

**Voting Management System**

ET0706 - Object Oriented Programming

Final Report

DCPE/3A/01

Brians Tjipto Meidianto (xxxxxxx)

**Table of Contents**

[**Introduction**](#_jd8tqdrwgw3m) **3**

[Problem](#_fgz7c943wk5i) 3

[Solution](#_4vxu7s902h7) 3

[Objective](#_uicfishsvycr) 3

[**Object and Class**](#_rz801csf51vj) **3**

[User.java](#_1oq6clkwl2) 3

[voteStatus.java](#_afpfvc28hopm) 3

[CandidateSelected.java](#_ji83d5v9n2l9) 4

[ModelTable.java](#_afzim0yvjxat) 5

[UML](#_dcpgnjrrc1up) 5

[**JavaFX**](#_31qluleb1n3a) **6**

[FXML](#_6lraaurcu37t) 6

[Login.fxml](#_vwufas5lva5p) 6

[Sign Up.fxml](#_a7b1ntcoz4lo) 6

[Home.fxml](#_vt7x2a6tz41d) 6

[Voting.fxml](#_r3o838d93t3j) 6

[FinalVotes.fxml](#_iqz2oahcshi3) 6

[Admin.fxml](#_rnf0mfhgiwgr) 6

[Controller](#_oi9bx3h0x7z2) 6

[LoginController.java](#_7vhp5edlj3pa) 6

[SignupController.java](#_f79d4o4wgq7c) 6

[HomeController.java](#_wgfln5nrzjlq) 7

[VotingController.java](#_7874ijv0e8z7) 7

[FinalVotesController.java](#_j9anr8iqhs4r) 7

[AdminController.java](#_a2sfy9h28m9e) 8

[**Database design**](#_z81pk03hbk7d) **8**

[votes database](#_krs8aurdzgjj) 8

[candidates database](#_l86s7vbsoo37) 8

[Connection](#_4emp3g1opcld) 8

[**User Guide**](#_ubyvwv6qr8v3) **9**

[File Structure](#_vcme8u2e5ydt) 9

[Usage](#_qoxs2c28lvgi) 10

[**Additional Library**](#_d729d9ku0iw6) **14**

[Structured Query Language (SQL)](#_6nk8qoyygcv7) 14

[JDBC (Java Database Connectivity)](#_21iqslmm15dx) 14

[**Resources**](#_b5pe399p16mv) **14**

# Introduction

#### Problem

The classical electoral system is ancient and outdated, it requires a lot of staff members to hold the voting, as well as count the ballots one by one. It also restricts people with special needs like those with disability from voting, as it requires them to go to the voting site itself, as well as the chance of the miss counting the ballot due to human error can be an immense downside. Not to mention the extensive and labor-intensive time it takes to count each and every ballot one by one, tally it up, and finally post the results.

#### Solution

The solve this problem I have made a voting system that everyone can access and allows them to vote from the comfort of their homes. This can, not only increase the number of voters significantly, but also decrease the counting time, and having the final tally of the votes out in no time.

#### Objective

Make voting intuitive and intriguing for everyone to be able to vote in the comfort of their homes.

# Object and Class

#### User.java

This class has 2 overloading constructor that takes in either 2 string data, name, password, or 4 string data, name, password, voteFor, and DateofBirth, this is used to pass the login information through scenes in JavaFX.

It contains a getter and setter class to get and set the name, password, voteFor, and DateofBirth when passing the information through scenes.

#### voteStatus.java

This class takes has 2 overloading constructors that will either only take in a single voteForStr string (format of <can>\_<id> e.g., can\_1), or a voteForStr string as well as a DateofBirth string.

Inside the class have various methods:

* candidate() which will call the CandidateSelect class from CandidateSelect.java invoking the candidateDB() and calling the setCandidate() taking in a string voteForStr which will return the full name of the candidate and put it into a string voteForS.
* VoteStatus() which will read the voteFor string that was taken in, and return either “Voting status: Voted” if voteFor is not equal to 0 (i.e., contains a value other than 0), or will return “Voting status: Not Voted” if voteFor is equal to 0.
* VoteFor() which will read the voteFor string that was taken in, and return either “Voted for: ” + voteForS (full name of the candidate) if voteFor is not equal to 0, or will return “Voted For: Not Voted” if voteFor is equal to 0.
* FormatDate(String) will return the formatted date of DateofBirth by using the method FormatDate, it will format it from the yyyy-MM-dd format to the EEEE, MMM d, yyyy format (e.g., from 2004-05-29 to Saturday, May 29, 2004).
* calculateAge(LocalDate) will get the LocalDate (date today), and calculate the period year between the DateofBirth to the LocalDate returning the age of the user.
* FindAge() will return the age of the user, from the DateofBirth taken in by calling the method calculateAge().
* minAge() will access the database candidate table and get the minAge column and put it into a string MinAge.
* getMinAge() will return the minimum age, MinAge by calling the minAge() method.
* Eligibility() will return the eligibility of a user, seeing the age, MinAge by calling getMinAge(). If the age is equal to or above the MinAge then it will return "Voting Eligibility: Eligible" if it is lower than the MinAge it will return “Voting Eligibility: Ineligible” instead.

#### CandidateSelected.java

This class has 2 HashMap, first candidateMap, and second candidateMapOpp. It contains a few methods:

* candidateDB() This method access the idCandidate column and candidatename column from the candidate database table. candidateMap has the key set as the idCandidate, and the value set as candidatename, while candidateMapOpp is the opposite where, the key is the candidatename, while the value is the idCandidate.
* setCandidate(string) will split the candidate from the <can>\_<id> format to only get the <id>, and then call the hashmap candidateMap using the id as the key returning the candidate’s full name.
* getCandidate(string) will call the hashmap candidateMapOpp and get the id from the key string taken in canName, and return the candidate in the format of <can>\_<id>

#### ModelTable.java

This class constructor takes in 2 string data, id and name, this is used for the TableView in JavaFX.

It contains a getter and setter class to get and set the id and name when forming the table.

#### UML

# 

# JavaFX

#### FXML

##### Login.fxml

main login screen

##### Sign Up.fxml

sign up screen

##### Home.fxml

home screen after the user login

##### Voting.fxml

voting screen

##### FinalVotes.fxml

final votes screen that displays the winner and the statistics

##### Admin.fxml

admin screen

#### Controller

##### LoginController.java

Get the user login data of the username and password and cross-check with the database votes, if it hits the match on both the name and passwords, the user will be sent to Sign Up.fxml, else if it matches the admin credentials “Admin” as the name, and “1234” as the password, it will send the user to the Admin.fxml, but if the logins credentials are wrong it will show an incorrect login label. It also has a button to access the Sign Up.fxml for the user to sign up instead. This will call out the class User to create a new object for the name and password entered and pass the information to the HomeController.java

##### SignupController.java

Get the user sign-up data of username, password, genderSelected, voteFor (set to 0 automatically to state that the user has not voted when they sign up), and a formatted date of birth in the format of yyyy-MM-dd, and insert it into the database table votes. After insertion of the data, it will return the user to the login page to login into the account. It also has a button to access the Login.fxml for the user to log in instead.

##### HomeController.java

This page shows all the user's information, name, date of birth, age, voting eligibility, and voting status, as well as voted for. The name passed along from the object created in LoginController through the class User, and the class voteStatus is being called to get the other information from the methods inside voteStatus, calling the method FindAge(), VoteStatus(), VoteFor(), Eligibility(), and FomatDate(). This page allows the user to go to the Voting.fxml if the user is eligible to vote and pass the object class User, go to the FinalVotes.fxml to see the results passing the User class, or log out.

##### VotingController.java

This page allows the user to vote for the candidate, the candidate is shown in a table calling the ModelTable.java class and the candidates database table. The candidate that is from the table is in the form of the candidate’s full name so the class CandidateSelected is called to use the method getCandidate(string), getting the formatted candidate to be placed in the votes database table. voteStatus is also being called, to see if the user has just voted it will change the database votes so that the login-in user has voted and set the vote as the formatted voted for. After the user has cast or changed their votes the page refreshes to get the new information. This page allows the user to go to the Home.fxml to see the user's information, go to the FinalVotes.fxml to see the results passing the User class, or log out.

##### FinalVotesController.java

This page allows the user to see the final votes winner, pie chart, as well as statistics (percentage). The User class will return the name of the user and call out class votingStatus to get the voting status and voted for. Firstly, it will call the candidates database table, then It will then get the ModelTable class to get set the table layout and display the table counting the percentage of each of the candidates using the method findPercentage() this percentage will be shown in relation to the candidate, and finally, the data from the database will be placed into the pie chart and then the pie chart is displayed. The winner will be decided using the percentage, if there is any percentage that is equal to another, the winner label will display “Votes are tied”, but if there is a definitively largest percentage then it will be the winner, and the winner label will display "Winner is - " + candidatename. This page allows the user to go to the Home.fxml to see the user's information, or go to the Voting.fxml if the user is eligible to vote and pass the object class User, or log out.

##### AdminController.java

This page allows the user to change and edit the candidates using an editable table taking and updating the information from the candidates database table, calling the ModelTable Class to create the table view, It also allows the user to add more candidates to the candidate database table. The user can also change the minimum age required to be eligible to vote calling the voteStatus class to get the minimum age. This page allows the user to go to the FinalVotes.fxml to see the results, when calling the FinalVotesController a method adminFinal() is being called to hide the labels that are user-specific like the name, vote status, as well as who they voted for, or log out.

# Database design

#### votes database

|  |  |
| --- | --- |
| **idvotes** | int AI PK |
| password | varchar(45) |
| names | varchar(45) |
| gender | varchar(45) |
| voteFor | varchar(45) |
| DOBYMD | varchar(45) |

#### candidates database

|  |  |
| --- | --- |
| **idCandidate** | int AI PK |
| candidatename | varchar(45) |
| minAge | varchar(45) |

#### Connection

The Database is connected via a DBHandler file in the package DBConnection. It has a method getConnection() of type Connection to connect to the database using JDBC. The JDBC that was used in this project was from MySQL (JDBC Driver for MySQL (Connector/J)).

# User Guide

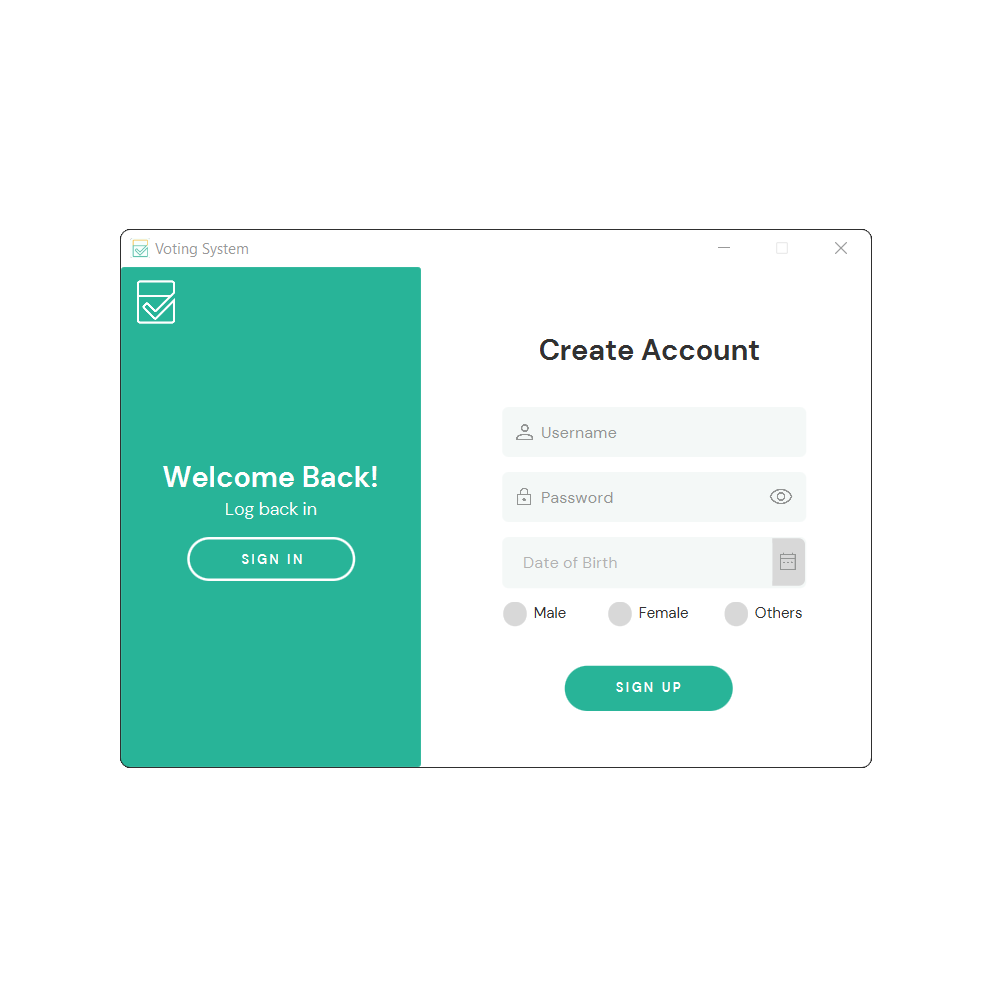
#### File Structure

Here is the file structure of the project, src contains all the files for the project itself, JRE System Library, JavaFX, JavaFX SDK, and Referenced Libraries contain the Java and JavaFX Java Development Kit (JDK), and Software development kit (SDK). And last, the lib folder contains the connector (mysql-connector-java.jar) to connect to MySQL database tables.

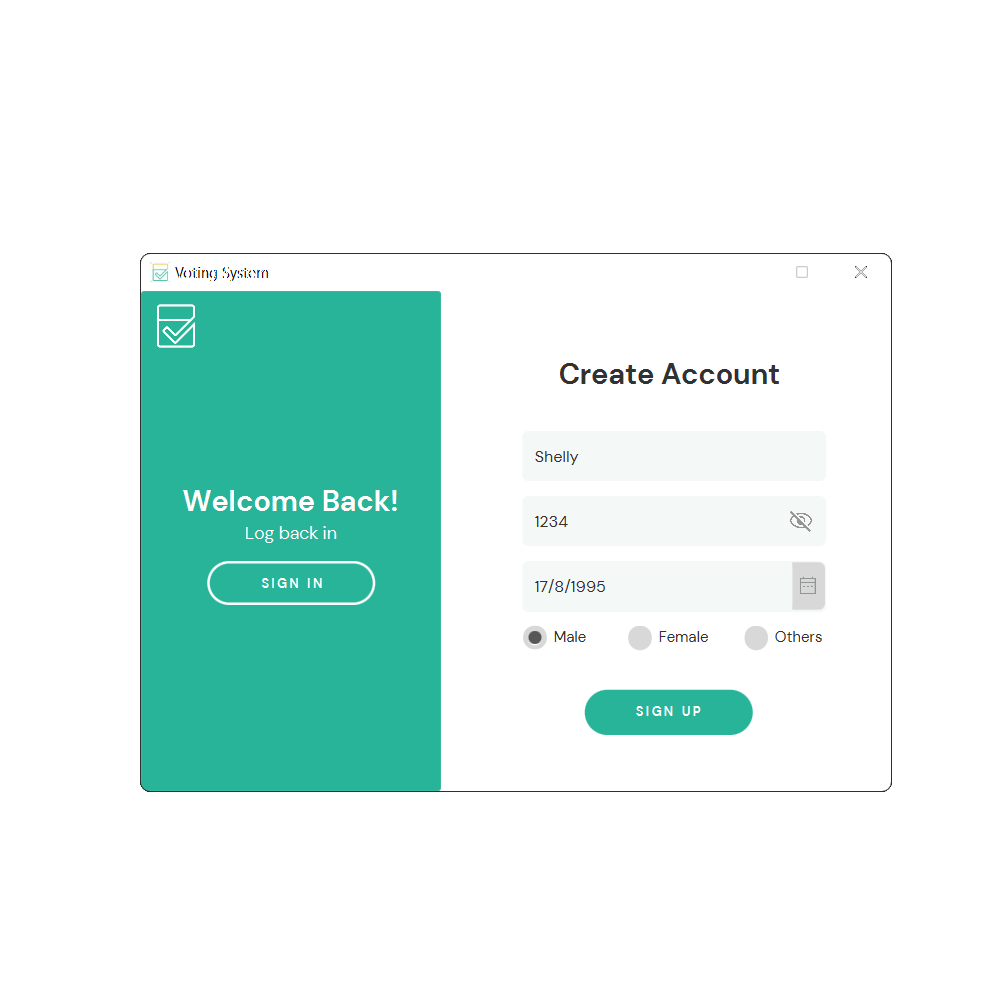
Under the src folder, it contains, application that has the main java file, all the classes, as well as the stylesheet (css) file. Controller contains all the JavaFX controllers. DBConnections contains the Handler class to connect to the database. FXML contains all the FXML files for the different scenes. img contains all the images used in the project.

#### Usage

Login Main Screen, user can sign in or go to the sign-up page.

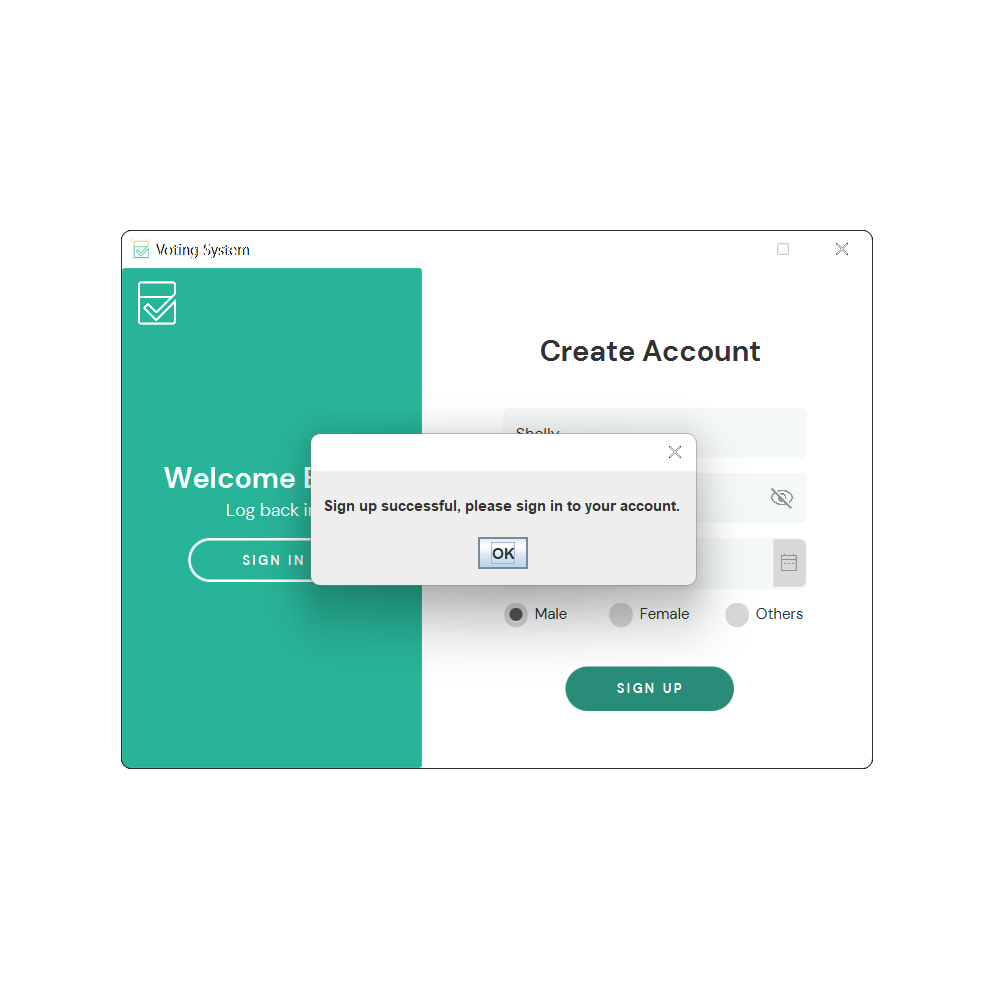


Sign up Screen for new users to create an account.

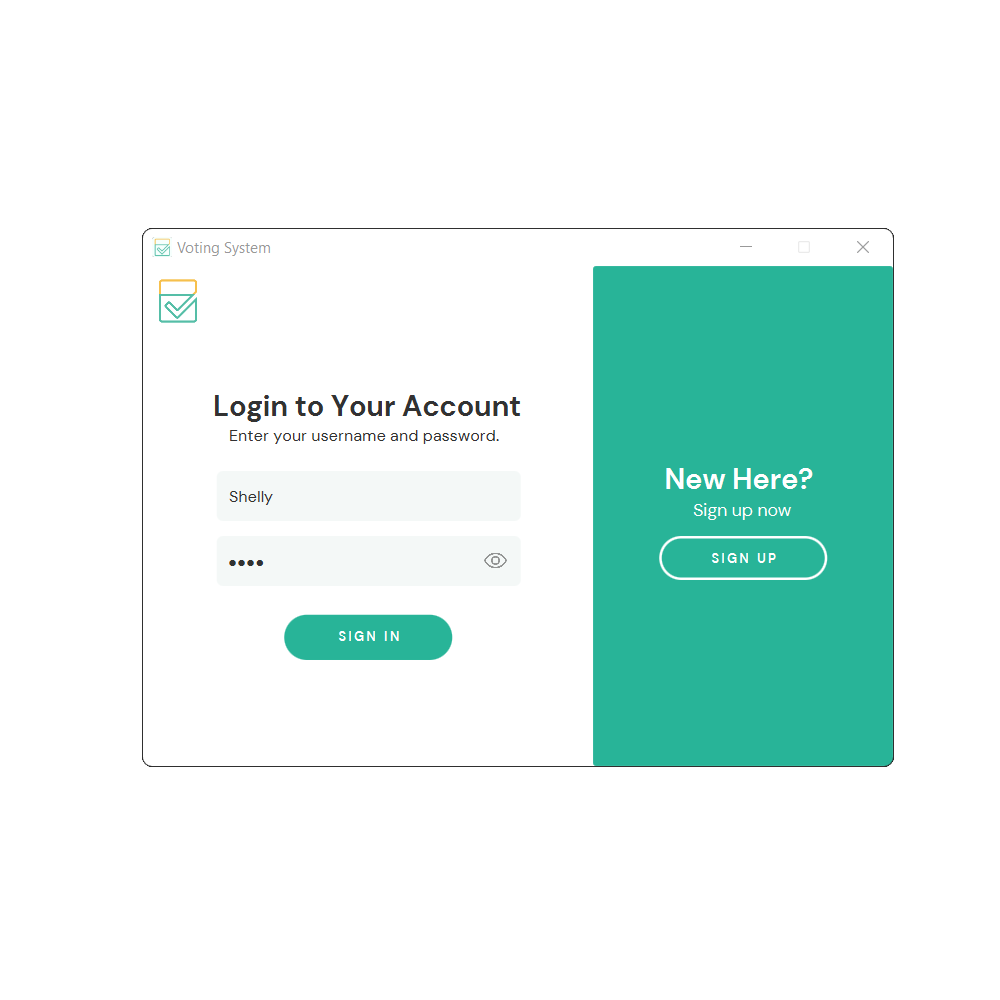


A new user signs up to create a new account. New users need to enter their name, create a password, their date of birth, as well as their gender.

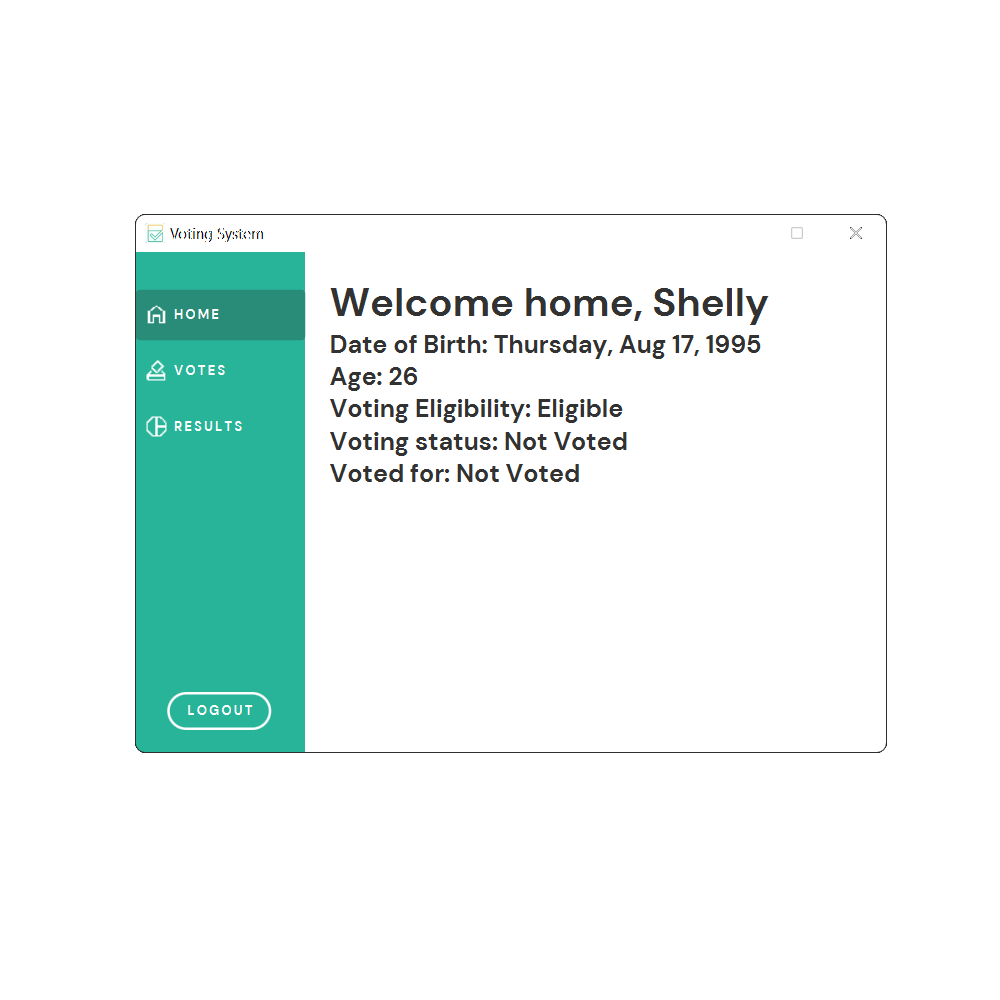
The eye logo allows the user to see the hidden password.



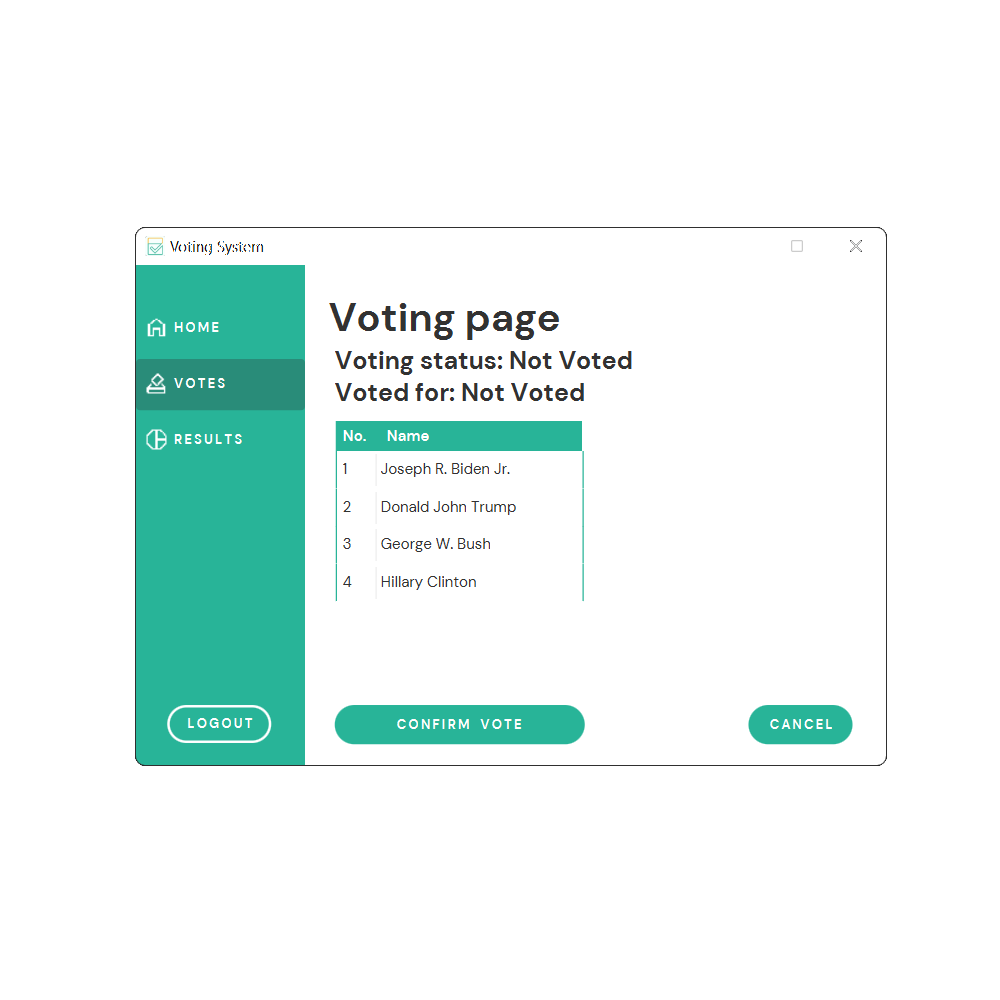
Prompt to after creating an account, to log in.



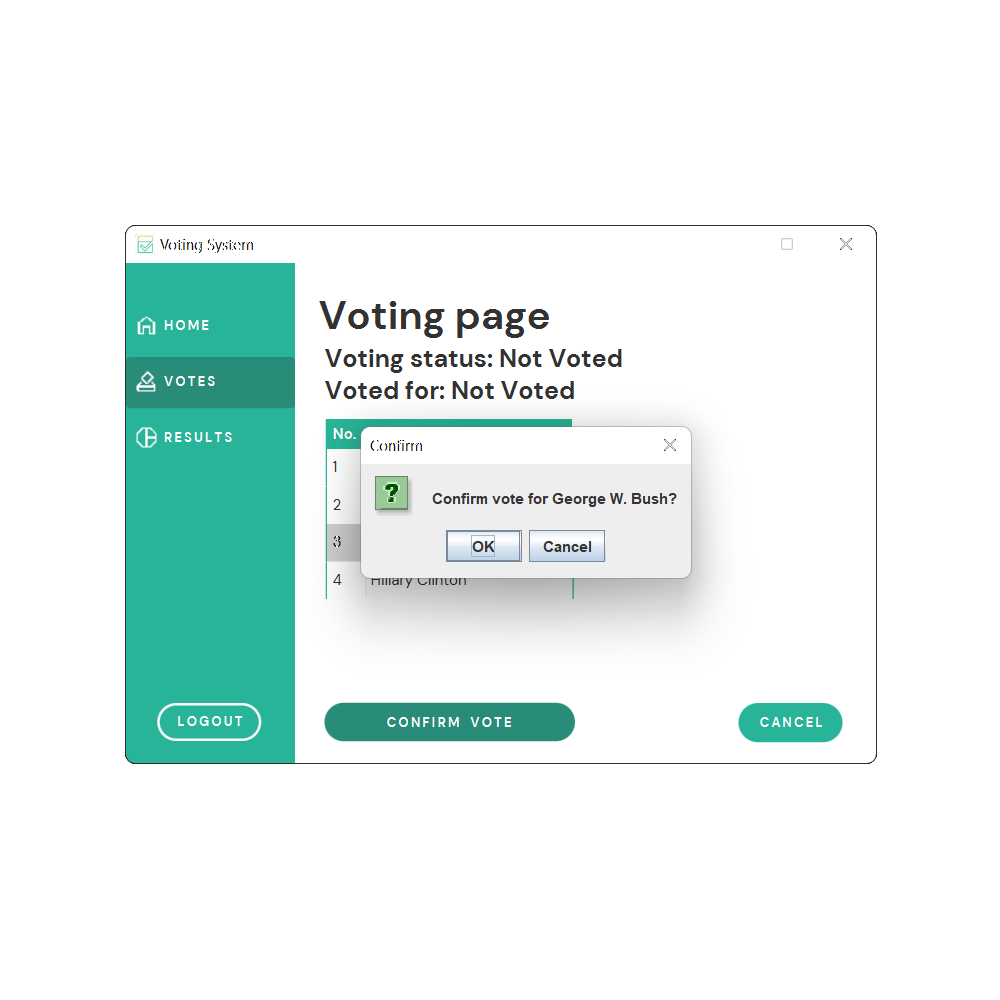
User logging into their account.



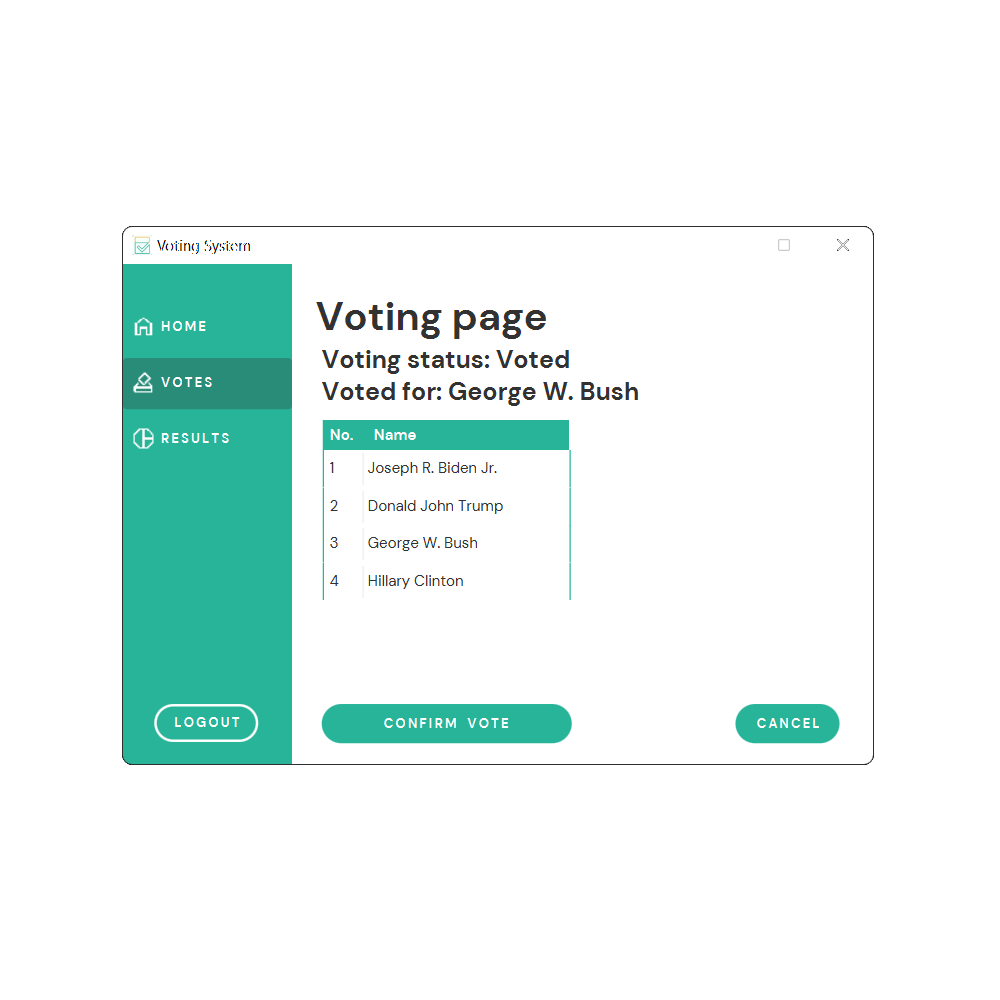
The home screen for the user, shows all the user information from the name, date of birth, age, voting eligibility, voting status, and voted for. Users can go to the votes screen, the results screen or they can also log out.



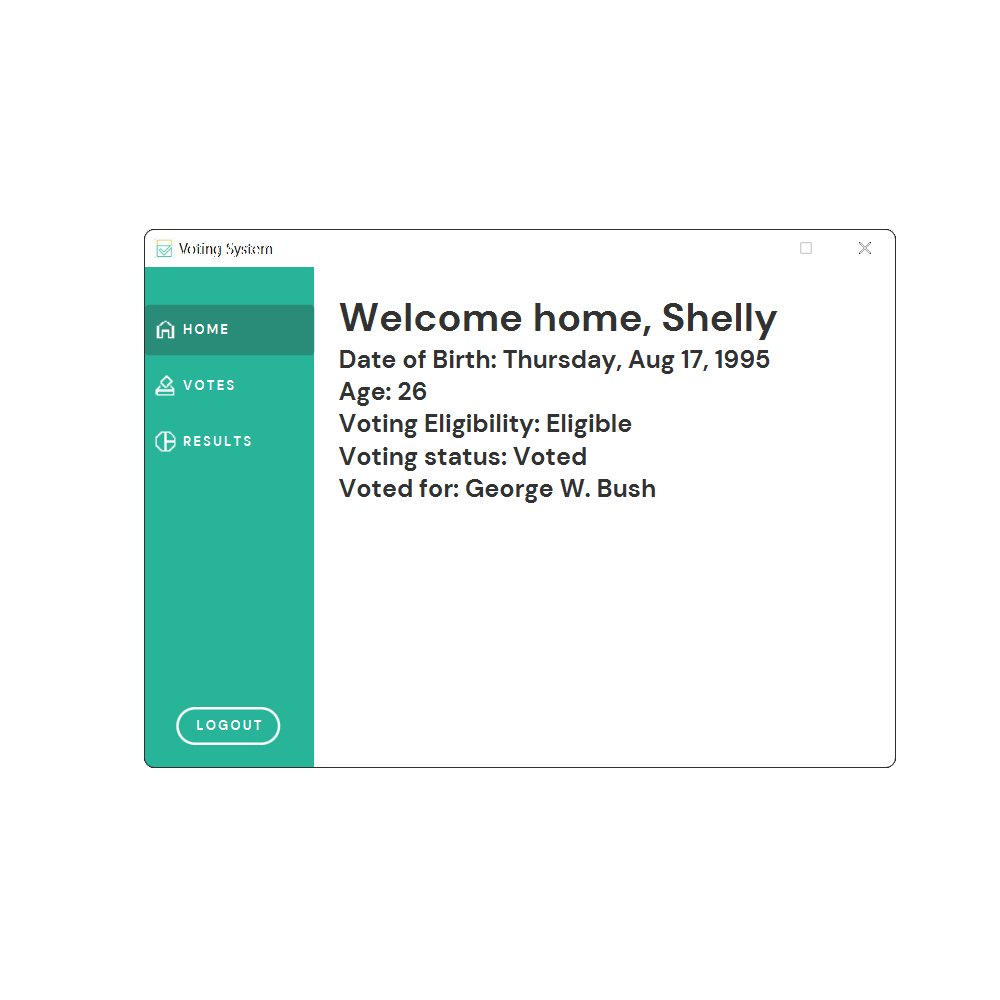
The voting page allows the user to vote and confirm their votes to change their vote status.



After the user has selected their vote, a prompt message is shown to confirm the vote.



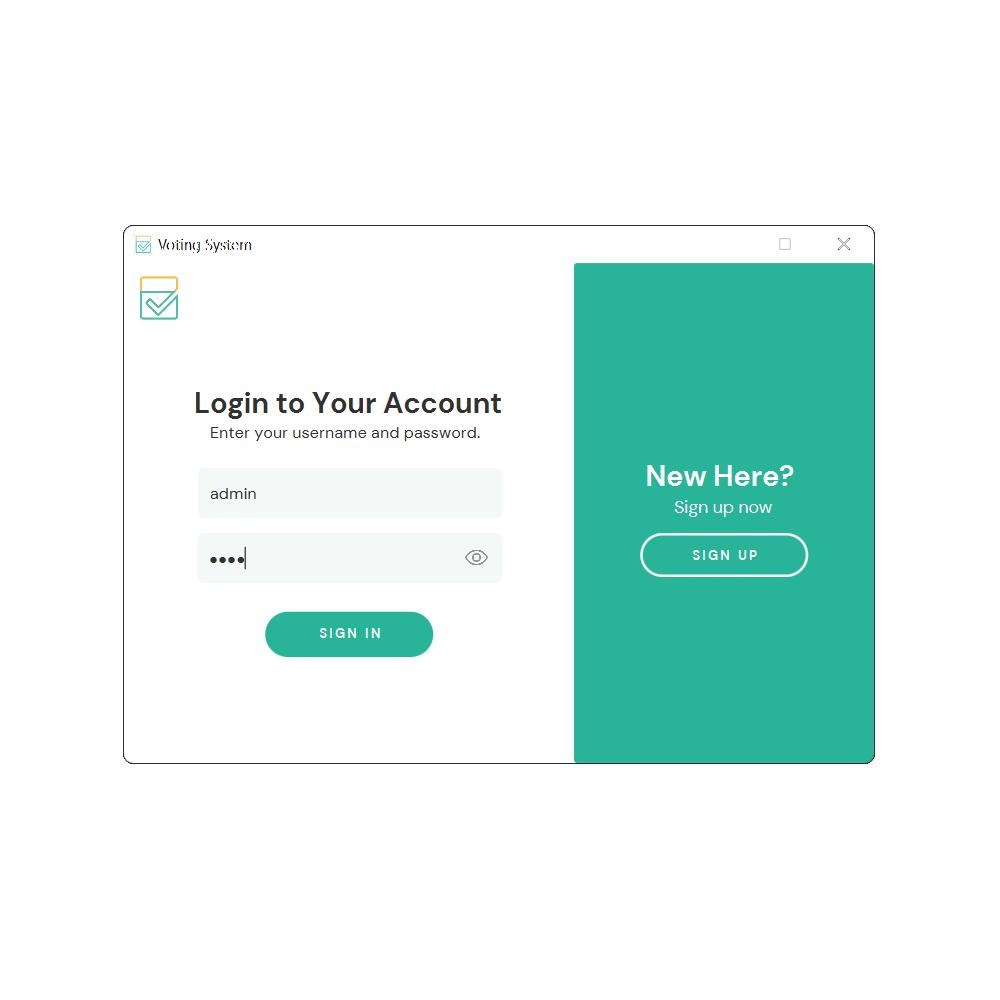
The voting screen will refresh and show the new information.



The home screen will also get updated after voting showing all the new information.

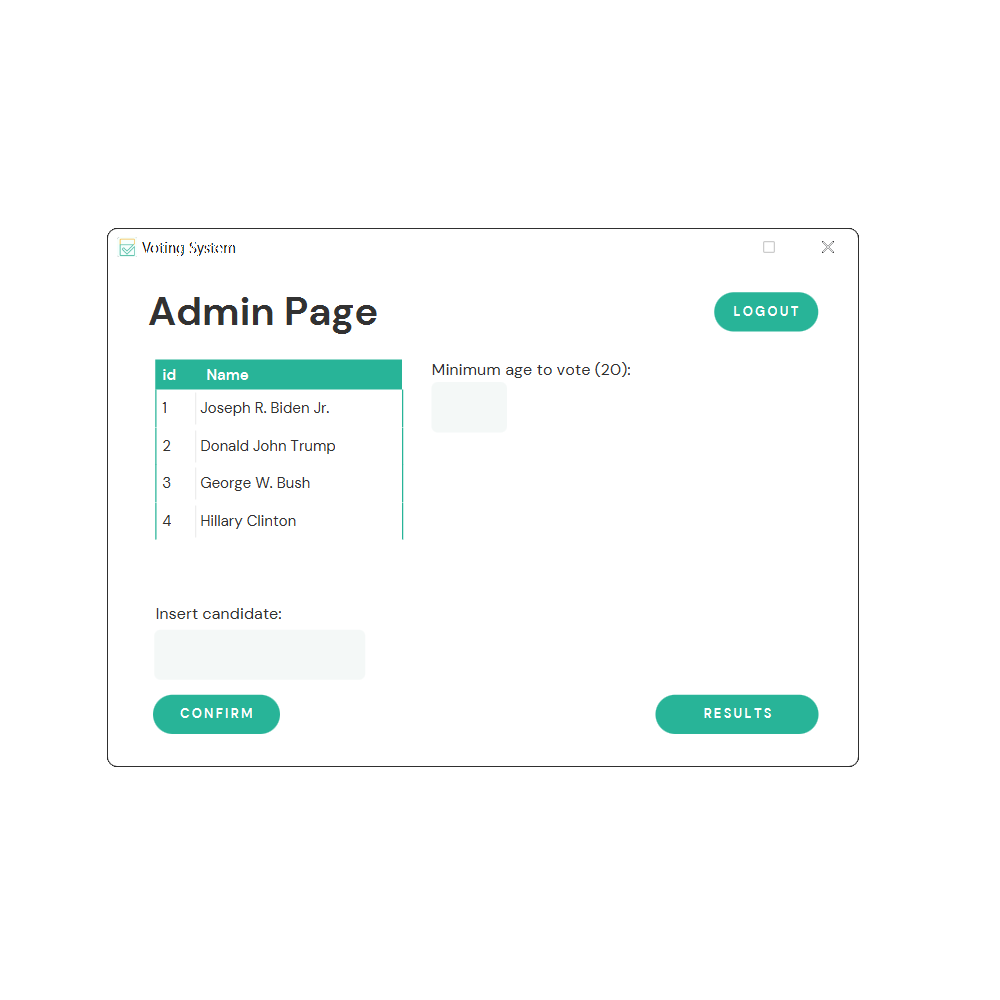


The final vote screen displays the results, it also shows the user’s basic information.

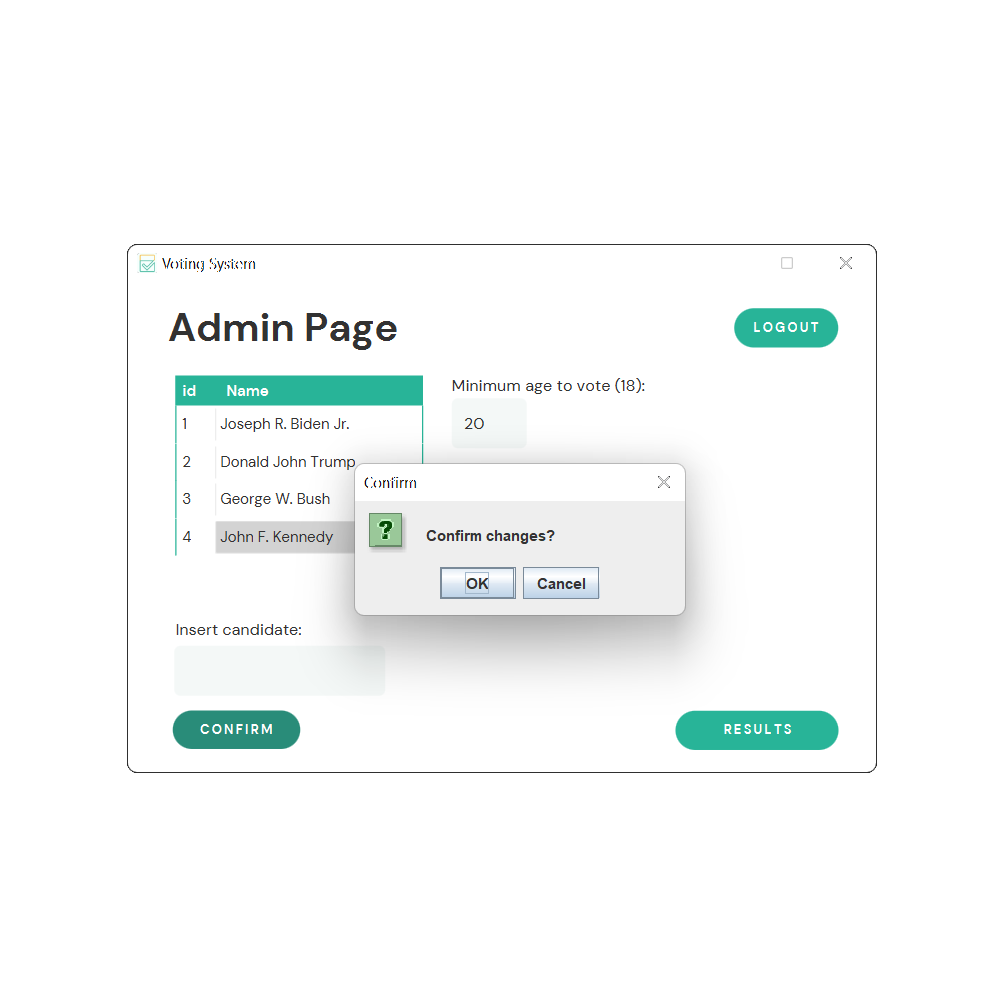


Admin Login with the username entered as “admin” and password as “1234”

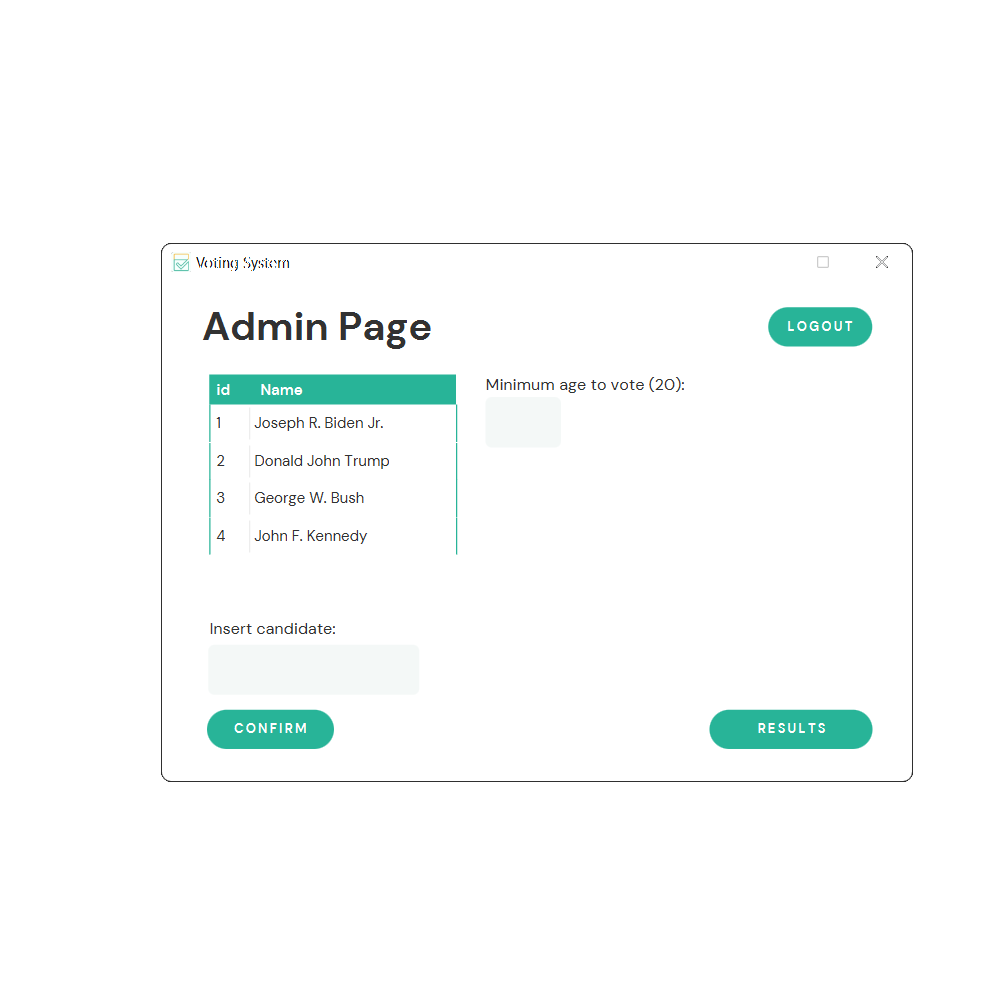
# 



Admin page screen, here the admin can change the candidate name, add new candidates, as well as change the minimum required age to vote. From here the admin can also see the results or final votes screen or logout.



After changing the name of the candidate and changing the minimum age to be eligible to vote, the admin is prompt with a confirm message.



Admin page after the admin changed the candidate name and minimum age.

# 

# Additional Library

#### Structured Query Language (SQL)

is a standardized programming language that is used to manage databases and perform various operations on the data in them, the application I used to access the SQL database tables is the MySQL Workbench 8.0 CE.

#### JDBC (Java Database Connectivity)

This library is used as it is the Java API that manages the connection to the SQL database, issuing queries and commands, and handling result sets obtained from the database.

# Resources

* 1. Google Fonts. 2022. *Material Symbols and Icons - Google Fonts*. [online] Available at: <<https://fonts.google.com/icons>>.
  2. www.javatpoint.com. 2022. *JavaFX Tutorial - javatpoint*. [online] Available at: <<https://www.javatpoint.com/javafx-tutorial>>.
  3. www.javatpoint.com. 2022. *JDBC Tutorial | What is Java Database Connectivity(JDBC) - javatpoint*. [online] Available at: <<https://www.javatpoint.com/java-jdbc>>.
  4. Mysql.com. 2022. *MySQL :: MySQL Workbench*. [online] Available at: <https://www.mysql.com/products/workbench/> [Accessed 18 July 2022].
  5. W3schools.com. 2022. *SQL Tutorial*. [online] Available at: <<https://www.w3schools.com/sql/>>.
  6. Mysql.com. 2022. *MySQL :: MySQL Connectors*. [online] Available at: <<https://www.mysql.com/products/connector/>>.
  7. App.diagrams.net. 2022. *Flowchart Maker & Online Diagram Software*. [online] Available at: <<https://app.diagrams.net/>>.
  8. Youtu.be. 2021. *JavaFX GUI Course ☕【𝙁𝙧𝙚𝙚】*. [online] Available at: <<https://youtu.be/9XJicRt_FaI>>.