

JAVA 2A GROUP 1

VOTING APPLICATION

DOCUMENTATION

JAVA 2A GROUP 1
VOTE APPLICATION DOCUMENTATION

1. BACKGROUND:

This is a voting application built by Java 2A Group 1. This application digitalizes the entire processes of voting which involves the use of ballot papers, ballot box, voting booth, ink, etc. in Accra Technical University.

2. PROBLEM STATEMENT:

With the existing system of voting (which includes the use of ballot boxes, etc.) there were problems such as:

1. Theft of ballot papers.
2. Voters tend to select multiple or no candidates, tear ballot papers, or print their thumbs anywhere aside the thumb-print area, etc. These actions render these ballot papers to be rejected.
3. A lot of hard copies in the system.
4. Counting of votes are assigned to humans which
 - a. makes the results of the votes unreliable yet unacceptable.
 - b. makes the results unreliable since humans are subjected to mistakes
 - c. makes the results unreliable since humans can add up already-voted ballots before counting
 - d. causes commotions due to mistrusts amongst humans

3. OBJECTIVE:

The main objective of this system is to create a reliable and a secure system for voting in Accra Technical University by:

1. Preventing the theft of ballot papers
2. Preventing voters from misusing the ballot papers
3. Control the use of hard copies
4. Prevent commotions

5. Make voting a fair deal
6. Making vote counts reliable since this system will handle the accumulation of the votes

4. **DESIGN AND IMPLEMENTATION:**

This application is designed on the basis of Object-Oriented Programming.

Candidate and Portfolio classes were created and instances or objects of these classes were constructed, making it easy to access properties or attributes like portfolio name, candidate votes, etc.

1. FUNCTIONAL REQUIREMENTS:

Some of the methods or functions used are as follows:

- a. canVote function: This is a method uses the voter's ID and the array of the three portfolios as parameters. It returns a Boolean value depending on whether the voter's ID parameter is present in all the three arrays passed to it.
- b. addPortfolio function: This method sets the name of the portfolios to the text of the portfolio checkboxes. It picks a JCheckBox and a Portfolio argument. This method was later overloaded to accept different parameters which includes a JCheckBox, a JLabel and a Portfolio object as arguments. It is used to set up the new portfolios
- c. showLabels function: This method picks an integer parameter and one or more labels and toggles their visibility based on the condition of the value of the integer being equal to zero or not
- d. swapPanels function: This method picks a JPanel object and an array of JPanel objects (variable arguments) and sets the visibility of the first argument to true and the rest, false. It helps to show the needed panel at

the moment when it is needed and hide all other panels that that's not needed at the moment.

- e. enableButtons function: This method picks JButton objects and an integer argument and toggles the enablement of the buttons based on the condition of the integer argument not equal to zero
- f. deselectCheckBoxes function: JCheckBoxes are passed to this function as parameters and the function deselects them.
- g. enableLabels function: This method sets the visibility of labels passed to the method (as parameters) to the value of the integer parameter not equal to zero
- h. clearLabels function: This method sets the icon of labels passed to it as parameters to null. Also, sets the text of these labels to an empty string
- i. showCheckBoxes function: This method operates with the same algorithm as enableLabels. But in this case, checkboxes are passed to the method as arguments. It set the visibility of these checkboxes to the value of the integer parameter $\neq 0$
- j. setCandidatesForVote function: Two or more Candidate objects are passed as arguments to this function. It works by getting the values of the candidates (image and name) and set's the labels in the candidatePane to the images and names to enable the user select them
- k. confirmVote function: This function pops up a confirmation frame to allow the user to confirm his or her vote

- l. saveVote function: This function picks the various portfolio checkboxes and decides which of the portfolios' array to save the voter's id. This is very important to the program as it helps to detect if the user already voted for a particular portfolio or is yet to vote for it.
- m. alertVote function: This function pops a frame and tells the user he has successfully voted for a particular candidate for a particular portfolio
- n. setCandidateResults function: This function sets the images of all candidates, their names and their respective vote counts.
- o. MenuBar function: This function returns a JMenuBar value with all the menu bar entities

5. GRAPHICAL USER INTERFACE:

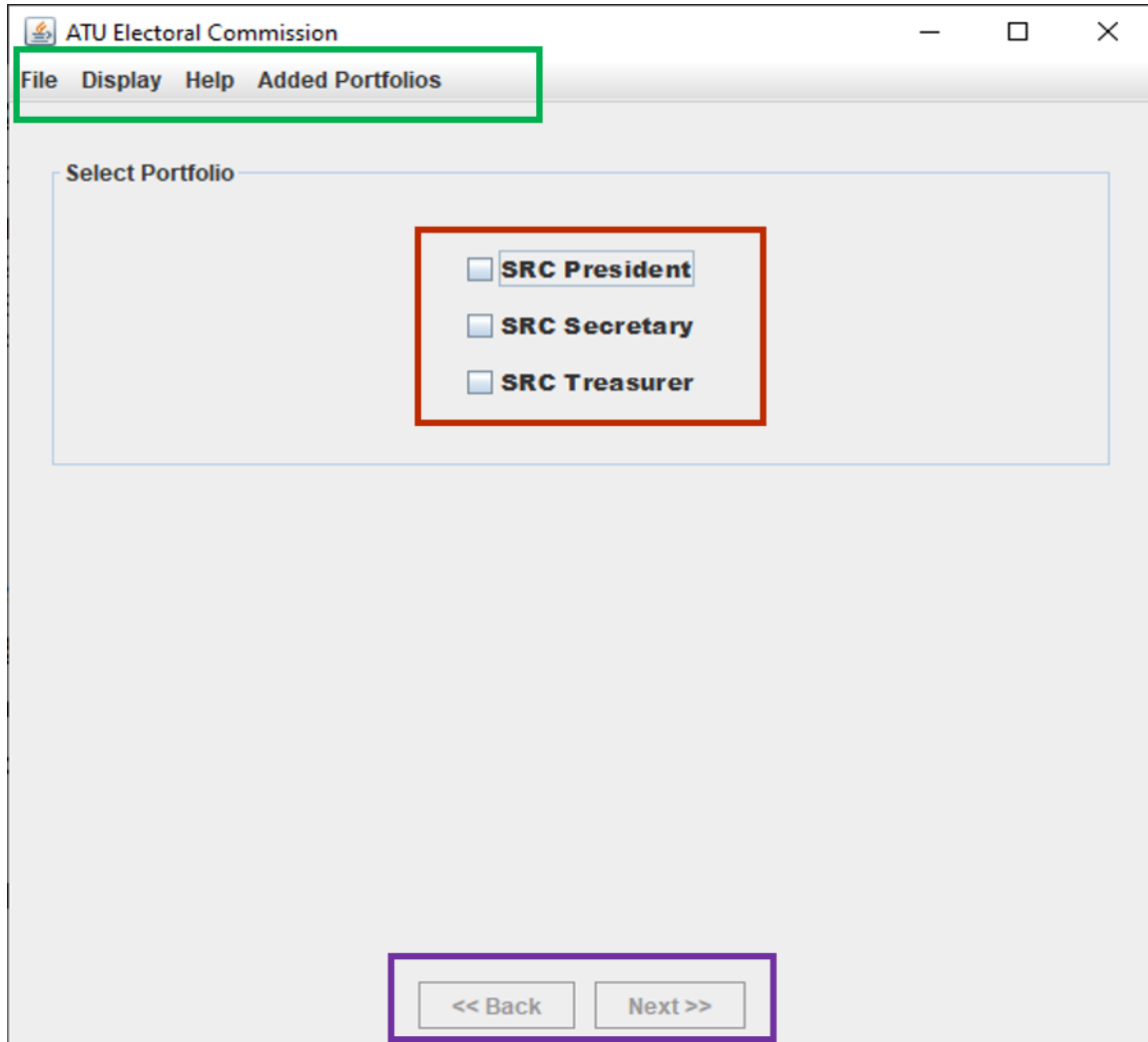


Figure 1

1. Section marked in green is the menu bar
2. Section marked in red is various portfolios' checkboxes
3. Section marked in purple is the navigation buttons

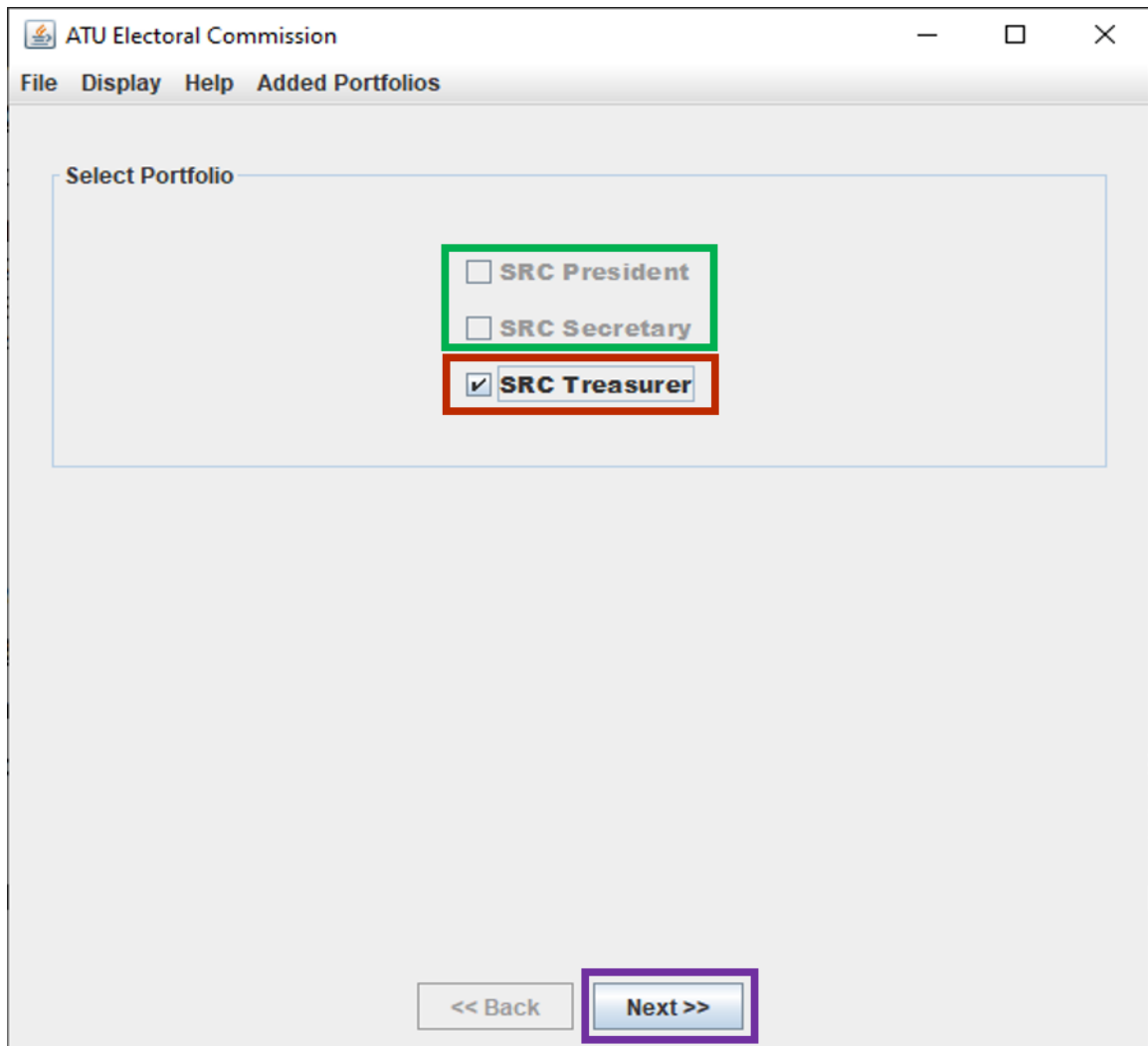


Figure 2

1. Section marked in green is the disabled portfolios' checkboxes
2. Section marked in red is various portfolios' checkboxes
3. Section marked in purple is the next navigation button enabled

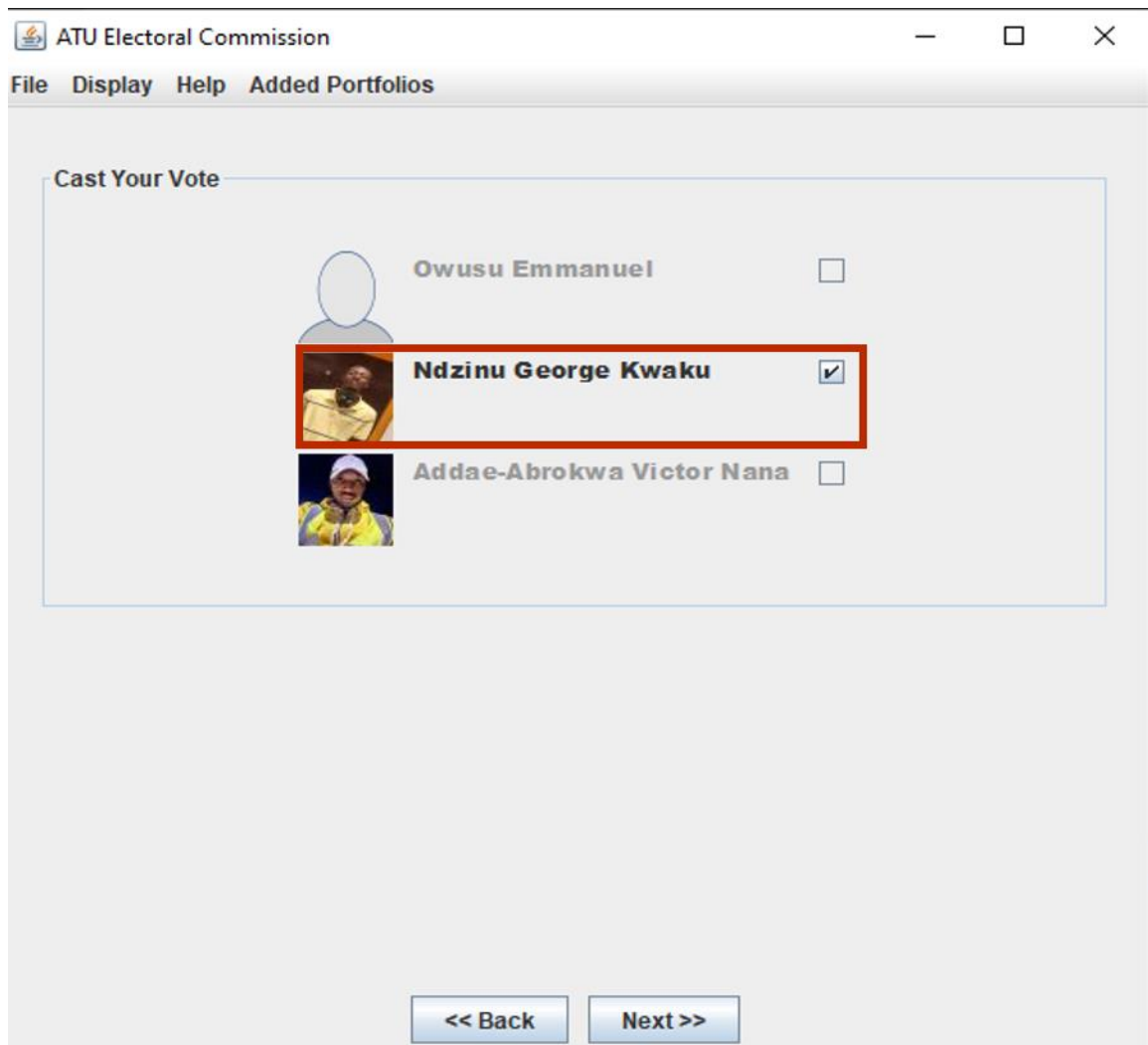


Figure 3

1. Section marked in red is a selected candidate

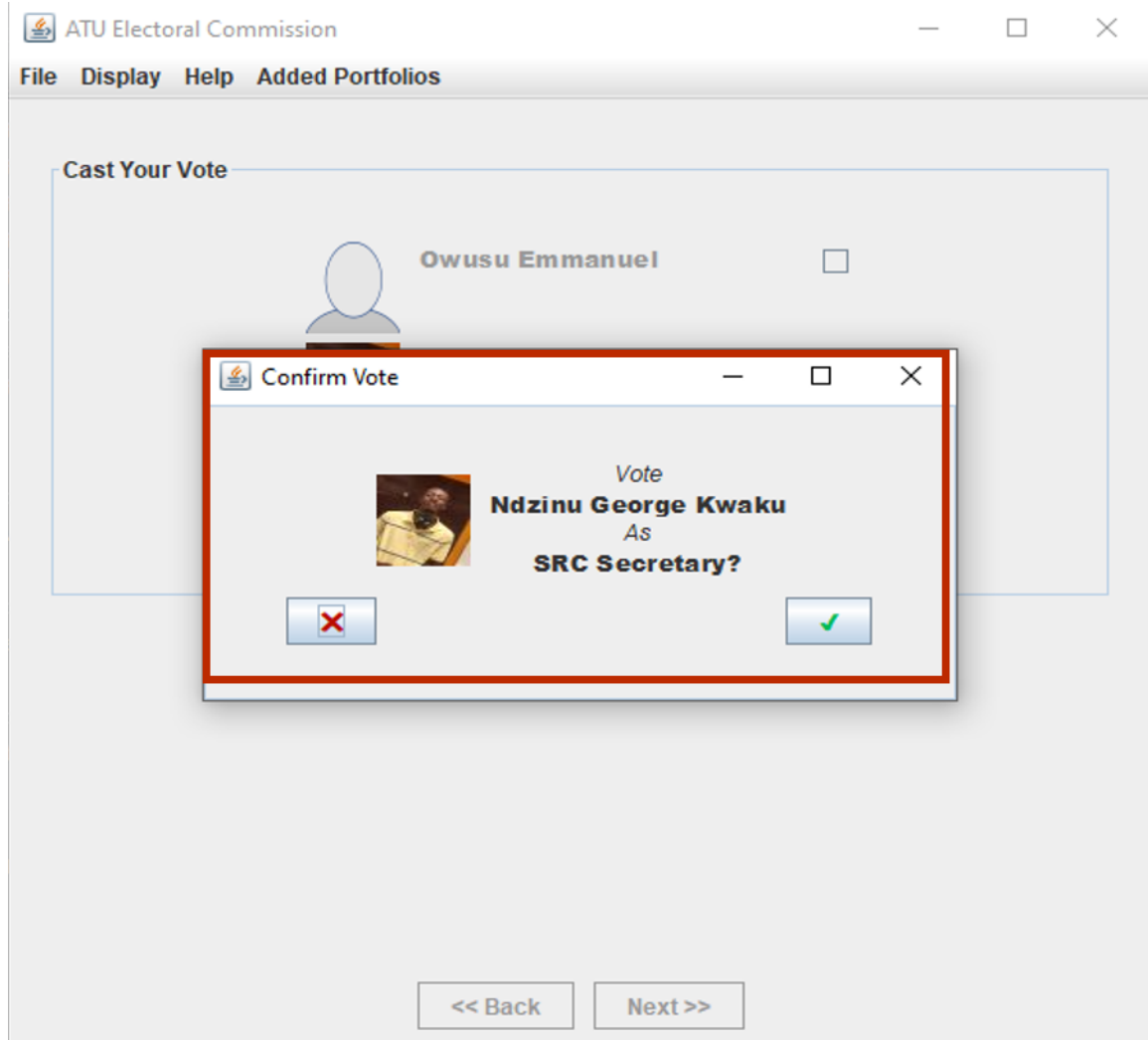


Figure 4

Figure 4 shows a pop up which appeared after a candidate has been selected for vote.

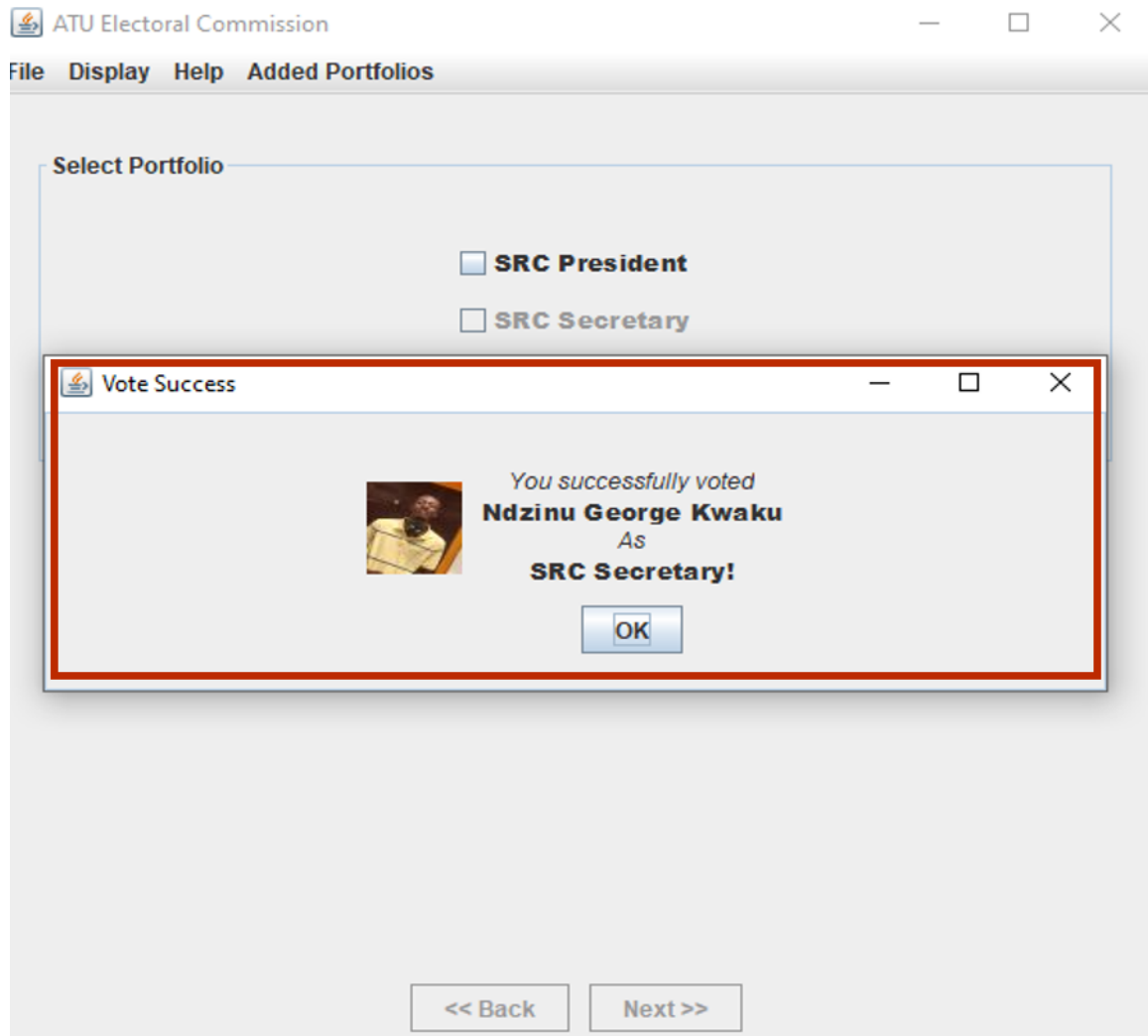


Figure 5

Figure 5 shows a pop up after the user confirmed a candidate

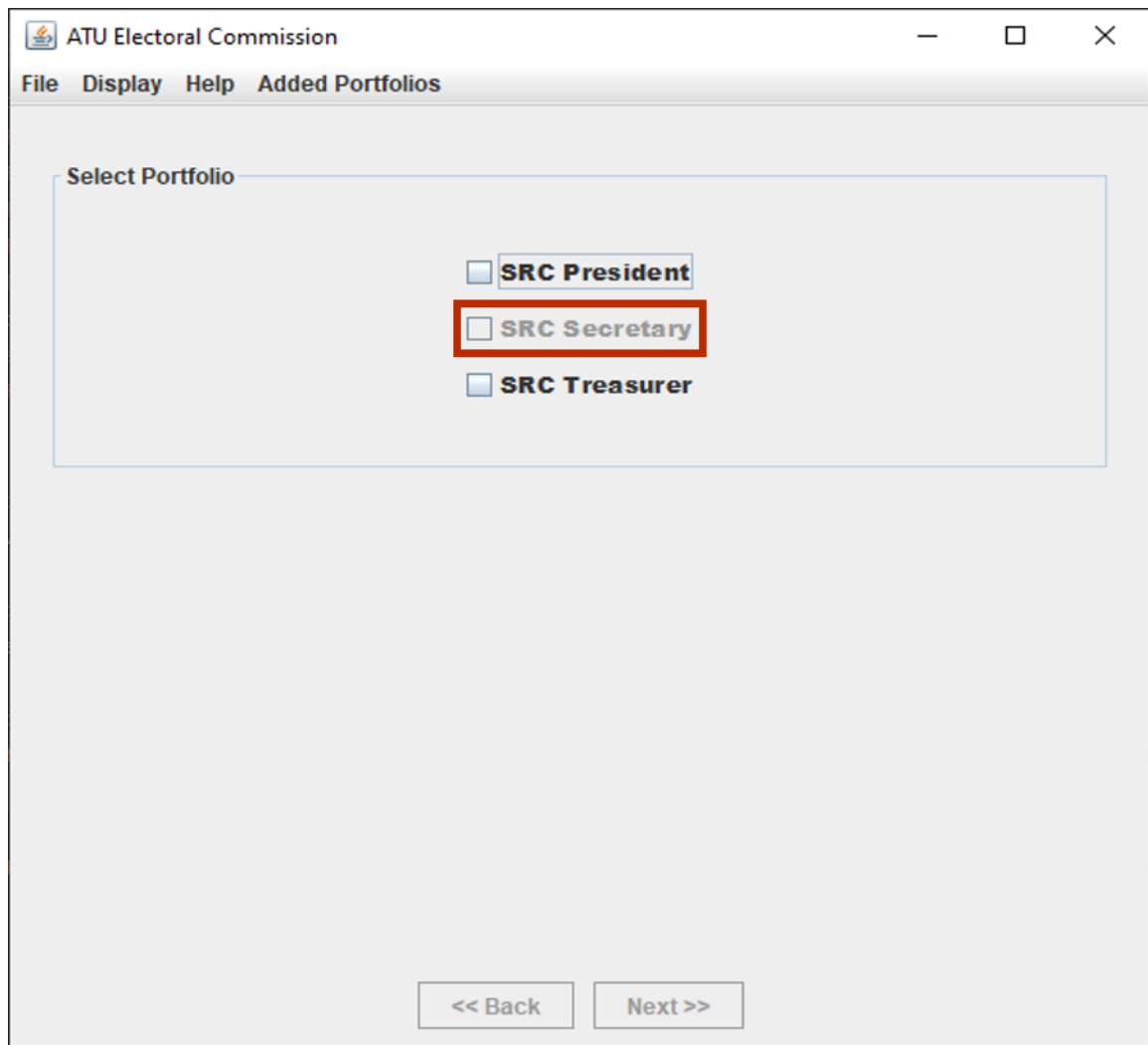


Figure 6

Figure 6 shows the state of the SRC Secretary portfolio checkbox after the user voted for a candidate in that portfolio

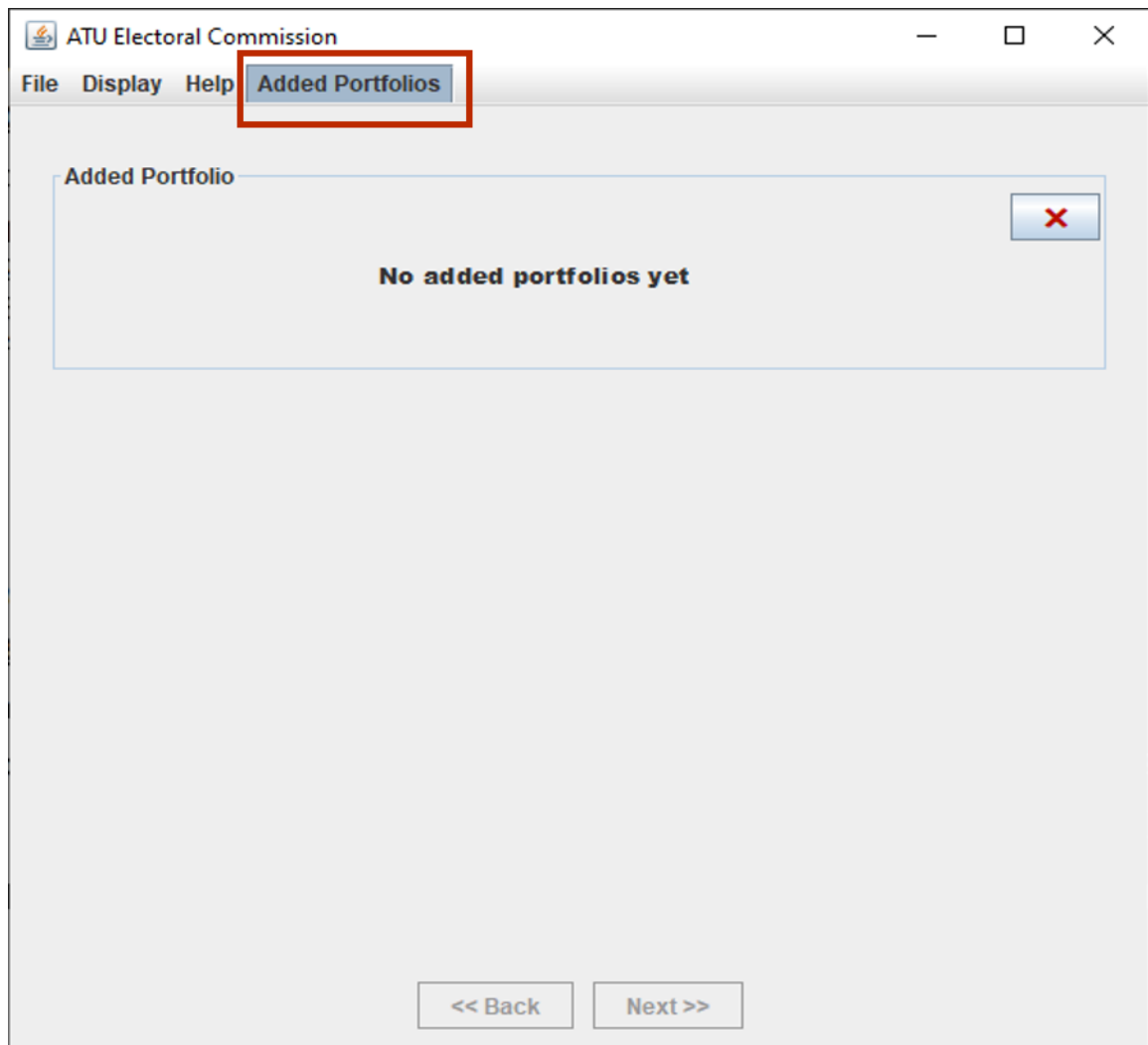


Figure 7

Figure 7 shows the "Added Portfolio" menu clicked showing the added portfolio buttons

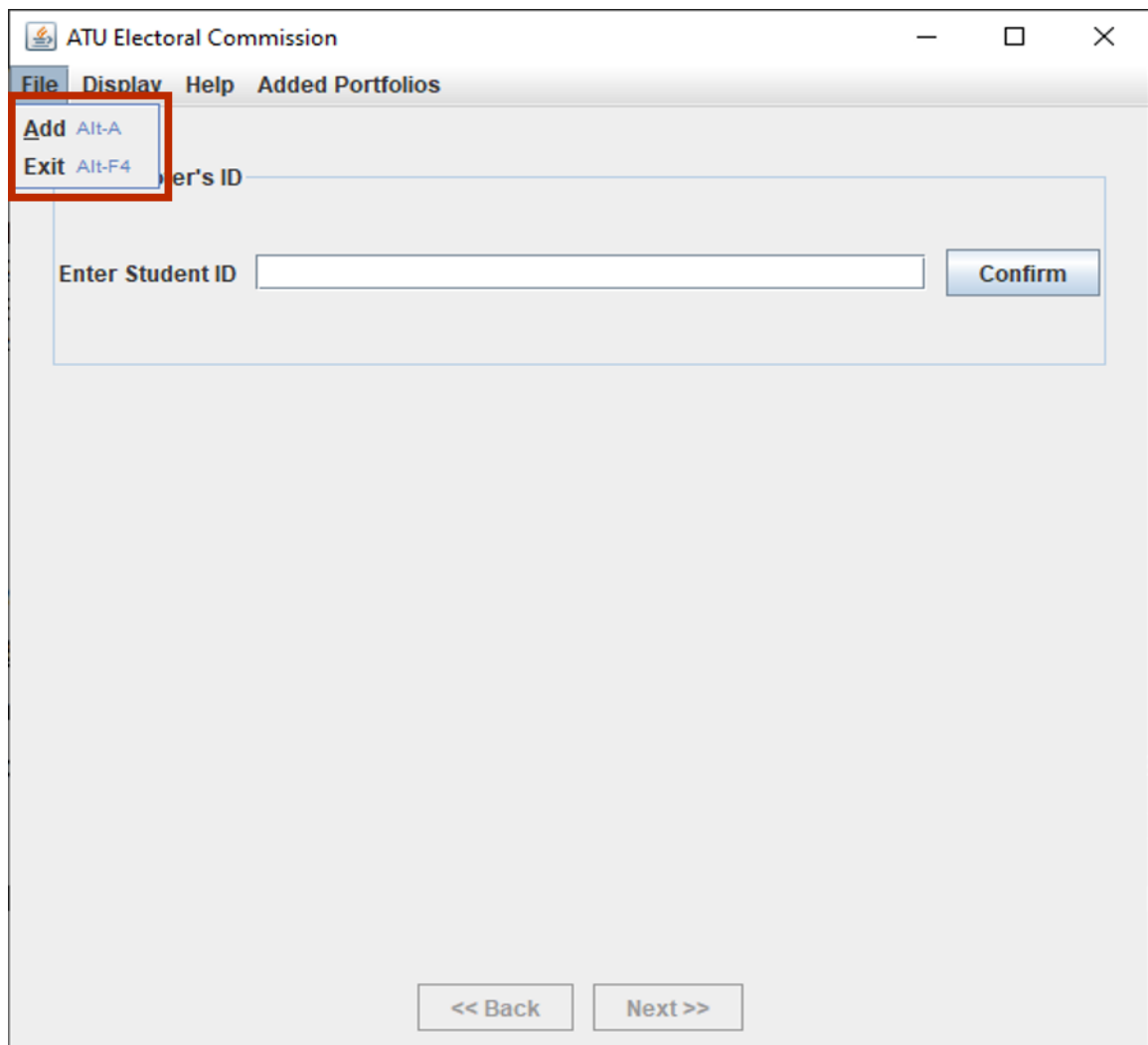


Figure 8

The section in red refers to the menu items of the "File" menu

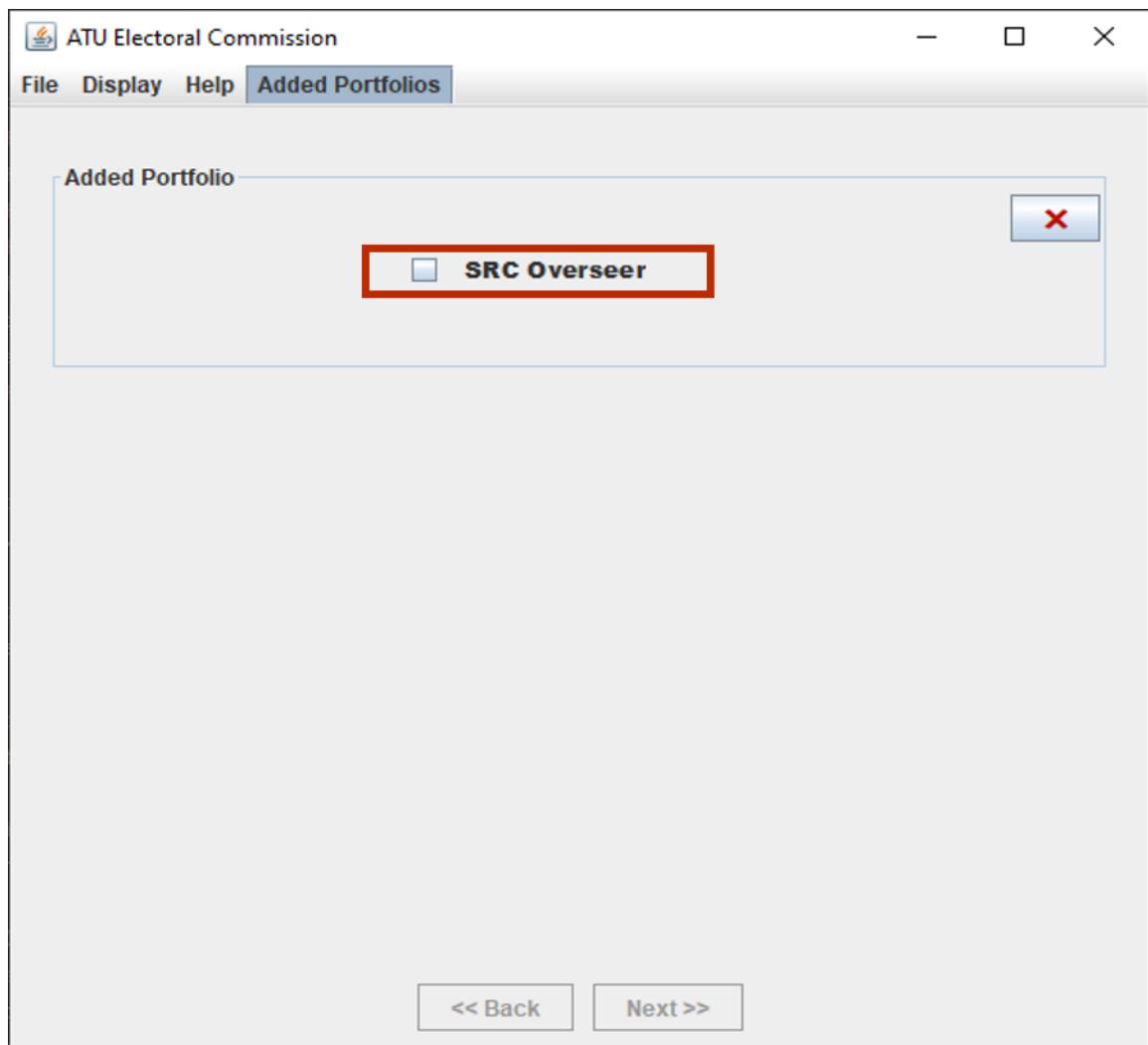


Figure 9

The section in red shows a newly added portfolio

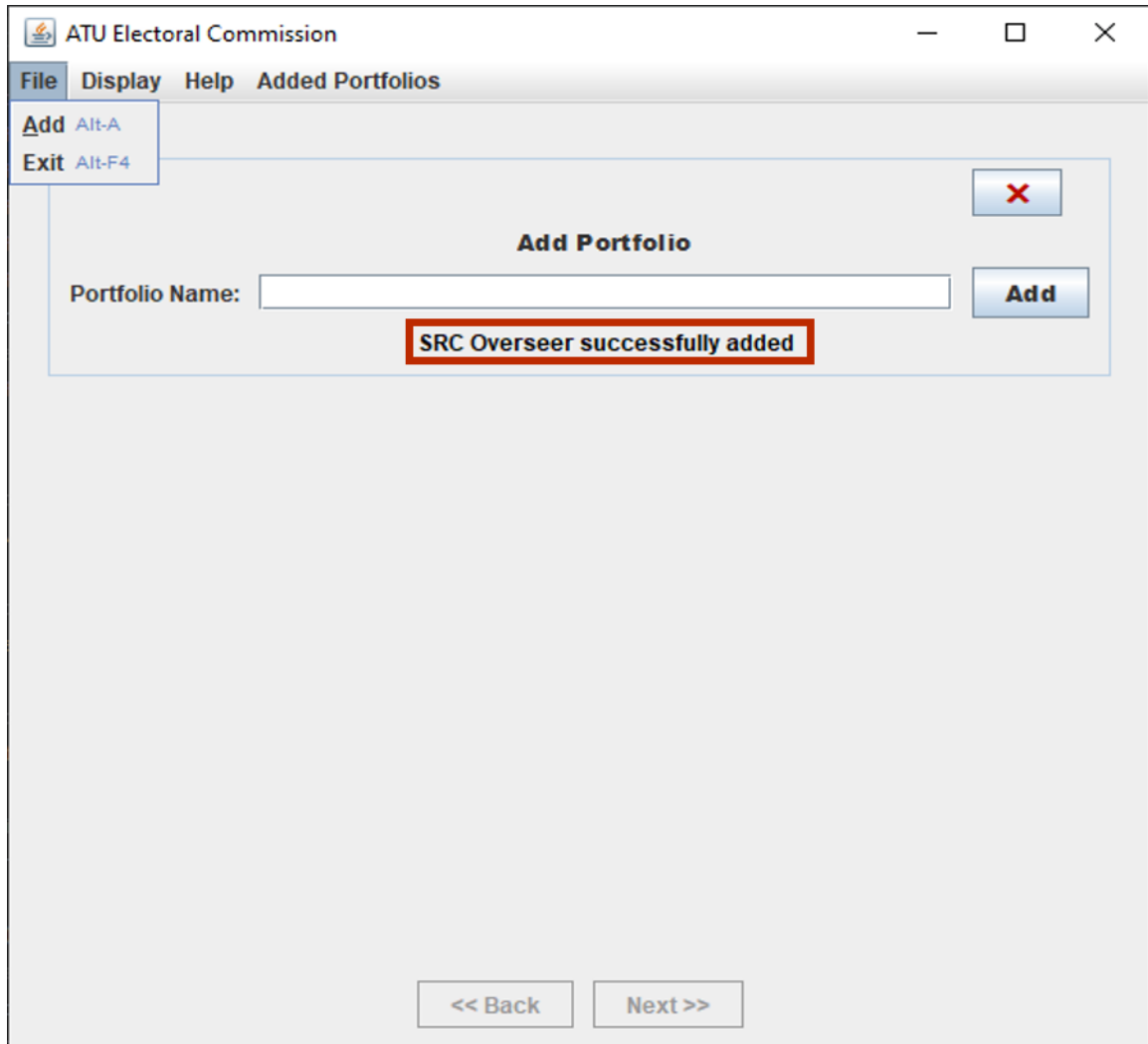


Figure 10

The section in red shows the message label of the portfolio

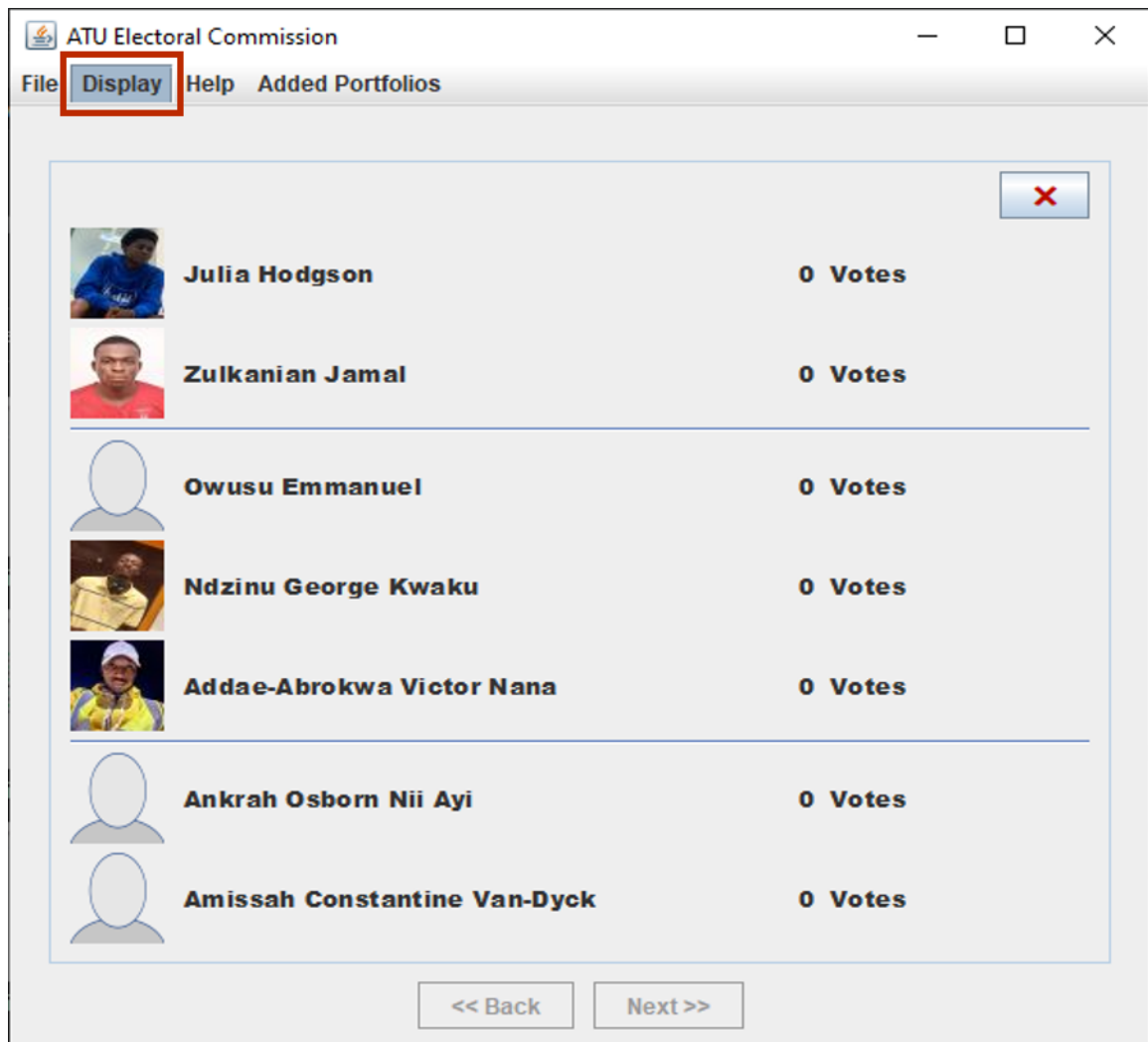
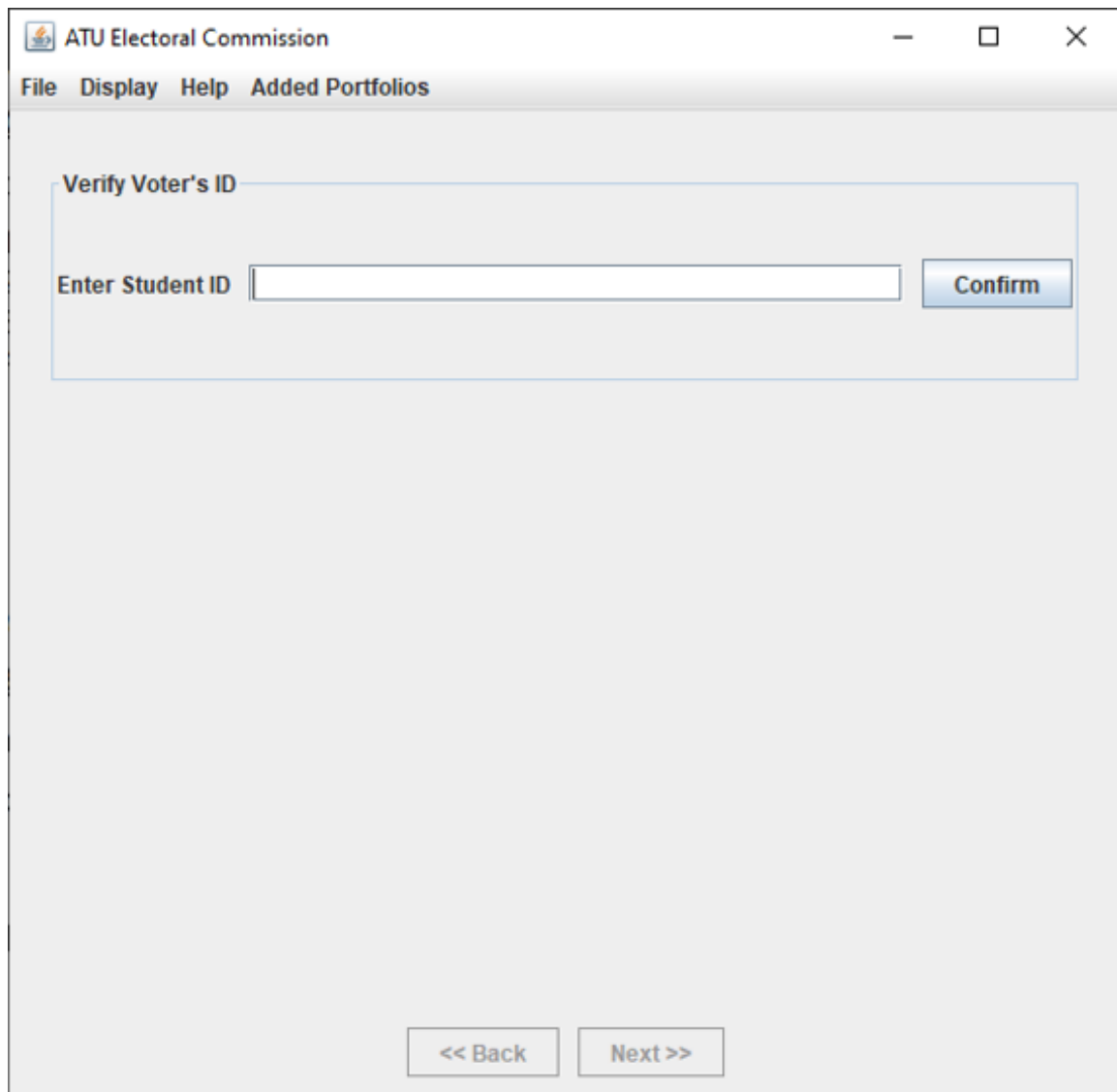


Figure 11

The section in red shows the "Display" menu. The rest of the panel shows the images, names and number of votes of each candidate.



ATU Electoral Commission

File Display Help Added Portfolios

Verify Voter's ID

Enter Student ID

The image shows a software window titled "ATU Electoral Commission" with standard window controls (minimize, maximize, close) in the top right. Below the title bar is a menu bar with "File", "Display", "Help", and "Added Portfolios". The main content area is titled "Verify Voter's ID" and contains a form. The form has a label "Enter Student ID" followed by a text input field. To the right of the input field is a "Confirm" button. At the bottom of the window, there are two buttons: "<< Back" and "Next >>".

Figure 12

Figure 12 shows the voter's id validation panel

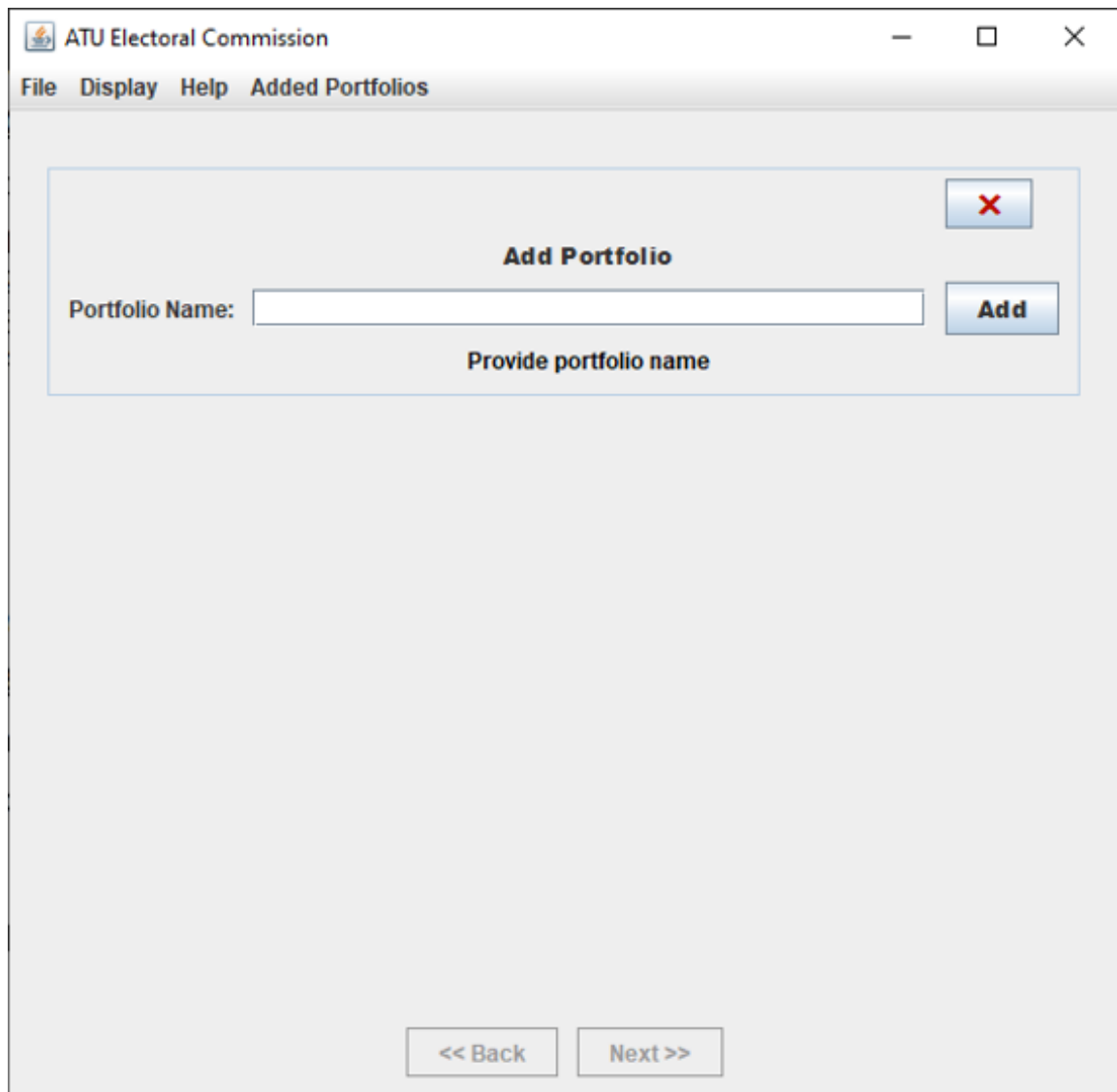


Figure 13

Figure 13 shows the panel that enables the user to add new portfolios

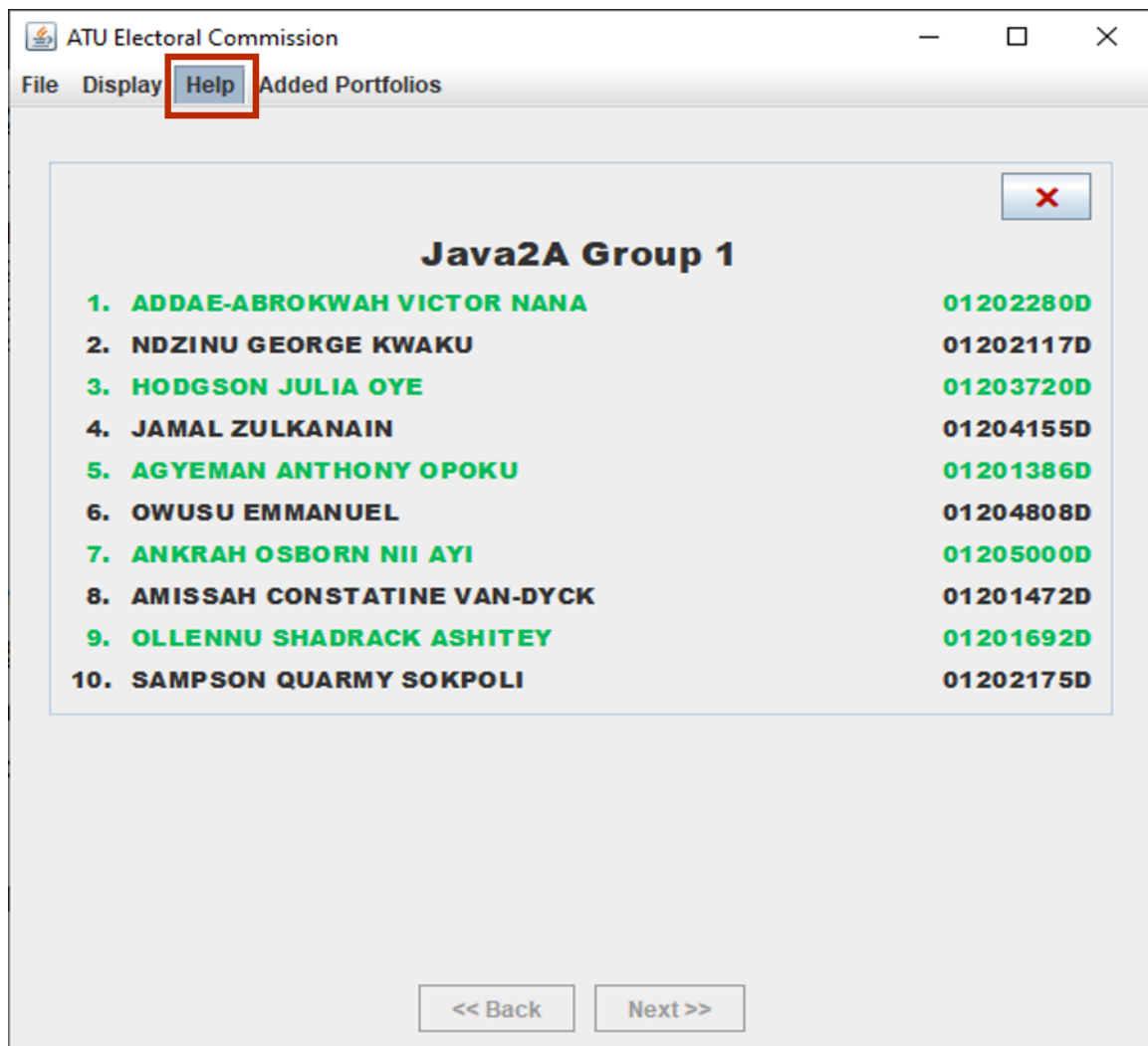


Figure 14

Figure 14 show the "Help" menu highlighted in red and the list of the group members

➤ **SWING COMPONENTS USED:**

1. ImageIcon
2. JPanel
3. JTextField
4. JButton
5. JLabel
6. JCheckBox
7. JFrame
8. JMenuBar
9. JMenu
10. JMenuItem

➤ **EVENTS HANDLED:**

1. WindowEvent
2. KeyEvent
3. InputEvent
4. MouseEvent
5. ActionEvent

➤ **EXCEPTIONS HANDLED:**

1. User providing not providing portfolio or voter's ID
2. User providing a voter's ID with less or more than 9 characters