



Cardiff
Metropolitan
University

Prifysgol
Metropolitan
Caerdydd

Cardiff School Of Technologies

Student ID: 20317001

***Module Name: Computer Science and
Applications***

Module Code: CMP4012

Project Title : Online Voting System (Prac1)

Module Leader: Dr Rajkumar Singh Rathore

Submitted By: Ahsanul Haque Mamun

Table of contents

1. Application Introduction.....	3
2. Requirements Analysis & Specification.....	4
3. System Design & Architecture	6
4. Implementation.....	7
i. Core Implementation	7
ii. Exception Handling.....	10
5. Testing and debugging.....	11
6. User Interface.....	13
7. Conclusion and Future work.....	15
8. Reference.....	16

1. Application Introduction

The Online Voting System is a desktop program written in Java that uses Java Swing, AWT components, and standard I/O features. Its main goal is to provide a safe and easy-to-use interface that lets people register and vote online while still letting administrators keep an eye on things. This app shows basic Java programming ideas, like object-oriented design, event-driven interaction, file handling, and graphical user interface (GUI) development. Its goal is to model a small-scale digital election system.

The application is structured around two user roles: **Users** and **Admins**. Regular users can register with their name, email, age, and gender, then cast a vote for one of the predefined candidates. Admins, on the other hand, can access features like viewing the list of registered users, resetting all vote counts, and reviewing the voting history to see who voted for whom. These functionalities are managed through separate GUI windows, each implemented as distinct JFrame subclasses to ensure modularity and maintainability. (Services, 2023)

Java was chosen as the programming language due to its portability, strong GUI capabilities via Swing, and ease of file-based persistence using streams and readers. The system uses basic file I/O to store user profiles (user_profiles.txt) and voting history (votes.txt), demonstrating Java's ability to manage persistent data without external databases. The application was developed using Java OpenJDK via Amazon Corretto, a production-ready distribution widely supported by the Java community. (Series, 2023)

This project demonstrates a real-world application of Java in the development of secure, interactive desktop software. Core programming concepts such as class encapsulation, interface implementation (ActionListener), and exception handling are all applied within the codebase. Additionally, Java's component layout managers and event-handling mechanisms are used to build responsive forms and buttons that react to user interactions.

Overall, this system not only highlights the practical application of Java in civic technology but also promotes a paperless and accessible voting method. While simplified for educational purposes, the structure is scalable for enhancements like password-protected logins, real-time vote analytics, and database integration, making it an excellent foundation for more complex voting systems in the future.

2. Requirements Analysis & Specifications

The Online Voting System is designed to replicate a secure and structured voting process through a desktop-based Java application. It serves two user roles: general users and system administrators. This section outlines the key functional and non-functional requirements that define what the system should and should not do.

Functional Requirements:

- Users must be able to register by providing their full name, email address, age, and gender.
- Upon successful registration, users gain access to a voting interface and may vote once for a listed candidate.
- Each vote is logged and stored persistently using local file handling.
- The admin must log in with valid credentials to access the administrative dashboard.
- The admin can view all registered user profiles, reset the total vote count, and access detailed records of who voted for whom.

Non-Functional Constraints:

- The system must validate user input to avoid incomplete or duplicate registrations.
- Users are strictly limited to one vote to maintain election integrity.
- Administrative access must be restricted through secure login credentials.
- Data storage is confined to local .txt files; database or web connectivity is not implemented in this version.

This version is intended for small-scale, offline use and is not intended for real-time or web-based elections.

Use case Diagram :

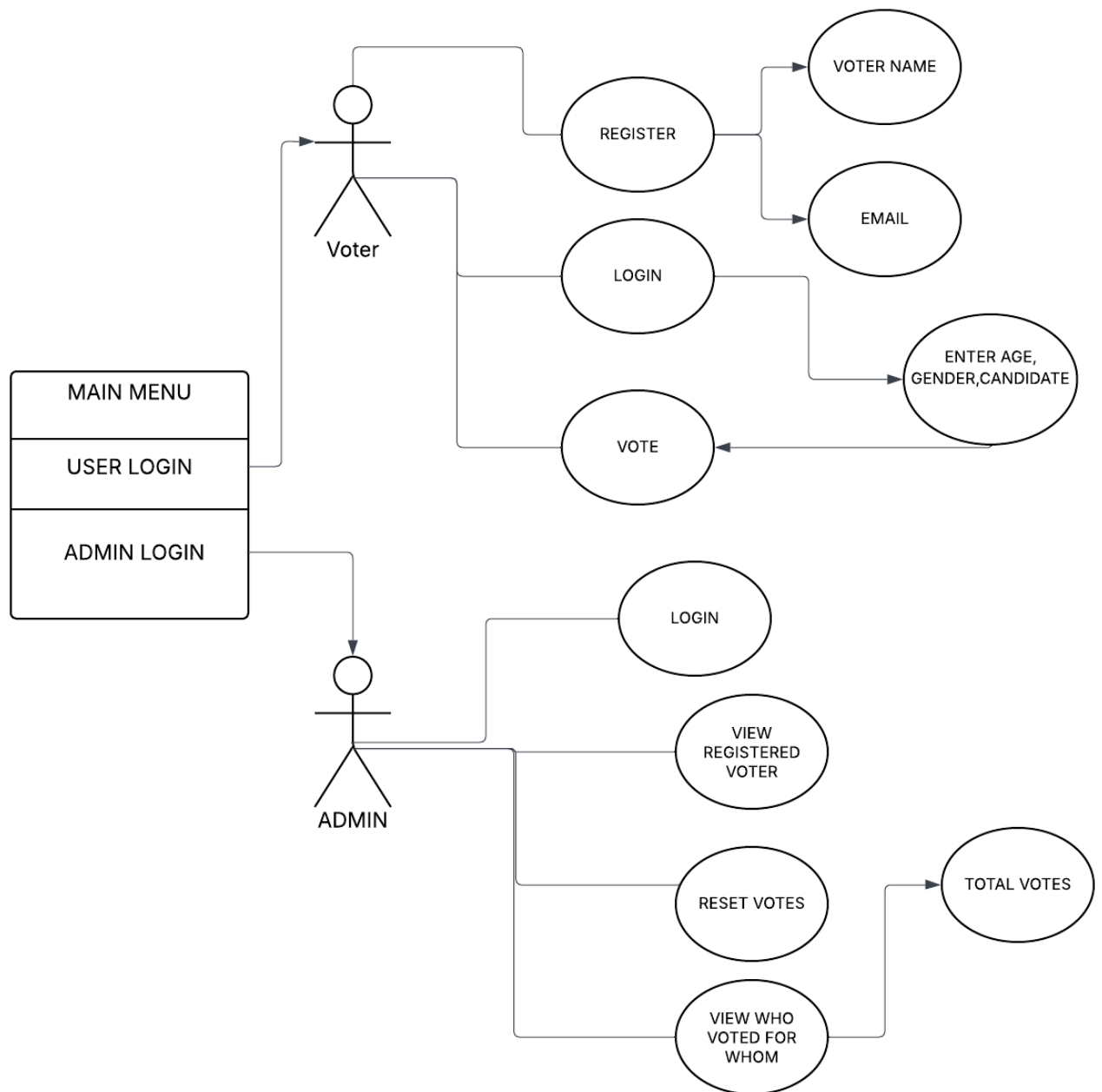


Figure 1: UML Use case diagram

Source: Lucid Chart (chart, 2024)

3. System Design & Architecture

The architecture of the app is modular, meaning that different classes handle different user tasks and functions. The graphical user interface (GUI) is made with Java Swing, and all exchanges with the user are handled by the ActionListener interface through event handling. The system is meant to be easy to use, easy to manage, and easy to add on to. At its heart, the OnlineVotingSystem class has the main() method, which starts the MainMenu interface and sets up the program. The MainMenu is where you can find your way around, and it has choices for "User Login" and "Admin Login." When people make a choice, they are taken to their own interfaces. (Services, 2023)

Class Structure:

- **OnlineVotingSystem:** The entry point of the program containing the main() method. It launches the MainMenu.
- **MainMenu:** Offers users a choice between "User Login" and "Admin Login."
- **UserLogin:** Captures basic user credentials and transitions to the UserPanel.
- **UserPanel:** Handles user registration, input validation, and voting. User details are saved to user_profiles.txt, and votes to votes.txt. It restricts each user to one vote.
- **AdminLogin:** Validates admin credentials.
- **AdminPanel:** Grants access to administrative features such as viewing users, resetting votes, and displaying vote records.

Interface Usage:

This interface is used by all GUI classes to handle button click events. This lets people connect in real time, like showing confirmation messages or buttons after a successful registration. To make an interface that is engaging and responsive, you can use Swing components like JFrame, JButton, JTextField, JTextArea, JComboBox, and JScrollPane. Instead of using external databases, the system relies on BufferedReader, BufferedWriter, and PrintWriter to read and write to files locally.

Packages:

In this implementation, all classes reside in the default package for simplicity. However, for future scalability, classes could be separated into logical packages such as gui (user interfaces), model (data structures), and service (data handling and business logic).

4. Implementation

This section details the core implementation and exception handling features of the Online Voting System software application. The design is modular, clean, and beginner-friendly, showcasing effective use of Java programming fundamentals.

i. Core Implementation:

An object-oriented computer language called Java is used to build the Online Voting System. The app is made up of modular classes, where each class holds its own set of functions, and a dynamic UI made with Java Swing components.

The main() method is in the main class, OnlineVotingSystem, which is also where the program starts. It starts the MainMenu screen when it's run, which is the system's navigator and lets the user choose between "User Login" and "Admin Login." Every button in the GUI is connected to an ActionListener that opens the right panel when the button is clicked.(Stack Overflow, 2024)

The UserLogin class captures user details like name and email. Once submitted, it passes this information to the UserPanel class. UserPanel is responsible for handling full registration (including age and gender), displaying a candidate list using a JComboBox, and processing the vote.

```

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == registerButton) {
        String age = ageField.getText().trim();
        String gender = (String) genderBox.getSelectedItem();

        if (age.isEmpty()) {
            JOptionPane.showMessageDialog(parentComponent: this, message: "Please enter your age.");
            return;
        }

        isRegistered = true;
        voteButton.setEnabled(true);
        registerButton.setEnabled(false);
        ageField.setEditable(false);

        saveUserProfile(username, email, age, gender);
        JOptionPane.showMessageDialog(parentComponent: this, message: "Registration successful!");
    }
}

```

Figure 1: User Registration Process Code

Voting is disabled until registration is complete. Once a vote is cast, it is stored using `PrintWriter` in a file named `votes.txt`. Registration data is similarly saved in `user_profiles.txt`. The system handles user data and vote recording using `FileWriter` and `BufferedReader` classes, following examples provided by (GeeksforGeeks, 2024)

```

else if (e.getSource() == voteButton && isRegistered) {
    String candidate = (String) candidateList.getSelectedItem();
    votes.put(candidate, votes.get(candidate) + 1);
    resultArea.setText("✅ You voted for: " + candidate);
    saveVoteRecord(username, candidate);
    voteButton.setEnabled(false);
}

```

Figure 2: Voting Function and Vote Saving Logic

The AdminLogin class prompts for credentials. If validated successfully, it opens the AdminPanel, which provides administrators with the ability to view all registered users, reset vote counts, and view a list of who voted for whom.

```
// === Admin Login ===
class AdminLogin extends JFrame { 1usage
    public AdminLogin() { 1usage
        setTitle("Admin Login");
        setSize( width: 300, height: 200);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new GridLayout( rows: 3, cols: 2));

        JTextField adminField = new JTextField();
        JPasswordField passField = new JPasswordField();

        add(new JLabel( text: "Admin ID:"));
        add(adminField);
        add(new JLabel( text: "Password:"));
        add(passField);

        JButton loginBtn = new JButton( text: "Login");
        add(new JLabel( text: ""));
        add(loginBtn);

        loginBtn.addActionListener( ActionEvent e -> {
            String admin = adminField.getText();
            String pass = new String( passField.getPassword());

            if (admin.equals("admin") && pass.equals("admin123")) {
                dispose();
                new AdminPanel();
            } else {
                JOptionPane.showMessageDialog( parentComponent: this, message: "Incorrect credentials.");
            }
        });
    }
}
```

Figure 3:AdminPanel with administrative controls.

The AdminPanel reads and writes from the same files used by users. This demonstrates consistent data handling across both roles using Java’s BufferedReader, BufferedWriter, and file-based streams.

All classes implement Java’s ActionListener to handle UI events. Buttons like “Register”, “Vote”, and “Reset Votes” trigger specific logic through the actionPerformed() method. Swing components such as JButton, JTextField, JTextArea, JLabel, and JComboBox are used extensively to design the interface, demonstrating an understanding of GUI programming in Java.

The system ensures that a user cannot vote more than once. Once registered and the vote is cast, the “Vote” button is disabled, and form fields are locked. This enforces a single-vote-per-user rule. Furthermore, the modular design allows for easy future expansion, such as integrating encryption for vote privacy, session tracking, or multi-language support. Each GUI panel operates independently, improving testability and maintainability. Responsiveness is achieved

by linking each user action to appropriate logic through clear and concise event-handling methods.

ii. Exception Handling:

The whole system has strong exception handling built in to handle file I/O tasks and bad user input. When reading or writing to files like votes.txt and user_profiles.txt, try-catch blocks are used to catch IOException events and show them. This keeps the program from crashing if files are missing or there are read/write failures. Try-catch blocks were used to make sure that file actions would work even if something went wrong, so the program wouldn't crash. (GeeksforGeeks, 2024)

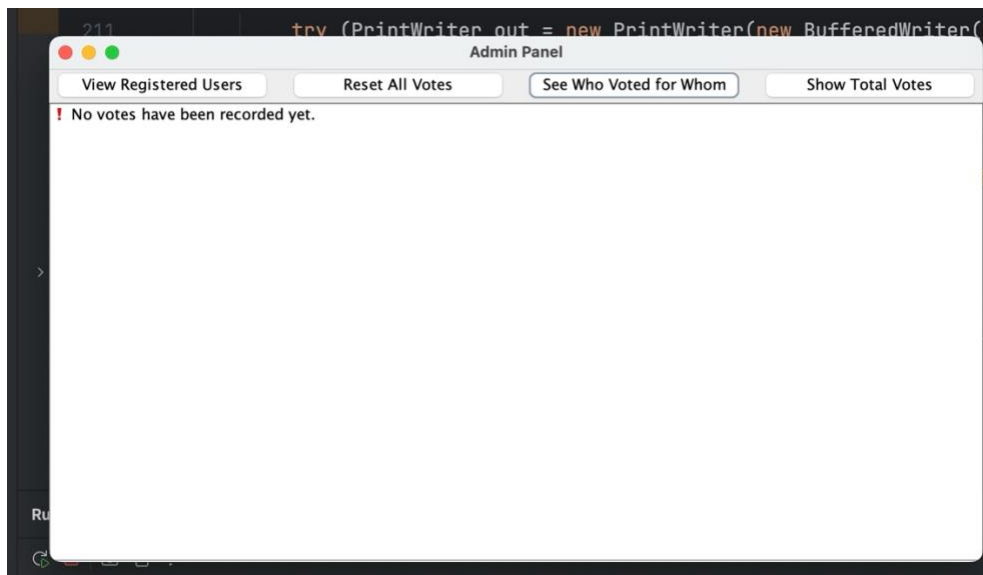


Figure 4 : Exception handling for missing vote records.

In the UserPanel, input fields are validated before submission. If any required field is left empty, a JOptionPane alert is triggered informing the user to complete all fields. This prevents incomplete data from being stored.

The admin login is protected with a hardcoded username/password (admin / admin123). If incorrect credentials are entered, the system provides user feedback using dialog boxes and denies access.




This comprehensive implementation of GUI event handling, file I/O, validation, and error reporting ensures the system operates reliably while maintaining a user-friendly experience.

5. Testing and Debugging

Testing and debugging were essential phases in the development of the Online Voting System to ensure functionality, reliability, and a smooth user experience. The system was tested manually through a series of test cases for both user and admin roles, covering registration, voting, data saving, and access control

Table 1: Test Case Matrix for Online Voting System

TEST CASES	INPUT	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC01	Name: "Ahsan", Email: "ahsan@mail.com"	User logged in and User Panel opens	As expected,	Pass
TC02	Name: "", Email: "ahsan@mail.com"	Error message: "Invalid name or email."	As expected,	Pass
TC03	Admin ID: "admin", Password: "admin123"	Admin Panel opens	As expected,	Pass
TC04	Admin ID: "admin", Password: "wrongpass"	Error message: "Incorrect credentials."	As expected,	Pass
TC05	Age: "22", Gender: "Male" (after login)	Registration success message shown, vote button enabled	As expected,	Pass
TC06	Age: "" (blank), Gender: "Female"	Error message: "Please enter your age."	As expected,	Pass
TC07	Click "Vote" before registration	Vote button disabled	As expected	Pass
TC08	Select candidate "Piyush" and click Vote after registering	Message: "✅ You voted for: Piyash"	As expected	Pass
TC09	Re-click Vote after already voting	Vote button disabled, vote not accepted again	As expected	Pass
TC10	Admin clicks "View Registered Users"	Shows contents of user_profiles.txt	As expected	Pass

TC11	Admin clicks “See Who Voted for Whom”	Shows contents of votes.txt	As expected	Pass
TC12	Admin clicks “Show Total Votes”	Message: “  Total Votes Casted: X”	As expected	Pass
TC13	Admin clicks “Reset All Votes”	votes.txt cleared, message: “  All votes have been reset.”	As expected,	Pass
TC14	votes.txt deleted, Admin clicks “See Who Voted for Whom”	Message: “  No votes have been recorded yet.”	As expected,	Pass

Unit Testing

Each method and action were tested individually during development. The user registration and vote processing functionalities were tested by inputting both valid and invalid data. For example, test cases were written to ensure that:

- Users cannot vote without registering.
- Fields such as name, email, and age cannot be left empty.
- The vote is successfully saved and reflected in votes.txt.

The admin panel was tested for correct credential validation and file read operations. Testing confirmed that the admin can view users and vote records only if the correct username and password are entered.

Bug Identification & Fixing

During development, several bugs were found and fixed. One big problem was that the app wouldn't read from votes.txt if the file didn't exist. It was fixed by making sure the file existed before reading it and showing a custom warning message instead of crashing.

Another issue was users being able to click the “Vote” button multiple times. This was resolved by disabling the button after the vote was recorded.

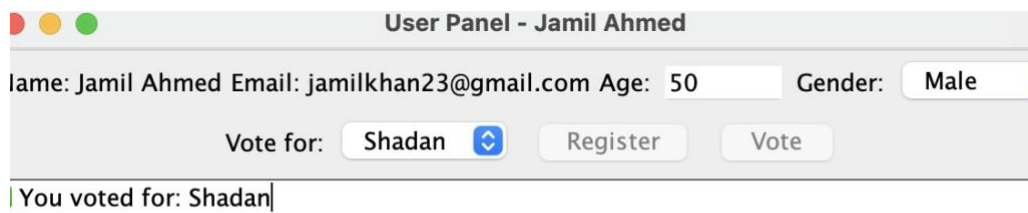


Figure 5 : Vote button disabled post-vote to prevent duplicates.

Overall, the application passed all manual test cases and behaves reliably under normal use.

6. User Interface

The user interface (UI) of the Online Voting System is designed using Java Swing, prioritising simplicity, clarity, and user-friendliness. Each user role—User and Admin—is provided with a clean, intuitive layout, followed principles on (W3Schools, 2024). The Main Menu serves as a navigation hub, while the UserPanel includes labelled fields, dropdowns, and clear instructions for registration and voting. The AdminPanel presents administrative controls with clearly labeled buttons and a scrollable display area for results and records. Visual feedback such as confirmation messages and error pop-ups enhance usability. The interface is fully responsive to user actions and maintains accessibility across all screen sizes. GUI interface with screenshot is given below.

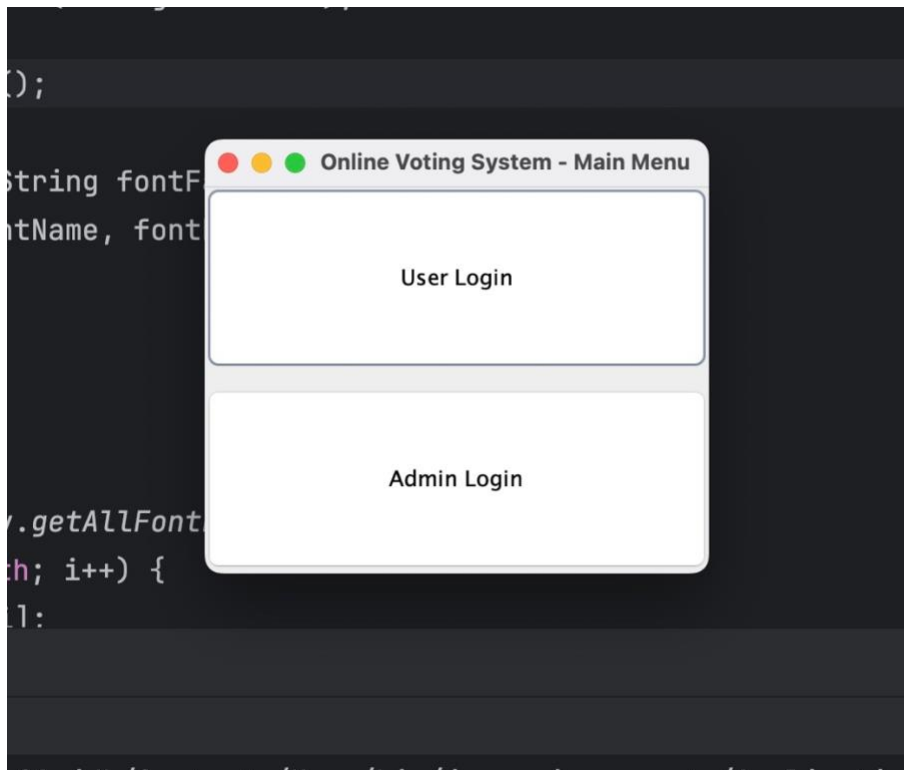


Figure 6: Main Menu of the Online Voting System

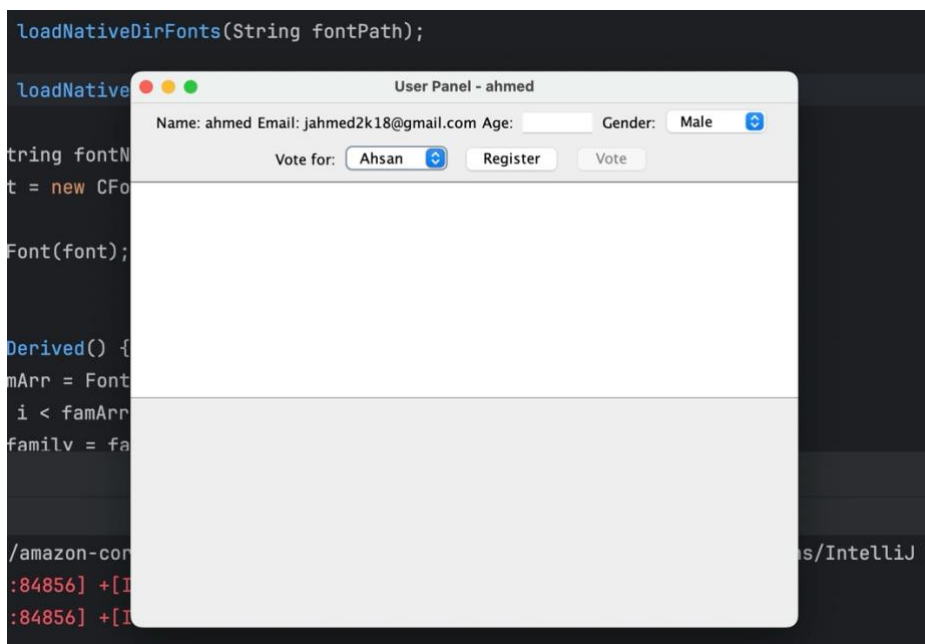


Figure 7: User Panel Interface with Registration and Voting Options

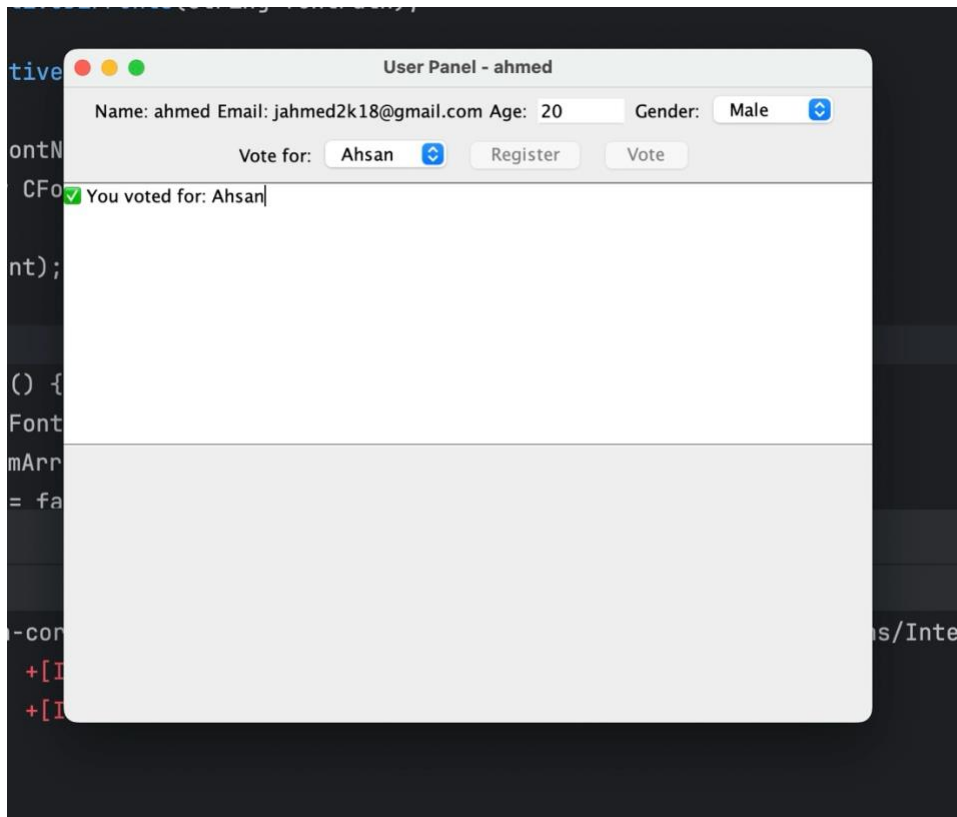


Figure 8: Vote Confirmation Message in User Panel

7. Conclusion & Future work

Core Java programming ideas, such as GUI development, event-driven logic, file handling, and role-based access control, are successfully used in the Online Voting System. This system works like a secure voting system, letting users sign up and cast their ballots while managers control and keep an eye on the system. The program makes sure it is reliable and easy to use by carefully implementing, validating input, and handling errors.

Even though the current version does what it's supposed to do, it could be much better in the future. Using hashed passwords for role-based authentication and protected logins could make security better. Instead of text files, a relational database like MySQL could be used to improve data longevity. A voting data dashboard and dynamic vote charts could also help administrators understand things better. Adding network support to the system to make it work in online or spread settings would also make it more useful in the real world. This project builds a strong base for digital vote apps that can be used by many people.

8. References

- chart, L., 2024. *Create UML Class Diagrams Online*. [Online]
Available at: <https://www.lucidchart.com/pages/uml-class-diagram>
[Accessed 8 May 2025].
- GeeksforGeeks, 2024. *Java File Handling (With Examples)*. [Online]
Available at: <https://www.geeksforgeeks.org/file-handling-in-java/>
[Accessed 7 May 2025].
- GeeksforGeeks, 2024. *Java Swing Tutorial*. [Online]
Available at: <https://www.geeksforgeeks.org/java-swing/>
[Accessed 7 May 2025].
- Series, A. W., 2023. *Amazon Corretto: Production-ready distribution of OpenJDK*. [Online]
Available at: <https://aws.amazon.com/corretto/>
[Accessed 7 May 2025].
- Services, A. C. A. W., 2023. *Amazon Corretto: Production-ready distribution of OpenJDK*. [Online]
Available at: <https://aws.amazon.com/corretto/>
[Accessed 7 May 2025].
- Stack Overflow, 2024. *How to use ActionListener with buttons in Java*. [Online]
Available at: <https://stackoverflow.com/questions/>
[Accessed 7 May 2025].
- W3Schools, 2024. *Java Tutorial*. [Online]
Available at: <https://www.w3schools.com/java/>
[Accessed 7 May 2025].

You can access my Java Code by Clicking Here: [OnlineVotingSystem \(1\).java](#)