman
Confirm
Confirm
Cancel
Success
Votes submitted.
OK



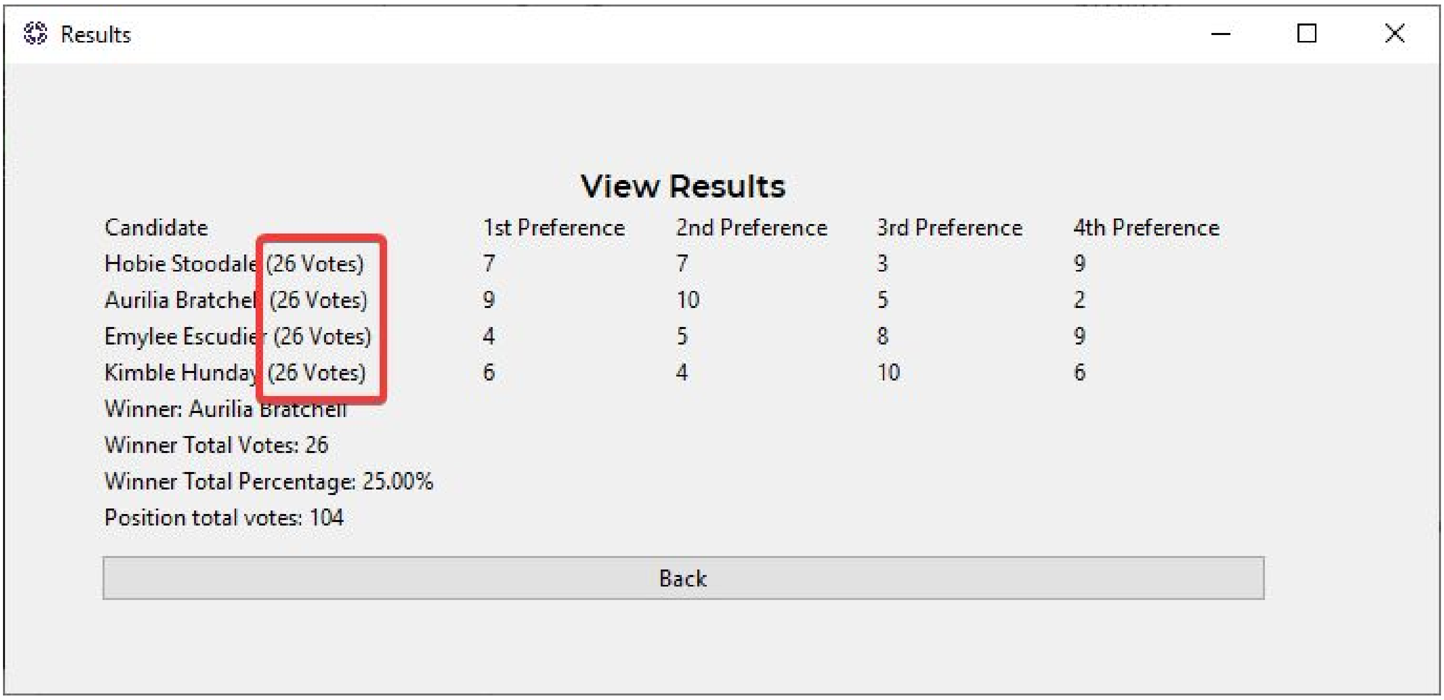
Select Position
Select Position
1 - Start time: 2020-01-17 00:00:00 - End Time: 2020-03-01 00:00:00
Select Position
5 - Faculty of Liberal Arts and Sciences Officer
7 - Faculty of Liberal Arts and Sciences Officer
8 - Faculty of Liberal Arts and Sciences Officer
9 - Business School Faculty Officer
10 - Business School Faculty Officer
11 - Business School Faculty Officer
12 - Business School Faculty Officer
13 - Faculty of Education and Health Officer
14 - Faculty of Education and Health Officer
15 - Faculty of Education and Health Officer

If you sign in with the same student it will not allow you to vote for the same election, but this student can vote for any other position they haven't yet voted for.

Feature A.5

```
def calculate_highest_votes(self, votes, candidates):
        # Create 0 values.
        candidate_one_count = 0
        candidate_two_count = 0
        candidate_three_count = 0
        candidate_four_count = 0

        results = {}

        # If vote selects candidate, add 1 to their count.
        for i in range(len(votes)):
            if (candidates[0][1] == votes[i][0]):
                candidate_one_count += 1
            elif (candidates[1][1] == votes[i][0]):
                candidate_two_count += 1
            elif (candidates[2][1] == votes[i][0]):
                candidate_three_count += 1
            elif (candidates[3][1] == votes[i][0]):
                candidate_four_count += 1

        # If there are more than zero candidates in the candidates list
        # Add candidate one count to 2d results array.
        if len(candidates) > 0:
            results[candidates[0][0]] = candidate_one_count

        # Repeat adding to 2d array for every additional candidate.
        if (len(candidates) > 1) and (candidate_two_count != 0):
            results[candidates[1][0]] = candidate_two_count

        if (len(candidates) > 2) and (candidate_three_count != 0):
            results[candidates[2][0]] = candidate_three_count

        if (len(candidates) > 3) and (candidate_four_count != 0):
            results[candidates[3][0]] = candidate_four_count

        # Sort 2d array into largest to smallest by candidate.
        highest_values = [i for i, candidate in results.items() if candidate == max(results.values())]

        return(highest_values, results)
```

We count the votes by checking how many times a candidate ID appears in the database, then we will add these up within this function. After this we will compare each candidate together to find out who is the highest, after which we can output the candidate with the most votes (the winning candidate).



View Results

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference
Hobie Stoodale (26 Votes)	7	7	3	9
Aurilia Bratchel (26 Votes)	9	10	5	2
Emylee Escudie (26 Votes)	4	5	8	9
Kimble Hunday (26 Votes)	6	4	10	6

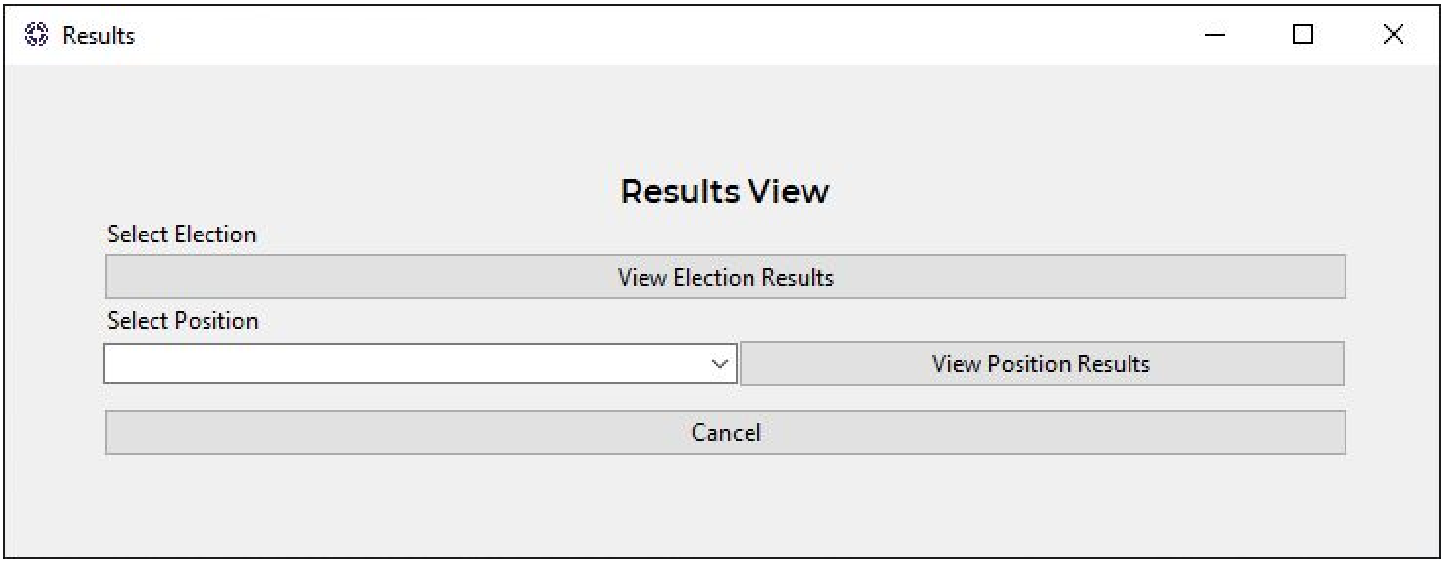
Winner: Aurilia Bratchell

Winner Total Votes: 26

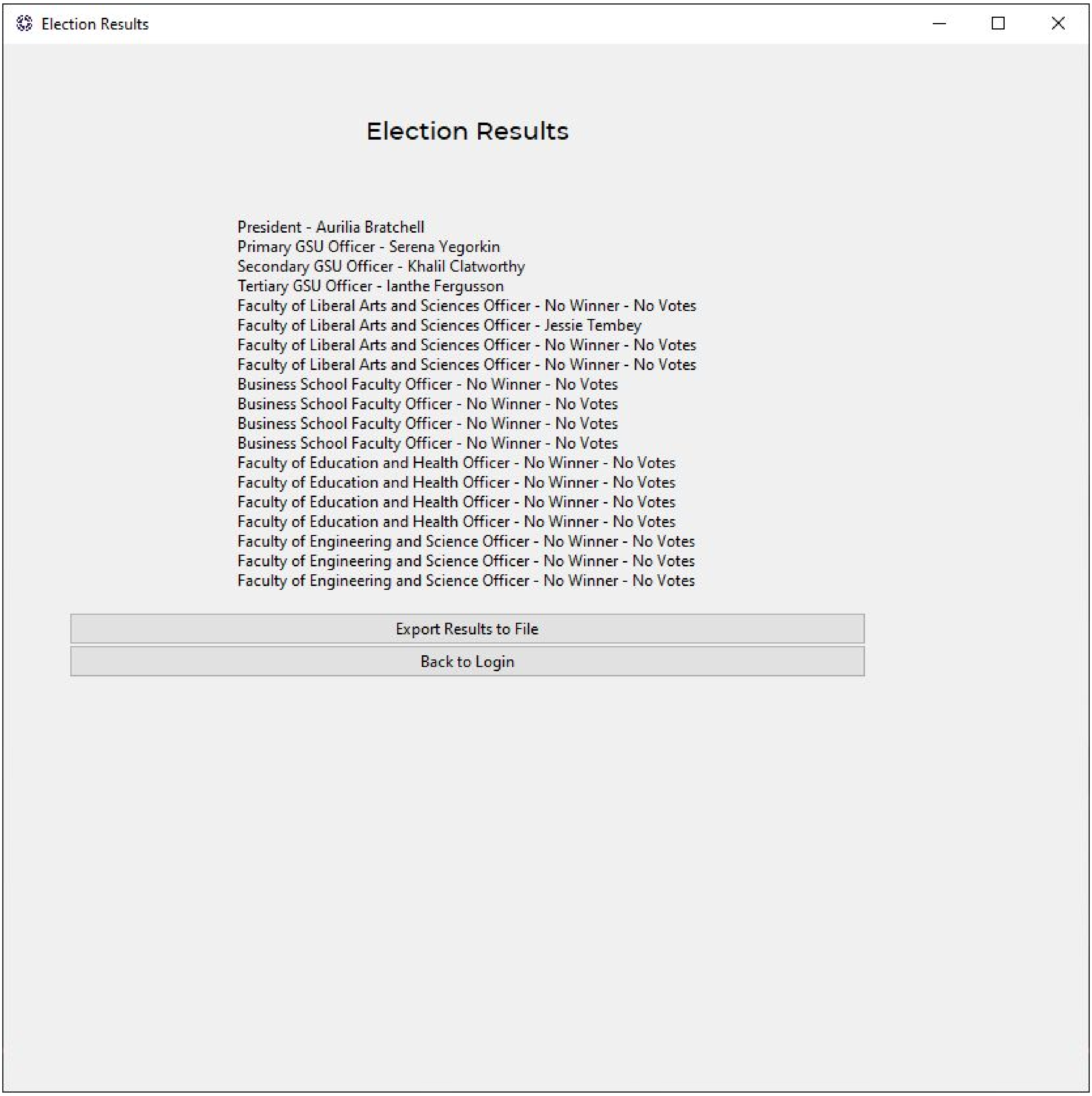
Winner Total Percentage: 25.00%

Position total votes: 104

Feature A.6



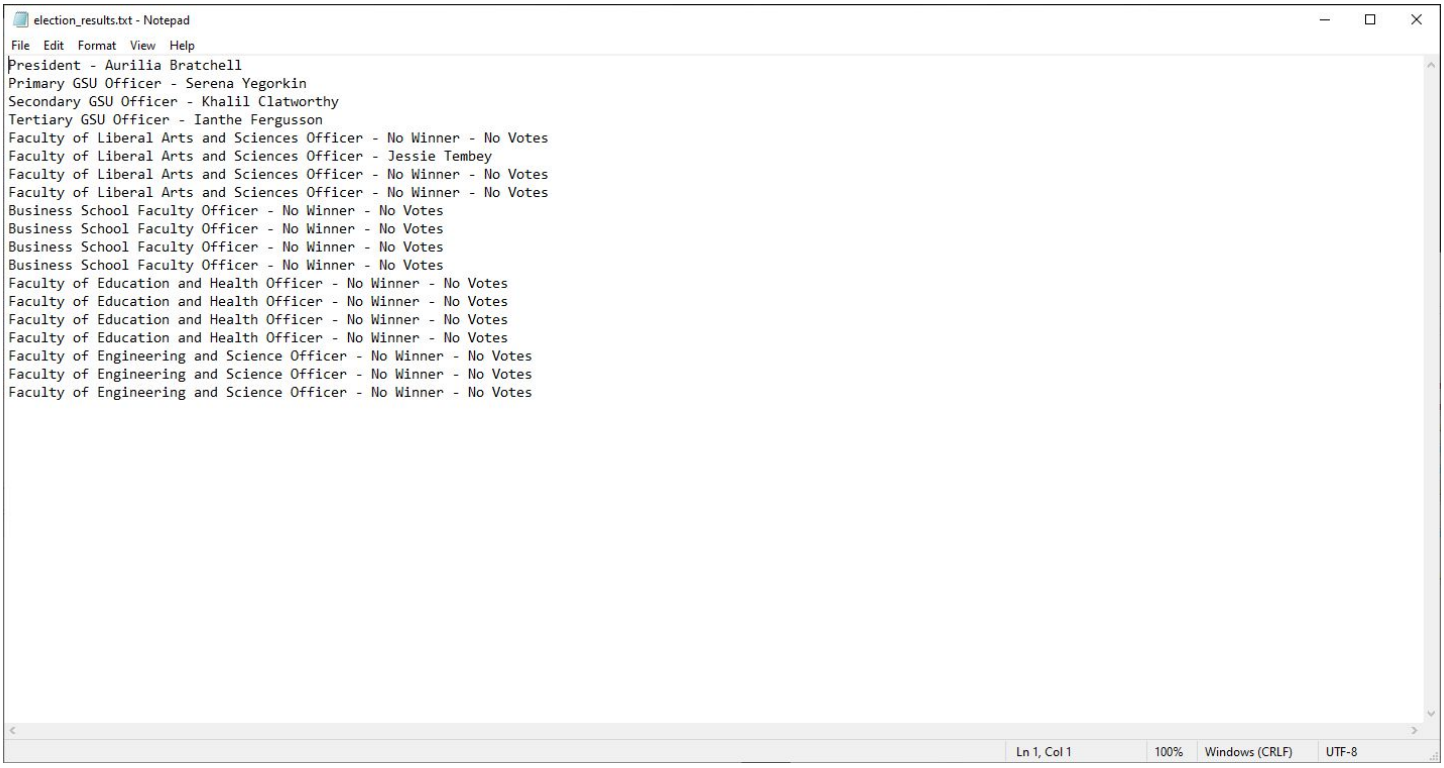
When logged in as a student you can click ‘View Election Results’ to see all positions



Election Results

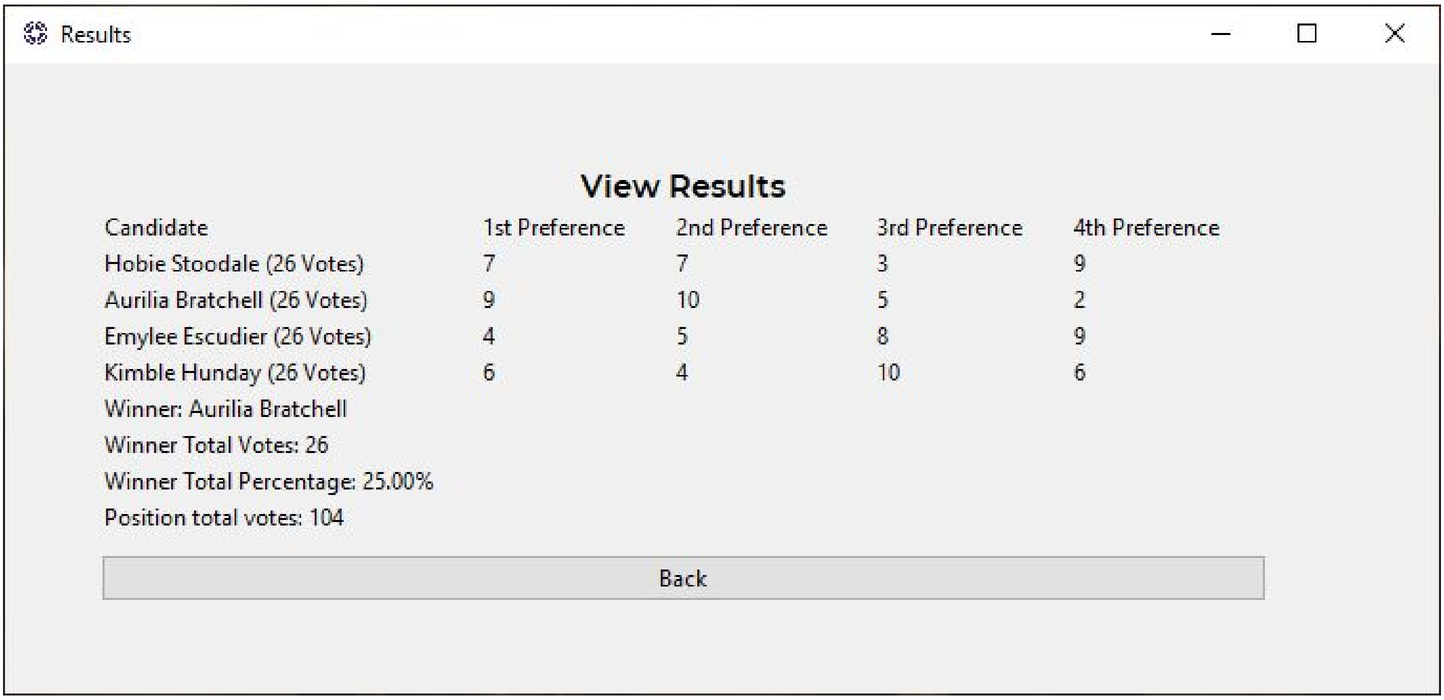
```
President - Aurilia Bratchell
Primary GSU Officer - Serena Yegorkin
Secondary GSU Officer - Khalil Clatworthy
Tertiary GSU Officer - Ianthe Fergusson
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Faculty of Liberal Arts and Sciences Officer - Jessie Tembey
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
```

A summary of the results are able to be displayed. Lots of the positions still require results from students to be able to pick a winner.



```
President - Aurilia Bratchell
Primary GSU Officer - Serena Yegorkin
Secondary GSU Officer - Khalil Clatworthy
Tertiary GSU Officer - Ianthe Fergusson
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Faculty of Liberal Arts and Sciences Officer - Jessie Tembey
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Faculty of Liberal Arts and Sciences Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Business School Faculty Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Education and Health Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
Faculty of Engineering and Science Officer - No Winner - No Votes
```

The summary can be exported to the file as defined in the spec.



View Results

Candidate	1st Preference	2nd Preference	3rd Preference	4th Preference
Hobie Stoodale (26 Votes)	7	7	3	9
Aurilia Bratchell (26 Votes)	9	10	5	2
Emylee Escudier (26 Votes)	4	5	8	9
Kimble Hunday (26 Votes)	6	4	10	6

Winner: Aurilia Bratchell

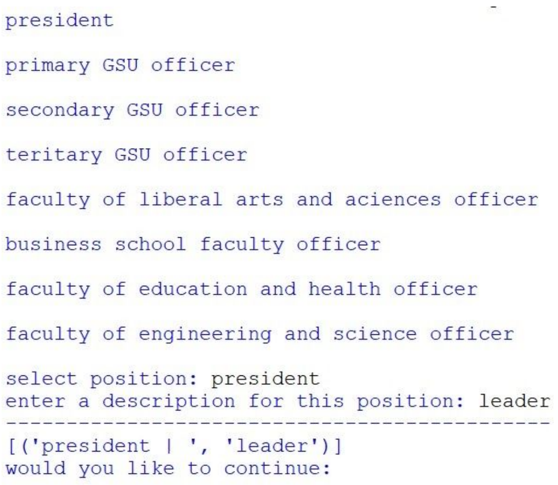
Winner Total Votes: 26

Winner Total Percentage: 25.00%

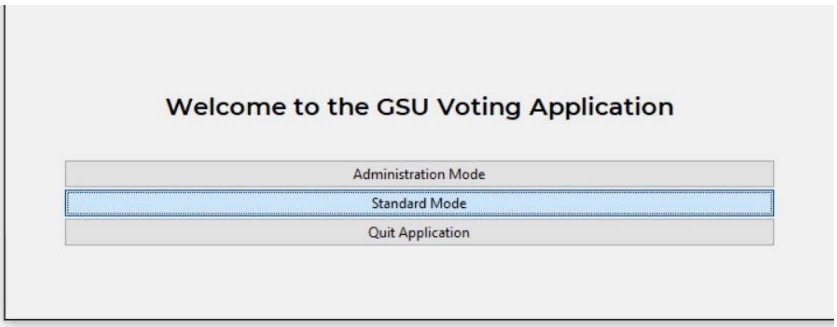
Position total votes: 104

You can also look at the individual position results and find out how many votes each person received as well as total votes and the Winner % of votes.

Feature B.1 - Aiden Silk - Task 1

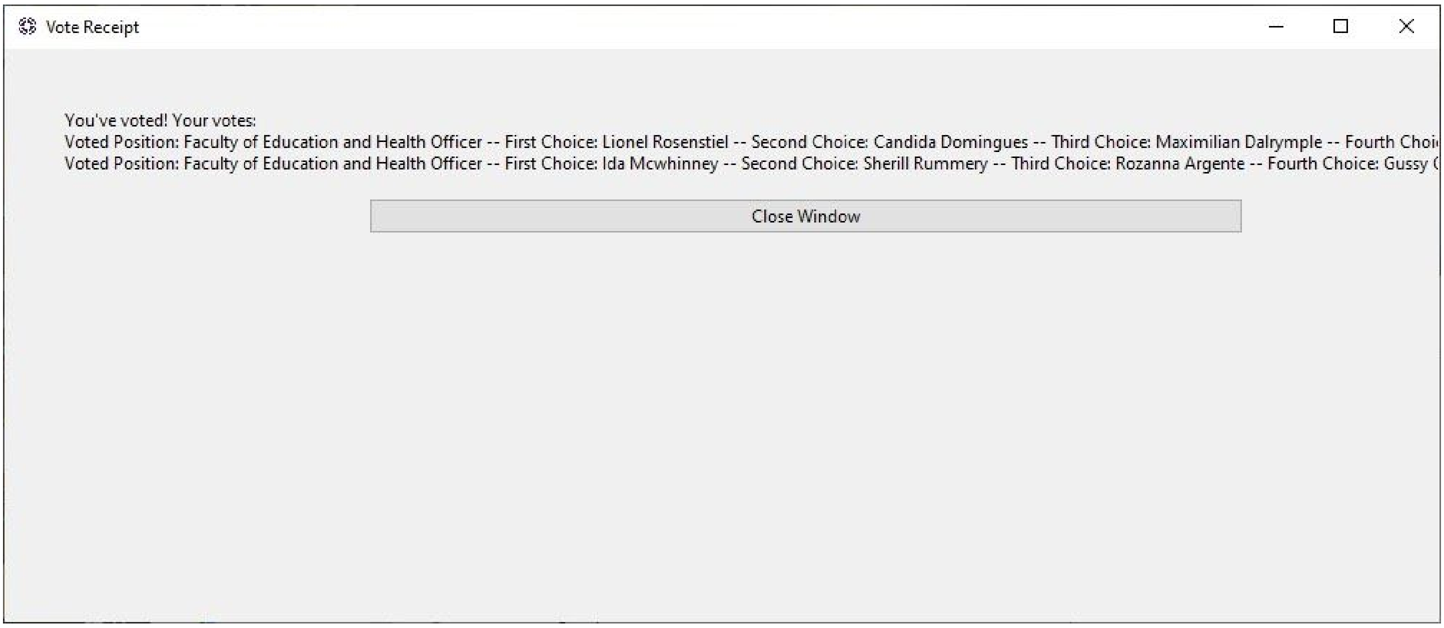


```
president

primary GSU officer

secondary GSU officer

teritary GSU officer

faculty of liberal arts and aciences officer

business school faculty officer

faculty of education and health officer

faculty of engineering and science officer

select position: president
enter a description for this position: leader
---------------------------------------------
[('president | ', 'leader')]
would you like to continue:
```

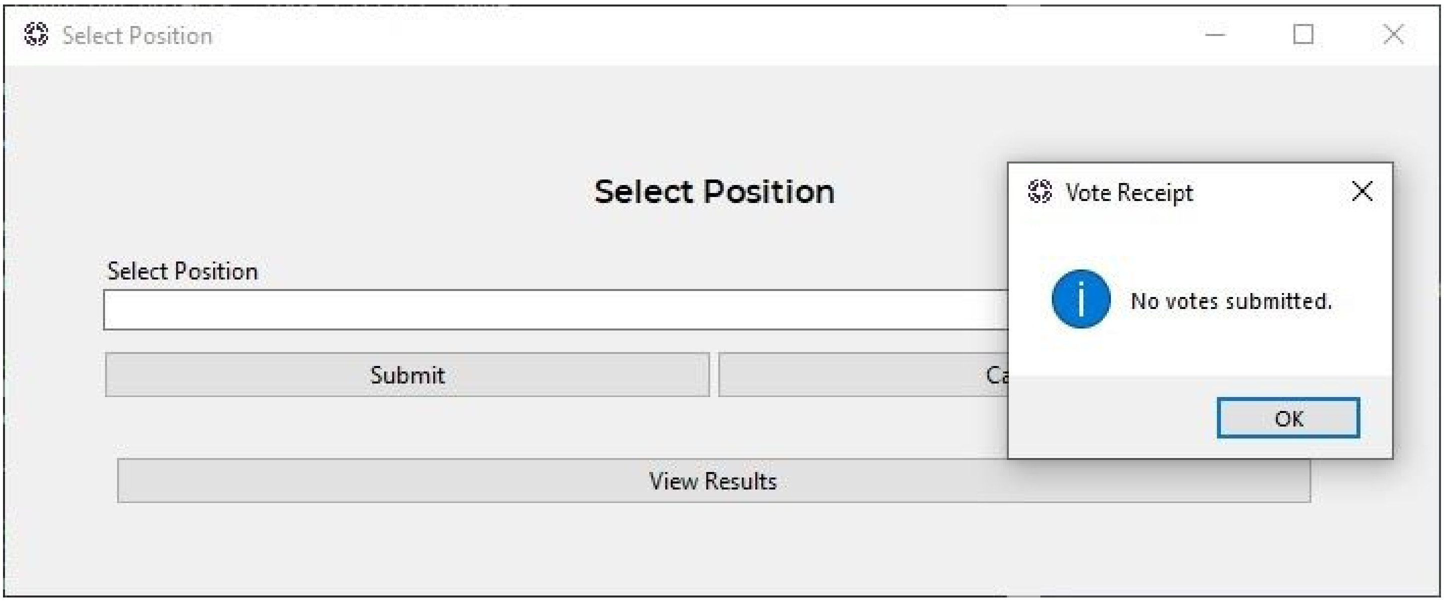


Aiden implemented task 1, he got the script running on program startup, allowing for input via the CLI to interact with it.

Feature B.3 - William Phillips - Task 4

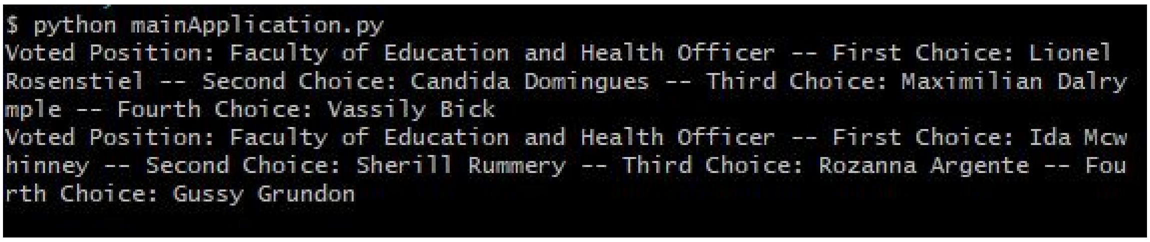


You've voted! Your votes:
Voted Position: Faculty of Education and Health Officer -- First Choice: Lionel Rosenstiel -- Second Choice: Candida Domingues -- Third Choice: Maximilian Dalrymple -- Fourth Choi
Voted Position: Faculty of Education and Health Officer -- First Choice: Ida Mcwhinney -- Second Choice: Sherill Rummery -- Third Choice: Rozanna Argente -- Fourth Choice: Gussy (



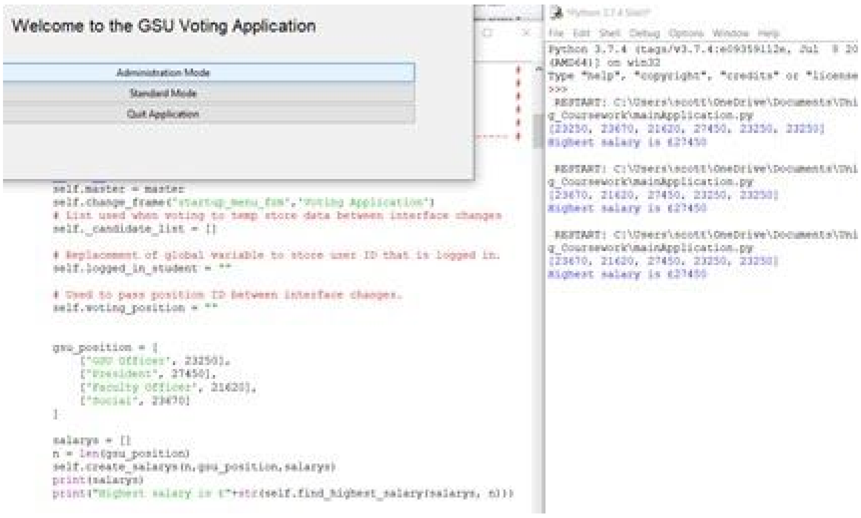
No votes submitted.

My feature fully integrated with the user interface, displaying a message if no votes were submitted. If the student had voting during their time logged in, on logout, the user would be redirected to a new page containing the receipt of their votes. It also printed to the CLI to make debugging easier.



```
$ python mainApplication.py
Voted Position: Faculty of Education and Health Officer -- First Choice: Lionel
Rosenstiel -- Second Choice: Candida Domingues -- Third Choice: Maximilian Dalry
mple -- Fourth Choice: Vassily Bick
Voted Position: Faculty of Education and Health Officer -- First Choice: Ida Mcw
hinney -- Second Choice: Sherill Rummery -- Third Choice: Rozanna Argente -- Fou
rth Choice: Gussy Grundon
```

Feature B.4 - Scott Williams - Task 7


Welcome to the GSU Voting Application
Administration Mode
Standard Mode
Quit Application
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help

```
self.master = master
self.change_frame('startup_menu_frm','Voting Application')
# List used when voting to temp store data between interface changes
self._candidate_list = []

# Replacement of global variable to store user ID that is logged in.
self.logged_in_student = ""

# Used to pass position ID between interface changes.
self.voting_position = ""


gsu_position = [
    ["GSU Officer", 23250],
    ["President", 27450],
    ["Faculty Officer", 21620],
    ["Social", 23670]
]

salarys = []
n = len(gsu_position)
self.create_salarys(n,gsu_position,salarys)
print(salarys)
print("Highest salary is £"+str(self.find_highest_salary(salarys, n)))
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 20
(AMD64)] on win32
Type "help", "copyright", "credits" or "license
>>>
 RESTART: C:\Users\scott\OneDrive\Documents\Uni
g_Coursework\mainApplication.py
[23250, 23670, 21620, 27450, 23250, 23250]
Highest salary is £27450

 RESTART: C:\Users\scott\OneDrive\Documents\Uni
g_Coursework\mainApplication.py
[23670, 21620, 27450, 23250, 23250]
Highest salary is £27450

 RESTART: C:\Users\scott\OneDrive\Documents\Uni
g_Coursework\mainApplication.py
[23670, 21620, 27450, 23250, 23250]
Highest salary is £27450
```

Scott’s feature was implemented to the application, displaying the correct results in the CLI when the application started. Integration with the GUI was not requested and was not performed.

References

1. Pythonprogramming.net. (2019). *Python Programming Tutorials*. [online] Available at: https://pythonprogramming.net/change-show-new-frame-tkinter/ [Accessed 26 Nov. 2019].

2. GitHub. (2019). *How to get string from an entry box? · Issue #138 · alejandroautalan/pygubu*. [online] Available at: https://github.com/alejandroautalan/pygubu/issues/138 [Accessed 26 Nov. 2019].

3. window?, H., Gregory, M., Gregory, M. and Guiles, M. (2008). *How do I close a tkinter window?*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/110923/how-do-i-close-a-tkinter-window [Accessed 26 Nov. 2019].

4. GitHub. (2019). *alejandroautalan/pygubu*. [online] Available at: https://github.com/alejandroautalan/pygubu/wiki/6-Capture-DeleteWindowEvent [Accessed 26 Nov. 2019].

5. GitHub. (2019). *How to open a new window (frame) ? · Issue #132 · alejandroautalan/pygubu*. [online] Available at: https://github.com/alejandroautalan/pygubu/issues/132 [Accessed 26 Nov. 2019].

6. GitHub. (2019). *alejandroautalan/pygubu*. [online] Available at: https://github.com/alejandroautalan/pygubu/wiki/CommandProperties [Accessed 26 Nov. 2019].

7. Dev.mysql.com. (2019). MySQL :: MySQL Connector/Python Developer Guide :: 5.1 Connecting to MySQL Using Connector/Python. [online] Available at: https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html [Accessed 26 Nov. 2019].

8. Raspberrypi.org. (2019). Can I hide a tkinter Frame - Raspberry Pi Forums. [online] Available at: https://www.raspberrypi.org/forums/viewtopic.php?t=101199 [Accessed 26 Nov. 2019].

9. DaniWeb. (2011). clear and replace all widgets on tkinter frame. [online] Available at: https://www.daniweb.com/programming/software-development/threads/386718/clear-and-replace-all-widgets-on-tkinter-frame [Accessed 26 Nov. 2019].

10. MySQL, I., Christensen, R. and Peisert, C. (2009). Inserting a Python datetime.datetime object into MySQL. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/1136437/inserting-a-python-datetime-datetime-object-into-mysql [Accessed 26 Nov. 2019].

11. class, P., Orozco, T. and Soltanto, M. (2012). Python calling method in class. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/14086830/python-calling-method-in-class [Accessed 26 Nov. 2019].

12. GitHub. (2019). alejandroautalan/pygubu. [online] Available at: https://github.com/alejandroautalan/pygubu [Accessed 26 Nov. 2019].

13. PyPI. (2019). pygubu. [online] Available at: https://pypi.org/project/pygubu/ [Accessed 26 Nov. 2019].

14. Python, H. (2018). How to change label text in Pygubu / Python. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/50777243/how-to-change-label-text-in-pygubu-python [Accessed 26 Nov. 2019].

15. Molina, A. (2018). Hashing Passwords in Python – Useful code. [online] Vitoshacademy.com. Available at: https://www.vitoshacademy.com/hashing-passwords-in-python/ [Accessed 27 Nov. 2019].

16. Vollebregt, B. (2019). How To Hash Passwords In Python. [online] Nitratine.net. Available at: https://nitratine.net/blog/post/how-to-hash-passwords-in-python/ [Accessed 27 Nov. 2019].

17. TypeError: a bytes-like object is required, n., 412389421, 2. and Pedramfar, M. (2019). TypeError: a bytes-like object is required, not 'str' - hmac. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/54051747/typeerror-a-bytes-like-object-is-required-not-str-hmac [Accessed 27 Nov. 2019].

18. DaniWeb. (2006). Draw a Bar Graph (Python). [online] Available at: https://www.daniweb.com/programming/software-development/code/216816/draw-a-bar-graph-python [Accessed 29 Nov. 2019].

19. values, U. and Kraaijeveld, J. (2012). *Using Python's max to return two equally large values*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/9853302/using-pythons-max-to-return-two-equally-large-values [Accessed 9 Dec. 2019].

20. Gaudard, O. (2017) #11 Grouped barplot, [online] The Python Graph Gallery. Available at: https://python-graph-gallery.com/11-grouped-barplot/ (Accessed: 10 December 2019).

21. Gaudard, O. (2017) *#5 Control width and space in barplots*, [online] *The Python Graph Gallery*. Available at: https://python-graph-gallery.com/5-control-width-and-space-in-barplots/ (Accessed: 10 December 2019).

22. datetime, C., Harrington, P., Willison, S. and Peak, S. (2009). Converting string into datetime. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/466345/converting-string-into-datetime [Accessed 17 Jan. 2020].

23. icon, S., Daulton, S., Farkash, Y. and B., M. (2013). Set window icon. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/18537918/set-window-icon [Accessed 18 Jan. 2020].