Voting Machine

Summary of work I did in developing voting machine for my group project.

In our group project to design and implement a fully functional voting machine, I played a key role across several aspects of the development process. I contributed significantly to the detailed design diagrams, ensuring that the overall system architecture was well-documented and comprehensible. My involvement included defining software requirement specifications, which outlined the necessary functionalities and performance criteria the voting machine needed to meet. This foundational work set the stage for the subsequent development phases.

I actively participated in creating the software architecture and design, focusing on developing robust logic diagrams and interfaces that facilitated smooth communication between different system components. My work in this area ensured that the software was structured in a modular and efficient manner, allowing for easier maintenance and scalability.

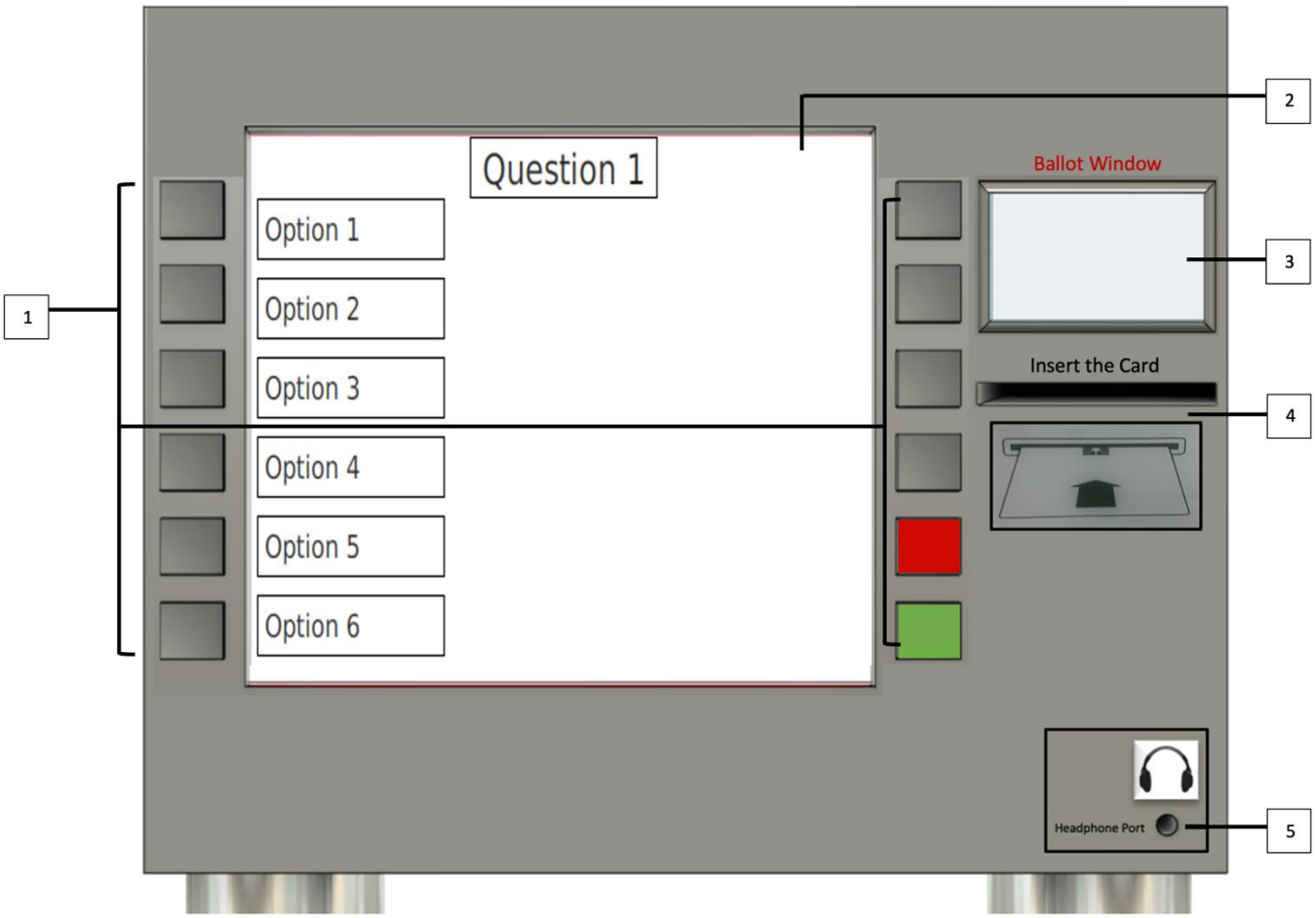
Moreover, I contributed to coding the voting machine software, where I provided critical code snippets and collaborated closely with other team members to integrate various modules. I also worked on developing a sample user interface (UI) for the voting machine, emphasizing user-friendly design and accessibility. Through collaborative efforts with interdisciplinary teams, I refined the UI to improve layout and usability, ensuring that the final product was intuitive and met user needs.

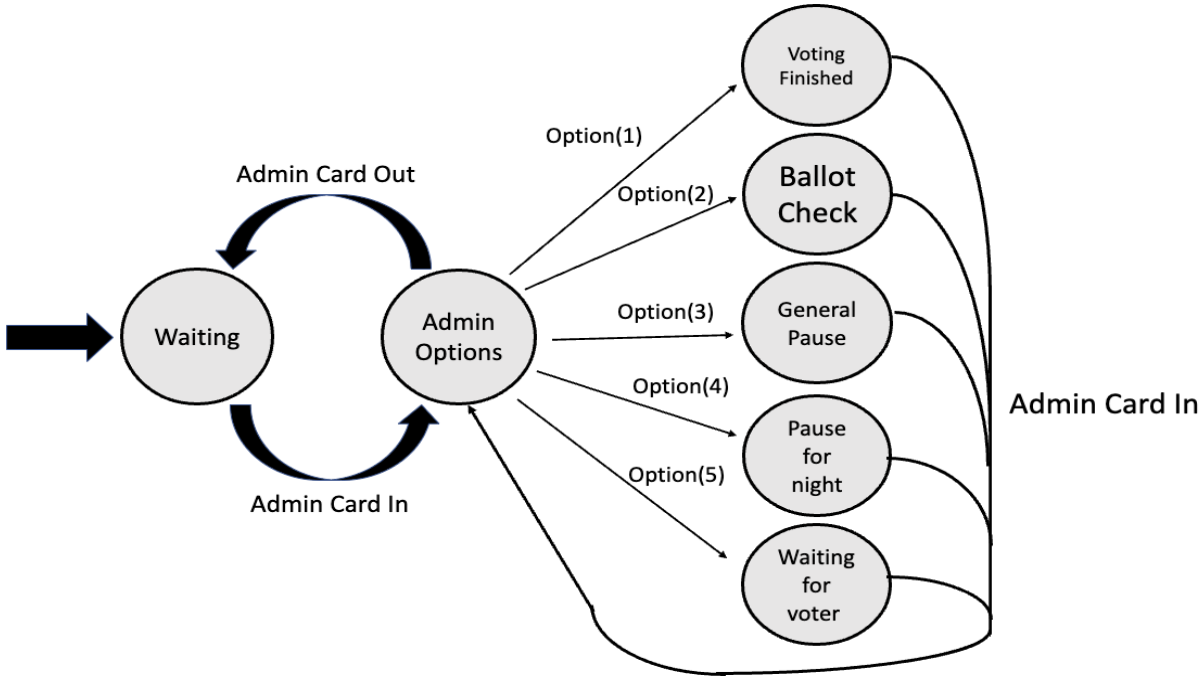
Throughout the project, I engaged in extensive research on voting machine technologies and best practices, bringing insights to group discussions and decision-making processes. This research informed our design choices and helped us align the machine's functionalities with current standards and expectations.

Overall, my contributions spanned from initial design and specifications to implementation and testing. I provided technical guidance, design expertise, and coding support, ensuring the project was completed successfully and met all defined objectives.

Figure: Voting Machine (Front)

Figure: Voting Machine (Back)


Question 1
Option 1
Option 2
Option 3
Option 4
Option 5
Option 6
Ballot Window
Insert the Card
Headphone Port
1
2
3
4
5



Machine States

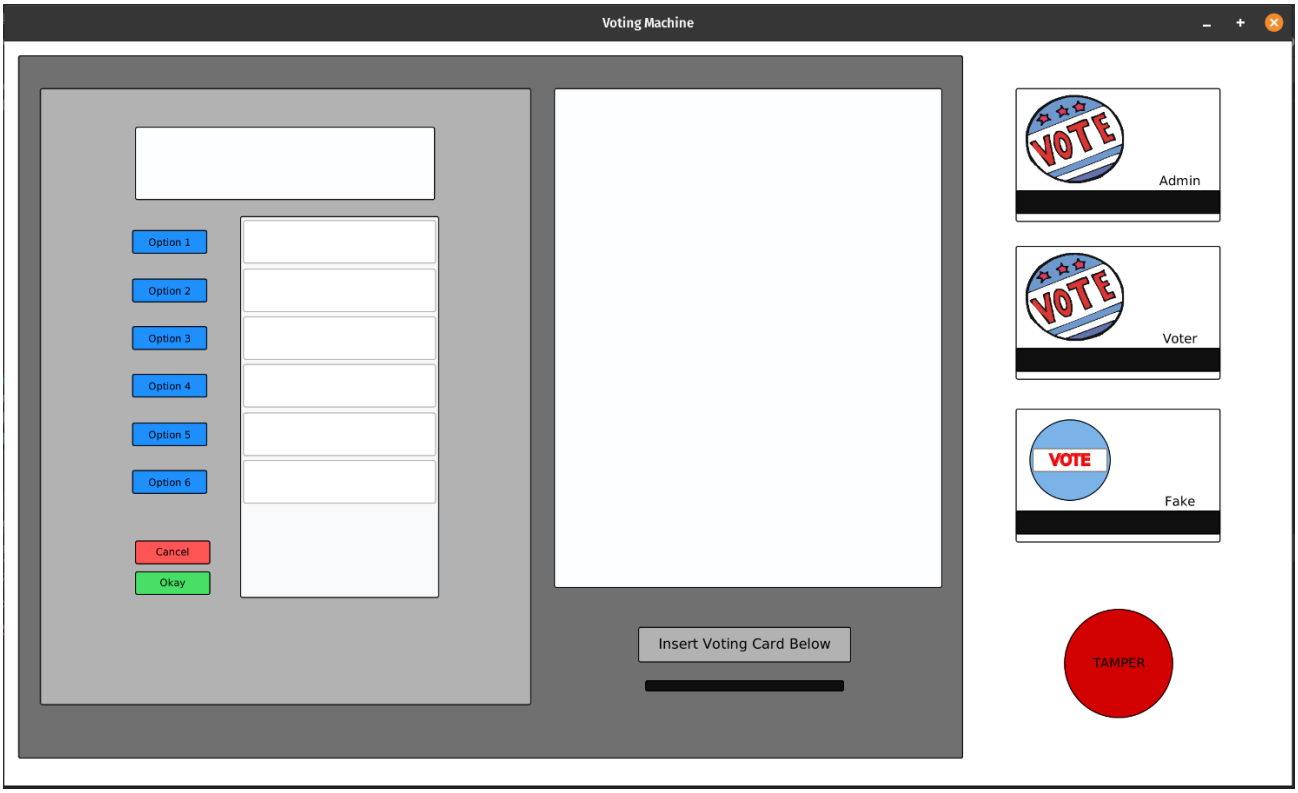


Figure 1.1: Opening screen on program start up

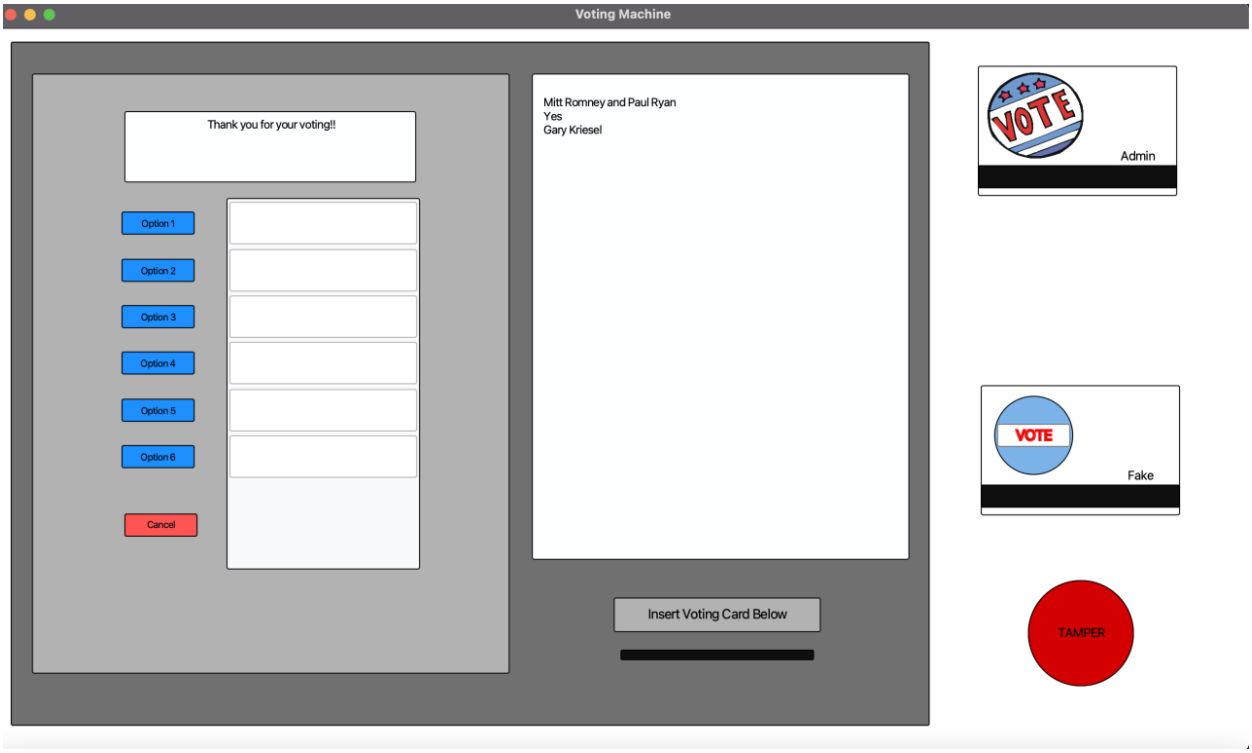


Figure 1.2: Final display after voter has confirmed final selections, printout of selections show to the right

Design Diagram

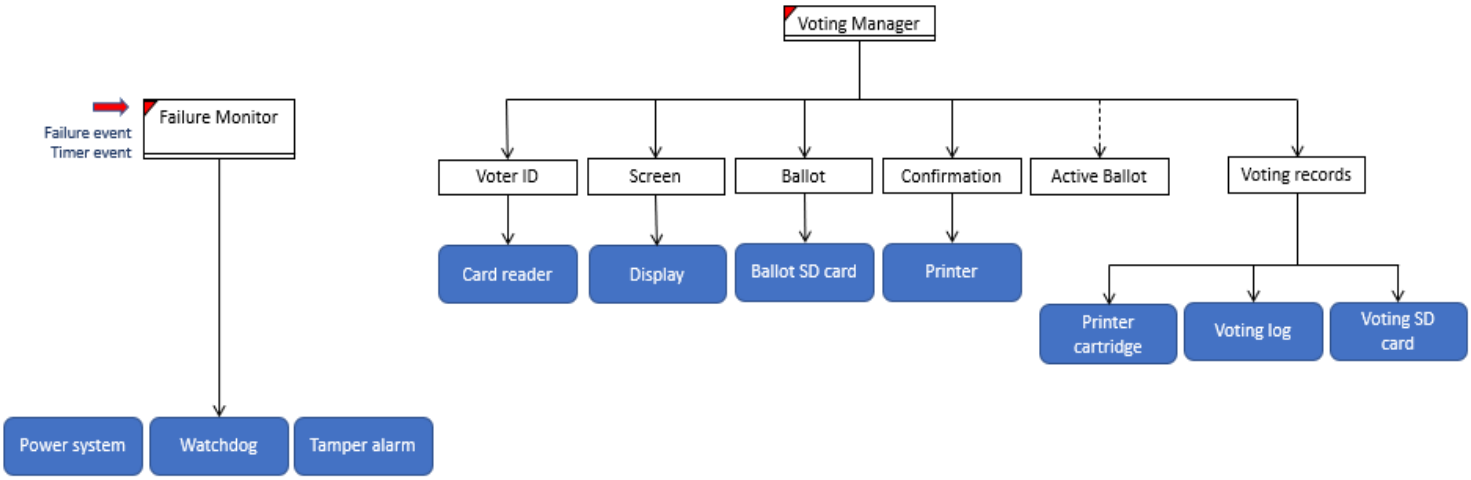


Figure 2 Design Diagram

1. Ballot Class

```java
package Voting_Machine;

import java.util.ArrayList;

public class Ballot {

    private  final ArrayList<Question> questions;


    public Ballot(ArrayList<String> question) { questions = Utils.parseBallot(question); }

    public ArrayList<Question> getQuestions() { return  questions; }
    public Question getQuestion(int i) { return  questions.get(i); }
    public int getLength() { return  questions.size(); }

}
```

Figure 3 Ballot Class Code Sample

2. Voting Manager

```
public class VotingManager extends Application{
    enum Machine_State{
        WAITING,
        VOTING_ACTIVE,
        VOTING_PAUSED,
        VOTING_DISABLED,
        FAILURE

    }
```

Figure 4 Voting Machine States Code Sample

```
    //    Temporary until Voter_ID has functions
    public boolean cardIn = false;
    public int ID = 0; //  0=invalid 1=admin 2=voter

    @Override
    public void start(Stage stage) throws Exception {
//      Starting watch dog thread used to catch failures
        Thread watchdog = new Thread(){
            public void run(){
                try {
                    FailureMonitor failure_monitor =
                            new FailureMonitor();
                } catch (Exception e){
                    e.printStackTrace();
                }
            }
        };
        watchdog.start();
```

Figure 5 Card Inserted, ID assigned and start Code Sample

```
public void cardDrag(MouseEvent mouseEvent){
    Node n = (Node)mouseEvent.getSource();
    n.setTranslateX(n.getTranslateX() + mouseEvent.getX());
    n.setTranslateY(n.getTranslateY() + mouseEvent.getY());

    dropZone.setVisible(true);
}

@FXML
public void cardDrop(MouseEvent mouseEvent){
    dropZone.setVisible(false);
    Node n = (Node)mouseEvent.getSource();
    double x = n.getLayoutX() + n.getTranslateX();
    double y = n.getLayoutY() + n.getTranslateY();

    if ( 890 > x && x > 610 && 679 > y && y > 573 && !cardIn){
        n.setVisible(false);
        insertedCard.setVisible(true);
        cardIn = true;
        if(Objects.equals(n.getId(), "adminCard")){
            ID = 1;
            header.setText("Choose an option. Current machine state is: "+
                    state.toString());
            option_txt_1.setText("Check Ballot");
            option_txt_2.setText("Start Voting");
            option_txt_3.setText("Pause Voting");
            option_txt_4.setText("End Voting");
        } else if (Objects.equals(n.getId(), "voterCard")){
            steps = ballot.getLength()-1;
            ID = 2;
            option_txt_1.setFill(Color.BLACK);
            option_txt_2.setFill(Color.BLACK);
            option_txt_3.setFill(Color.BLACK);
            option_txt_4.setFill(Color.BLACK);
            option_txt_5.setFill(Color.BLACK);
            option_txt_6.setFill(Color.BLACK);
            if (state != Machine_State.VOTING_ACTIVE){
                header.setText("Voting is inactive right now. Please alert a " +
                        "voting administrator to the machine's condition " +
                        "and move to a new machine. Thank you.");
            } else{
                header.setText("Push Ok Button to star your voting!");
            }

        } else if (Objects.equals(n.getId(), "fakeCard")){
            ID = 0;
        }
    }
```

Figure 6 Inserted Card Check and Management Code Sample

```
public void takeCardOut(MouseEvent mouseEvent){
    cardIn = false;
    insertedCard.setVisible(false);
    header.setText("");
    print_out.setText("");
    option_txt_1.setText("");
    option_txt_2.setText("");
    option_txt_3.setText("");
    option_txt_4.setText("");
    option_txt_5.setText("");
    option_txt_6.setText("");

    if (ID == 1){
        adminCard.setVisible(true);
    } else if (ID == 2){
        voterCard.setVisible(true);
    } else if (ID == 0){
        fakeCard.setVisible(true);
    }
}
```

Figure 7 Card Out state code Sample

```
public void buttonCl
    Node n = (Node)m
    ArrayList<String> rawBallot = null;
    //add V1 and O1
    this.ballotPathPrefix = Paths.get("ballots");
    try {
        rawBallot = Utils.readFile(this.ballotPathPrefix.resolve(Paths.get("V1.txt")).toString());
    } catch (FileNotFoundException e) {

    }
    this.ballot = new Ballot(rawBallot);
```

Figure 8 Register Choice/Click Code Sample

```
case "okay" -> {
    //change state
    if (ID == 1) {
        if (option_txt_1.getFill() == Color.RED) {
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Check Ballot!!!");
        } else if (option_txt_2.getFill() == Color.RED) {
            state = Machine_State.VOTING_ACTIVE;
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Start Voting!!!");
        } else if (option_txt_3.getFill() == Color.RED) {
            state = Machine_State.VOTING_PAUSED;
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Pause Voting!!!");
        } else if (option_txt_4.getFill() == Color.RED)  {
            state = Machine_State.VOTING_DISABLED;
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "End Voting!!!");
        } else {
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Please choose an option!!!");
        }
```

Figure 9 Machine State Code Sample

```
            }
            System.out.println(printOut);
        } else if(ID == 2 && state != Machine_State.VOTING_ACTIVE) {
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Ask Admin for help!");
        } else {
            header.setText("Choose an option. Current machine state is: "+
                    state.toString() + "\n" + "Please Insert Card!");
        }
        steps--;
    }

    case "tamper" -> {
        System.out.println("tamper!");
        state = Machine_State.FAILURE;
        header.setText("Choose an option. Current machine state is: "+
                state.toString() + "\n" + "Ask Admin for help!");
    }
```

Figure 10 UI Output Code Sample

3. Voter ID

```java
package Voting_Machine;

public class VoterID {
    int currentID;
    boolean isCurrentValid;
    boolean isNewVoter;

    public VoterID(){

    }
    boolean validVoter(){

        return false;
    }

    int getId(){

        return 0;
    }

    boolean newVoter(){

        return false;
    }

    int endVote(){

        return 0;
    }

}
```

Figure 11 Voter ID Code Sample