Dr. Mahalingam College of Engineering and Technology, Pollachi-642003

Department of Information Technology

Academic Year: 2024 - 2025

TRL and SDG Certificate

Project Title : Game of the Year – Voting System

Course Code & Name : 23ITI402 Database Management Systems

Department and Semester: Information Technology & IV SEM

Technology Readiness Level (TRL) of the Project	**: _____**
Sustainability Development Goals (SDG)-Goal Name	**: _____**

S.No.	Names and Roll Numbers of Student	Signature of the Student

Preparation (25 marks)	
Interpretation (30 marks)	
Output & Result (15 marks)	
Viva (5 marks)	
Total (75 marks)	
Signature of the faculty incharge	

Objective of the Project

1.1 Purpose and Vision

The primary objective of the *Game of the Year 2025 Voting System* is to provide a user-friendly and interactive platform where users can vote for their favorite video games. As gaming continues to grow in popularity and cultural relevance, recognizing outstanding titles becomes increasingly important. This system not only encourages active participation from users but also ensures transparency in tallying votes and determining the most popular game of the year. The goal is to celebrate excellence in game design, storytelling, and production by letting users express their support directly.

1.2 Digital Voting Experience

The system is built using Java Swing for the GUI and MySQL as the backend database. The application allows users to:

- **View a list of nominated games along with their production companies and vote counts.**
- **Vote for their favourite games in real-time with instant updates.**
- **Add new games to the list with valid production details.**
- **Reset all votes or delete specific entries as needed.**

This ensures an engaging digital experience that mimics real-time leaderboard behaviour while keeping the UI simple and aesthetically appealing.

1.3 Reliability and Extensibility

This project is designed with reliability and future scalability in mind. Through JDBC-based database connectivity, the voting records are stored securely and can be easily maintained. The use of a structured table model makes the system extensible, allowing for future enhancements such as login authentication, category-based voting, and detailed game analytics. By ensuring clear data flow between the frontend and backend, the system promotes accurate vote tracking and easy data management, which is crucial for a fair and trustworthy voting process.

2.Entire Project Code

```
 import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class App {
   static Connection conn;

   public static void main(String[] args) {
      connectDB();
      SwingUtilities.invokeLater(App::createUI);
   }

   static void connectDB() {
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         conn = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/dbms",
               "root",
               "Sukil@27*06");
      } catch (Exception e) {
         JOptionPane.showMessageDialog(null, "Database connection failed:
" + e.getMessage());
```

```
            System.exit(1);
        }
    }

    static void createUI() {
        JFrame frame = new JFrame("🎮 Game of the Year 2025");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(900, 600);
        frame.setLocationRelativeTo(null);

        // Load background image
                                ImageIcon backgroundIcon = new
ImageIcon("C:\\Users\\User\\Downloads\\41524.jpg");
        Image backgroundImage = backgroundIcon.getImage();

        JPanel backgroundPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
              g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
            }
        };
        backgroundPanel.setLayout(new BorderLayout());

        DefaultTableModel model = new DefaultTableModel();
```

```
        JTable table = new JTable(model);
        model.addColumn("Game");
        model.addColumn("Production");
        model.addColumn("Votes");

        table.setFont(new Font("Segoe UI", Font.PLAIN, 16));
        table.setRowHeight(28);
          table.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD,
16));
        table.getTableHeader().setBackground(new Color(60, 120, 200));
        table.getTableHeader().setForeground(Color.WHITE);

        JScrollPane scrollPane = new JScrollPane(table);
        scrollPane.setPreferredSize(new Dimension(650, 300));

        JPanel tablePanel = new JPanel();
        tablePanel.setOpaque(false);
        tablePanel.setLayout(new FlowLayout(FlowLayout.CENTER, 0, 50));
        tablePanel.add(scrollPane);

        JButton voteButton = new JButton("Vote");
        JButton finalizeButton = new JButton("Finalize");
        JButton addGameButton = new JButton("Add Game");
        JButton resetVotesButton = new JButton("Reset Votes");
        JButton deleteGameButton = new JButton("Delete Game");
```

```
        JButton[] buttons = { voteButton, finalizeButton, addGameButton,
resetVotesButton, deleteGameButton };
        for (JButton btn : buttons) {
            btn.setBackground(new Color(70, 130, 180));
            btn.setForeground(Color.WHITE);
            btn.setFont(new Font("Segoe UI", Font.BOLD, 14));
            btn.setFocusPainted(false);
        }

        JPanel buttonPanel = new JPanel();
        buttonPanel.setOpaque(false);
            buttonPanel.setLayout(new  FlowLayout(FlowLayout.CENTER,  20,
10));
        for (JButton btn : buttons)
            buttonPanel.add(btn);

        backgroundPanel.add(tablePanel, BorderLayout.CENTER);
        backgroundPanel.add(buttonPanel, BorderLayout.SOUTH);

        frame.setContentPane(backgroundPanel);
        frame.setVisible(true);

        loadGames(model);
```

```
        voteButton.addActionListener(e -> {
            int row = table.getSelectedRow();
            if (row != -1) {
                String game = (String) model.getValueAt(row, 0);
                voteForGame(game);
                loadGames(model);
            } else {
                JOptionPane.showMessageDialog(frame, "Please select a game to
vote for.");
            }
        });

        finalizeButton.addActionListener(e -> {
            String winner = getWinner();
            JOptionPane.showMessageDialog(frame, " 🏆 Winner: " + winner +
"!");
        });

        addGameButton.addActionListener(e -> {
            JTextField gameField = new JTextField();
            JTextField prodField = new JTextField();
            JPanel inputPanel = new JPanel(new GridLayout(2, 2));
            inputPanel.add(new JLabel("Game Name:"));
            inputPanel.add(gameField);
            inputPanel.add(new JLabel("Production:"));
```

```
        inputPanel.add(prodField);


        int result = JOptionPane.showConfirmDialog(null, inputPanel,
                                    "Enter new game and production", JOptionPane.OK_CANCEL_OPTION);
        if (result == JOptionPane.OK_OPTION) {
            String game = gameField.getText().trim();
            String prod = prodField.getText().trim();
            if (!game.isEmpty() && !prod.isEmpty()) {
                insertGame(game, prod);
                loadGames(model);
            }
        }
    });


    resetVotesButton.addActionListener(e -> {
      int confirm = JOptionPane.showConfirmDialog(frame, "Are you sure you want to reset all votes?",
                "Confirm Reset", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            resetAllVotes();
            loadGames(model);
        }
    });
```

```
        deleteGameButton.addActionListener(e -> {
            int row = table.getSelectedRow();
            if (row != -1) {
                String game = (String) model.getValueAt(row, 0);
                int confirm = JOptionPane.showConfirmDialog(frame, "Delete game: " + game + "?", "Confirm Delete",
                        JOptionPane.YES_NO_OPTION);
                if (confirm == JOptionPane.YES_OPTION) {
                    deleteGame(game);
                    loadGames(model);
                }
            } else {
                JOptionPane.showMessageDialog(frame, "Please select a game to delete.");
            }
        });
    }

    static void loadGames(DefaultTableModel model) {
        try {
            model.setRowCount(0);
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT name, production, votes FROM games ORDER BY votes DESC");
            while (rs.next()) {
```

```
            model.addRow(new Object[] {
                    rs.getString("name"),
                    rs.getString("production"),
                    rs.getInt("votes")
            });
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

static void voteForGame(String game) {
    try {
        PreparedStatement ps = conn.prepareStatement("UPDATE games
SET votes = votes + 1 WHERE name = ?");
        ps.setString(1, game);
        ps.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

static String getWinner() {
    try {
        Statement stmt = conn.createStatement();
```

```
            ResultSet rs = stmt.executeQuery("SELECT name FROM games ORDER BY votes DESC LIMIT 1");
            if (rs.next())
                return rs.getString("name");
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return "No data";
    }

    static void insertGame(String gameName, String production) {
        try {
            PreparedStatement ps = conn.prepareStatement("INSERT INTO games (name, production) VALUES (?, ?)");
            ps.setString(1, gameName);
            ps.setString(2, production);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(null, "Game added successfully!");
        } catch (SQLIntegrityConstraintViolationException e) {
            JOptionPane.showMessageDialog(null, "Game already exists!");
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Failed to add game: " + e.getMessage());
        }
```

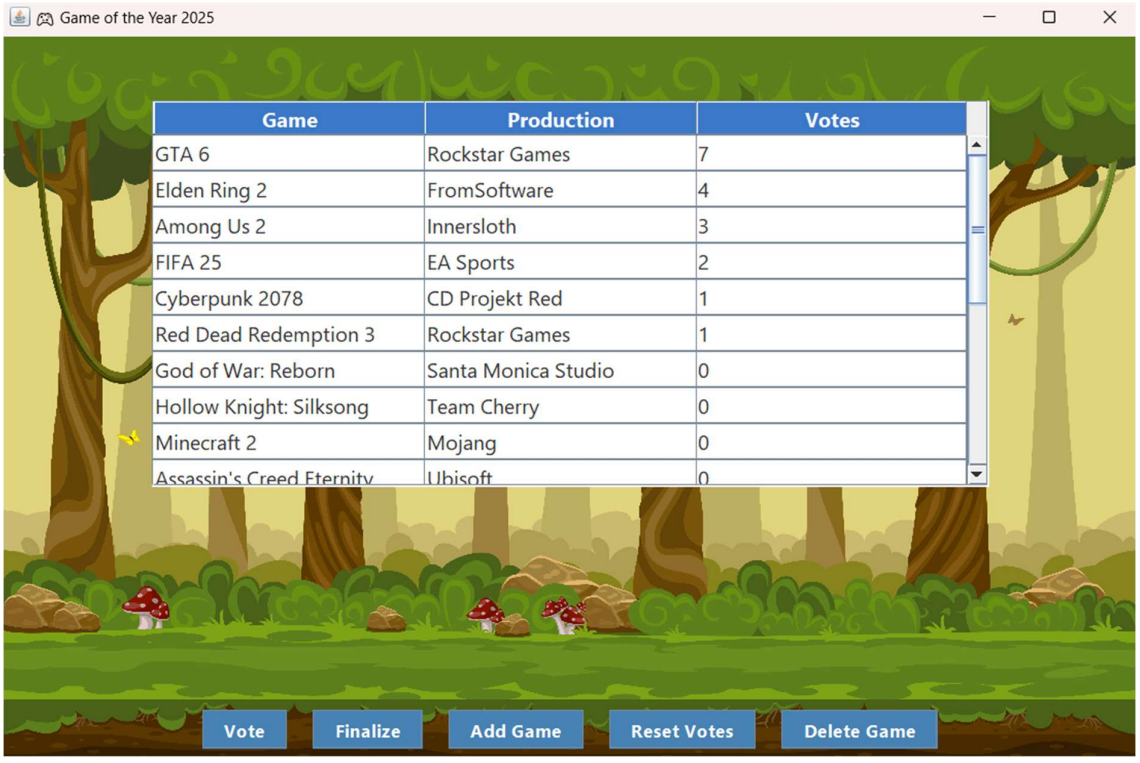
```
    }

    static void resetAllVotes() {
        try {
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("UPDATE games SET votes = 0");
            JOptionPane.showMessageDialog(null, "All votes reset to 0.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    static void deleteGame(String gameName) {
        try {
            PreparedStatement ps = conn.prepareStatement("DELETE FROM games WHERE name = ?");
            ps.setString(1, gameName);
            ps.executeUpdate();
                    JOptionPane.showMessageDialog(null, "Game deleted successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
```

Database view

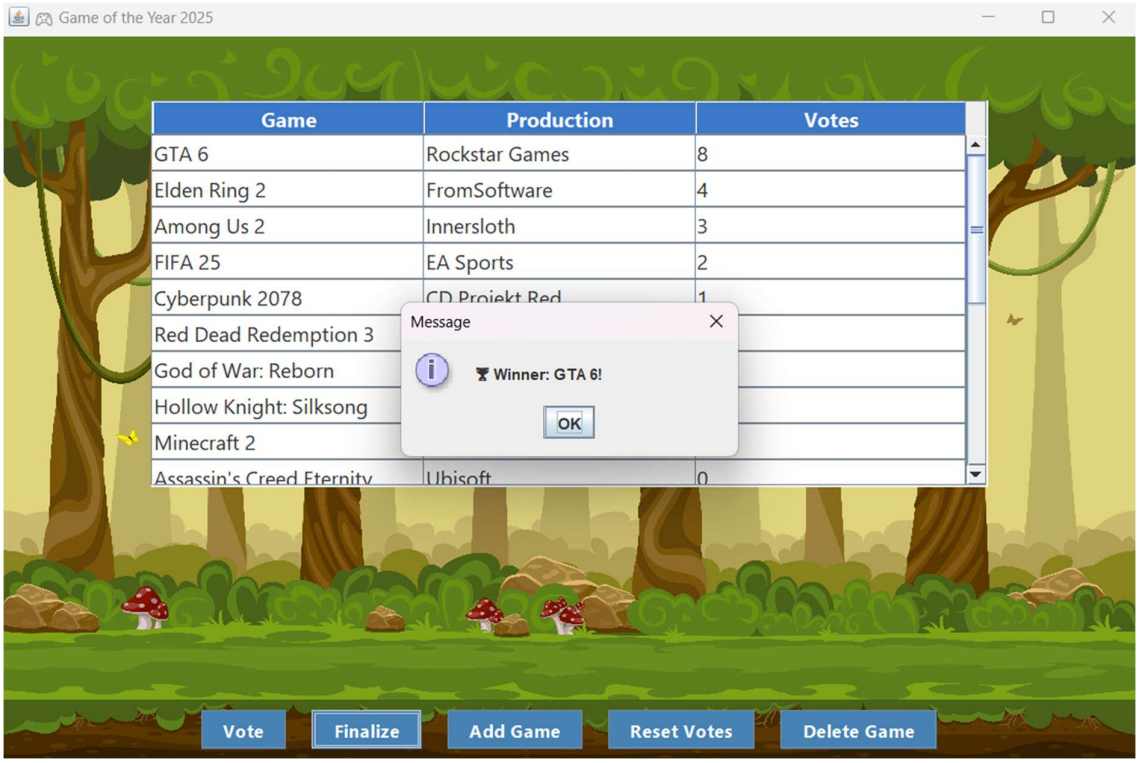
id	name	votes	production
1	Elden Ring 2	4	FromSoftware
2	Cyberpunk 2078	1	CD Projekt Red
3	GTA 6	7	Rockstar Games
4	God of War: Reborn	0	Santa Monica Studio
5	Hollow Knight: Silksong	0	Team Cherry
6	Minecraft 2	0	Mojang
7	Red Dead Redemption 3	1	Rockstar Games
8	Assassin's Creed Eternity	0	Ubisoft
9	Legend of Zelda: Skyfall	0	Nintendo
10	Fortnite Reloaded	0	Epic Games
11	Valorant 2	0	Riot Games
12	Call of Duty: Future Ops	0	Activision
13	League of Legends: Next Gen	0	Riot Games
14	Starfield Online	0	Bethesda
15	Among Us 2	3	Innersloth
16	Apex Legends Prime	0	Respawn Entertainment
17	Overwatch 3	0	Blizzard Entertainment
18	FIFA 25	2	EA Sports
19	The Witcher 4	0	CD Projekt Red
20	Batman: Rise of Shadows	0	Rocksteady Studios

Output

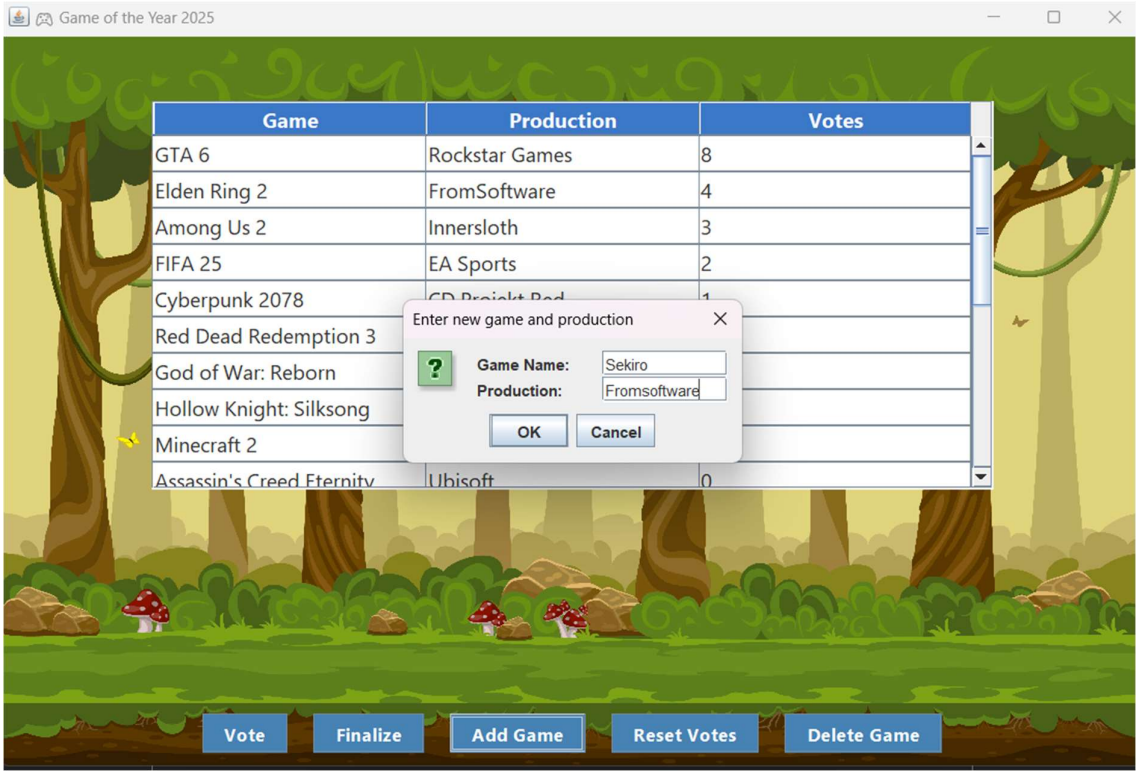
UIOutput



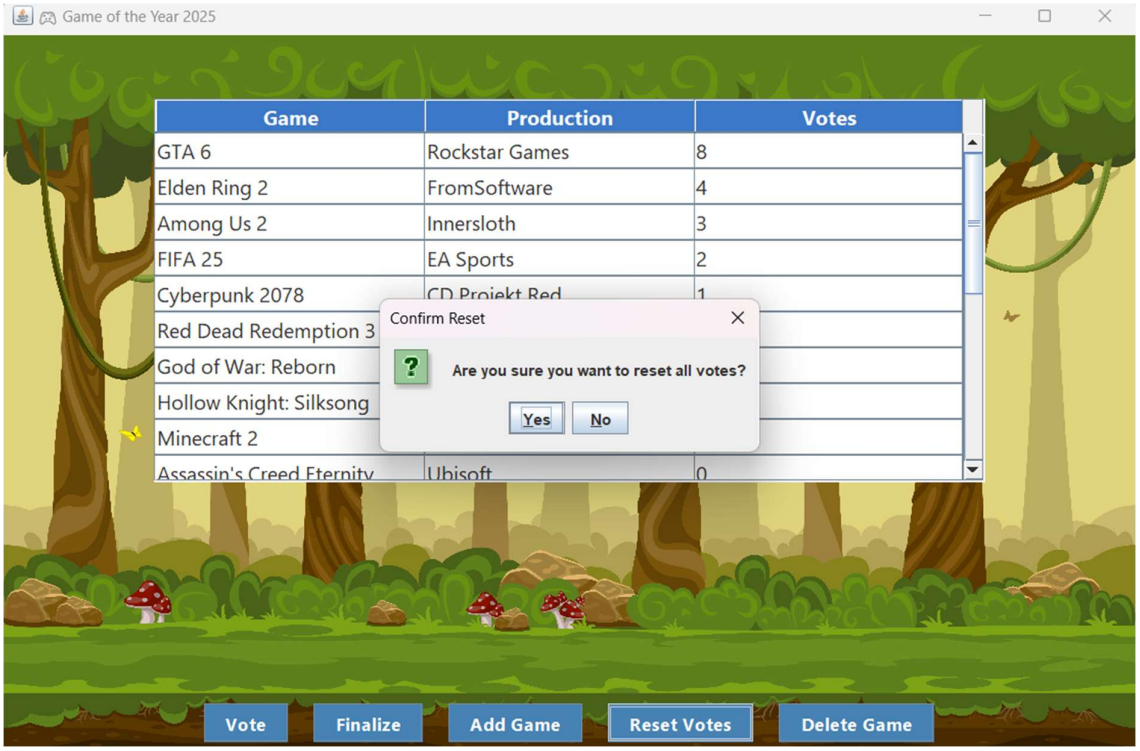
Finalizing Result



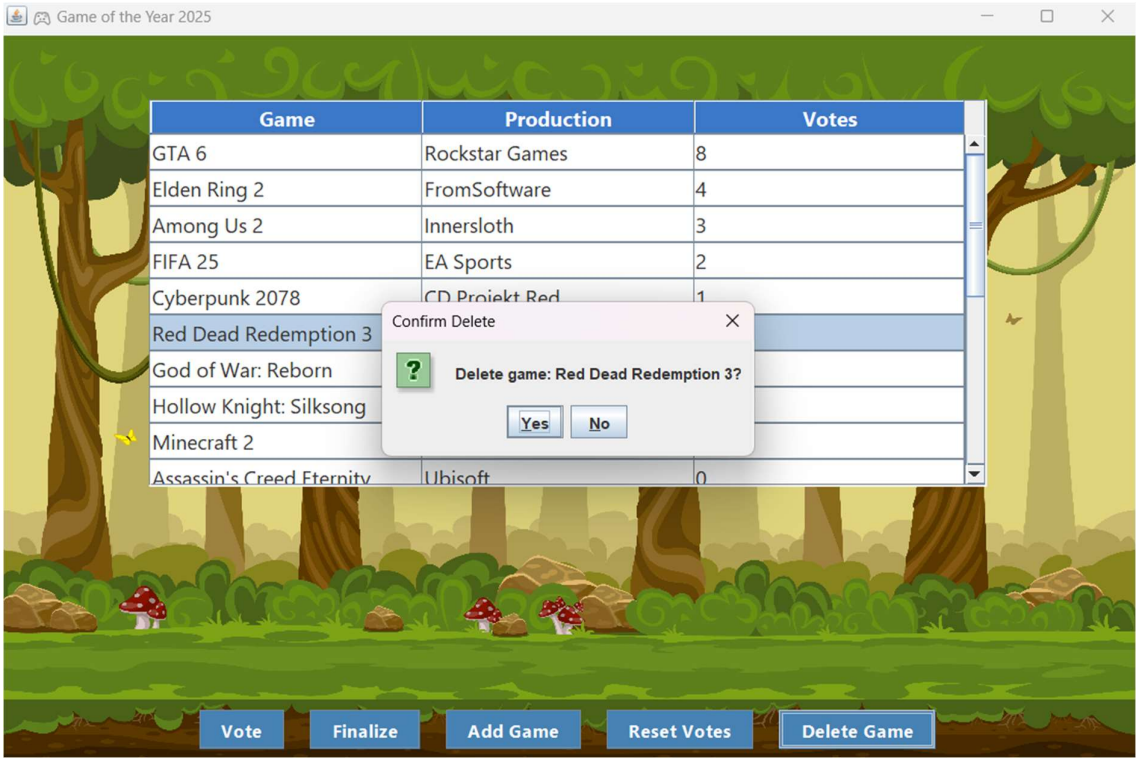
Adding game



Reset Votes



Delete Games



Result of the Project

The *Game of the Year 2025 Voting System* has been successfully implemented using Java Swing for the graphical user interface and MySQL for backend data management. The system meets all the defined objectives, providing a seamless and interactive voting experience for users. Key functionalities such as voting, adding new games, deleting games, resetting votes, and displaying the winner have been tested and verified.