DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

DEPARTMENT OF MASTER COMPUTER APPLICATION

MINI PROJECT-MCA I SEM

2024-25

OBJECT ORIENTED PROGRAMMING USING JAVA

COURSE CODE - MMC104

BY

RAHUL XAVIER - 1DT24MC069

SYED ARFA - 1DT24MC091

SOWMYA RANI M E - 1DT24MC085

TABLE OF CONTENTS

SL.NO	TOPICS	PAGE NO
1	ABSTRACT	3
2	INTRODUCTION	4
3	HARDWARE AND SOFTWARE REQUIREMENTS	5
4	SYSTEM DESIGN	6
5	CODE SNIPPETS	7
6	FORM DESIGN	11
7	TESTING AND IMPLEMENTATION	15
8	CHALLENGES FACED	16
9	FUTURE ENHANCEMENTS	18
10	CONCLUSION	20
11	REFERENCES	21

ABSTRACT

This project report presents the development of a Tic-Tac-Toe game using Java Swing for GUI. The game is designed to provide an interactive experience with features like player turns, win detection, a reset button, and user-friendly UI styling. The objective of this project is to demonstrate the application of Java programming concepts such as event handling, object-oriented programming, and graphical user interfaces.

This report covers the entire development process, including system design, code implementation, user interface design, testing, and potential future improvements.

INTRODUCTION

Tic-Tac-Toe is a popular two-player game played on a 3x3 grid. The goal of the game is for one player to align three of their marks (X or O) either horizontally, vertically, or diagonally before the opponent does. This project is implemented using Java Swing, making it a standalone desktop application. It is designed to be intuitive and visually appealing to users.

Objectives of the Project

- To develop an interactive GUI-based Tic-Tac-Toe game using Java Swing.
- To implement event-driven programming for handling user interactions.
- To provide a simple yet effective demonstration of game logic implementation.
- To ensure smooth gameplay and a user-friendly experience.

Scope of the Project

This project can be extended with additional features like:

- Multiplayer online mode.
- AI opponent using the Minimax algorithm.
- Score tracking system.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- Processor: Intel Core i3 or higher
- RAM: 4GB minimum
- Hard Disk: 100MB free space
- Operating System: Windows/Linux/MacOS

Software Requirements:

- Java Development Kit (JDK) 8 or higher
- Integrated Development Environment (IDE) such as Eclipse or IntelliJ IDEA
- Swing Library for GUI components

SYSTEM DESIGN

Architecture Overview

The Tic-Tac-Toe game follows the Model-View-Controller (MVC) architecture:

1. **Model:** Manages the game logic and state.
2. **View:** Displays the game board and UI components.
3. **Controller:** Handles user inputs and updates the game state.

Components of the System

- **Game Board:** A 3x3 grid implemented using Java Swing buttons.
- **Game Logic:** Methods to check for a win, track player turns, and determine a tie.
- **User Interface:** A GUI developed using JLabel and JPanel to display game status and results.
- **Event Handling:** ActionListeners to manage user interactions.

CODE SNIPPETS

Game Initialization

```
public TicTacToe() {
    frame.setSize(600, 700);
    frame.setLocationRelativeTo(null);
    frame.setResizable(false);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new BorderLayout());

    textLabel.setBackground(Color.darkGray);
    textLabel.setForeground(Color.white);
    textLabel.setFont(new Font("Arial", Font.BOLD, 50));
    textLabel.setHorizontalAlignment(JLabel.CENTER);
    textLabel.setText("Tic-Tac-Toe");
    textLabel.setOpaque(true);
    textPanel.setLayout(new BorderLayout());
    textPanel.add(textLabel);
    frame.add(textPanel, BorderLayout.NORTH);
}
```

Checking Winner Logic

```
 void checkWinner() {
       for (int r = 0; r < 3; r++) {
           if (!board[r][0].getText().equals("") &&
board[r][0].getText().equals(board[r][1].getText()) &&
board[r][1].getText().equals(board[r][2].getText())) {
               highlightWinner(new JButton[]{board[r][0],
board[r][1], board[r][2]});
               return;
           }
       }

       for (int c = 0; c < 3; c++) {
           if (!board[0][c].getText().equals("") &&
board[0][c].getText().equals(board[1][c].getText()) &&
board[1][c].getText().equals(board[2][c].getText())) {
               highlightWinner(new JButton[]{board[0][c],
board[1][c], board[2][c]});
               return;
           }
       }
```

```
        if (!board[0][0].getText().equals("") &&
board[0][0].getText().equals(board[1][1].getText()) &&
board[1][1].getText().equals(board[2][2].getText())) {
            highlightWinner(new JButton[]{board[0][0],
board[1][1], board[2][2]});
            return;
        }

        if (!board[0][2].getText().equals("") &&
board[0][2].getText().equals(board[1][1].getText()) &&
board[1][1].getText().equals(board[2][0].getText())) {
            highlightWinner(new JButton[]{board[0][2],
board[1][1], board[2][0]});
            return;
        }

        if (turns == 9) {
            textLabel.setText("It's a tie!");
            gameOver = true;
        }
    }
```

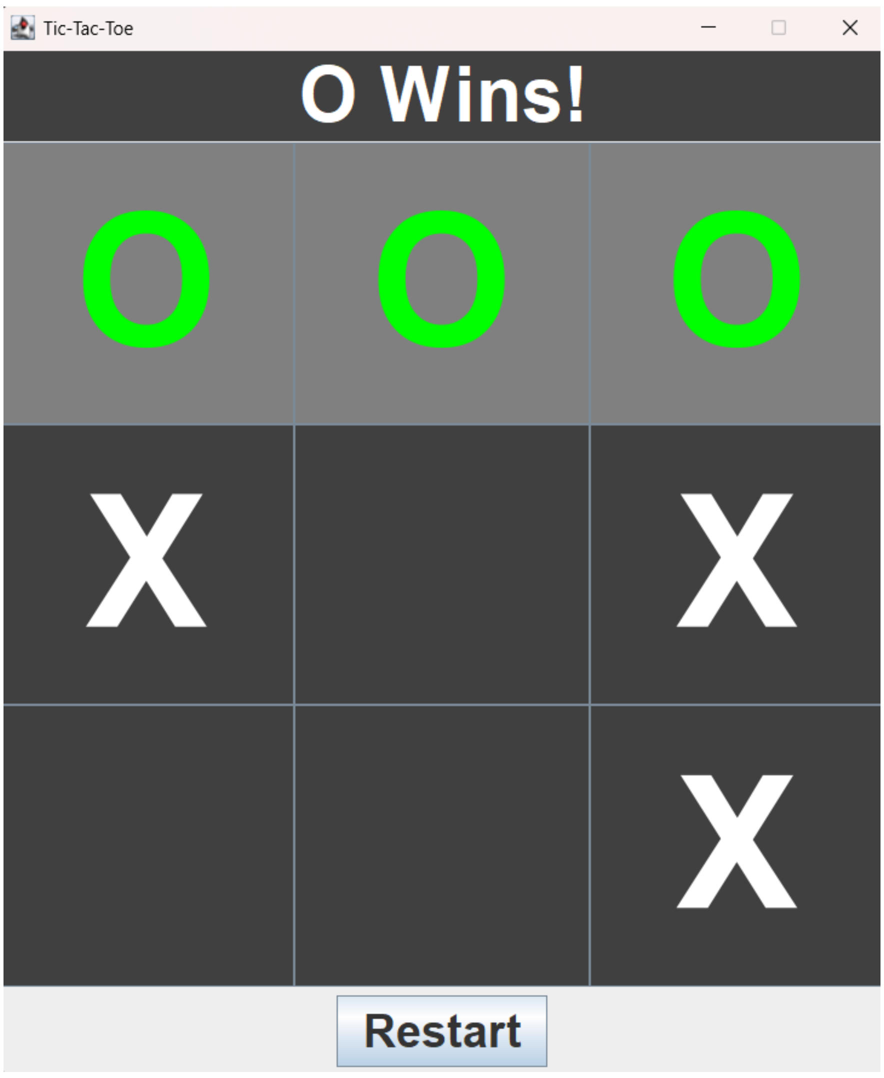
```
void highlightWinner(JButton[] tiles) {
    for (JButton tile : tiles) {
        tile.setForeground(Color.green);
        tile.setBackground(Color.gray);
    }
    textLabel.setText(currentPlayer + " Wins!");
    gameOver = true;
}

void resetGame() {
    for (int r = 0; r < 3; r++) {
        for (int c = 0; c < 3; c++) {
            board[r][c].setText("");
            board[r][c].setForeground(Color.white);
            board[r][c].setBackground(Color.darkGray);
        }
    }
    currentPlayer = playerX;
    textLabel.setText("Tic-Tac-Toe");
    gameOver = false;
    turns = 0 }
```

FORM DESIGN

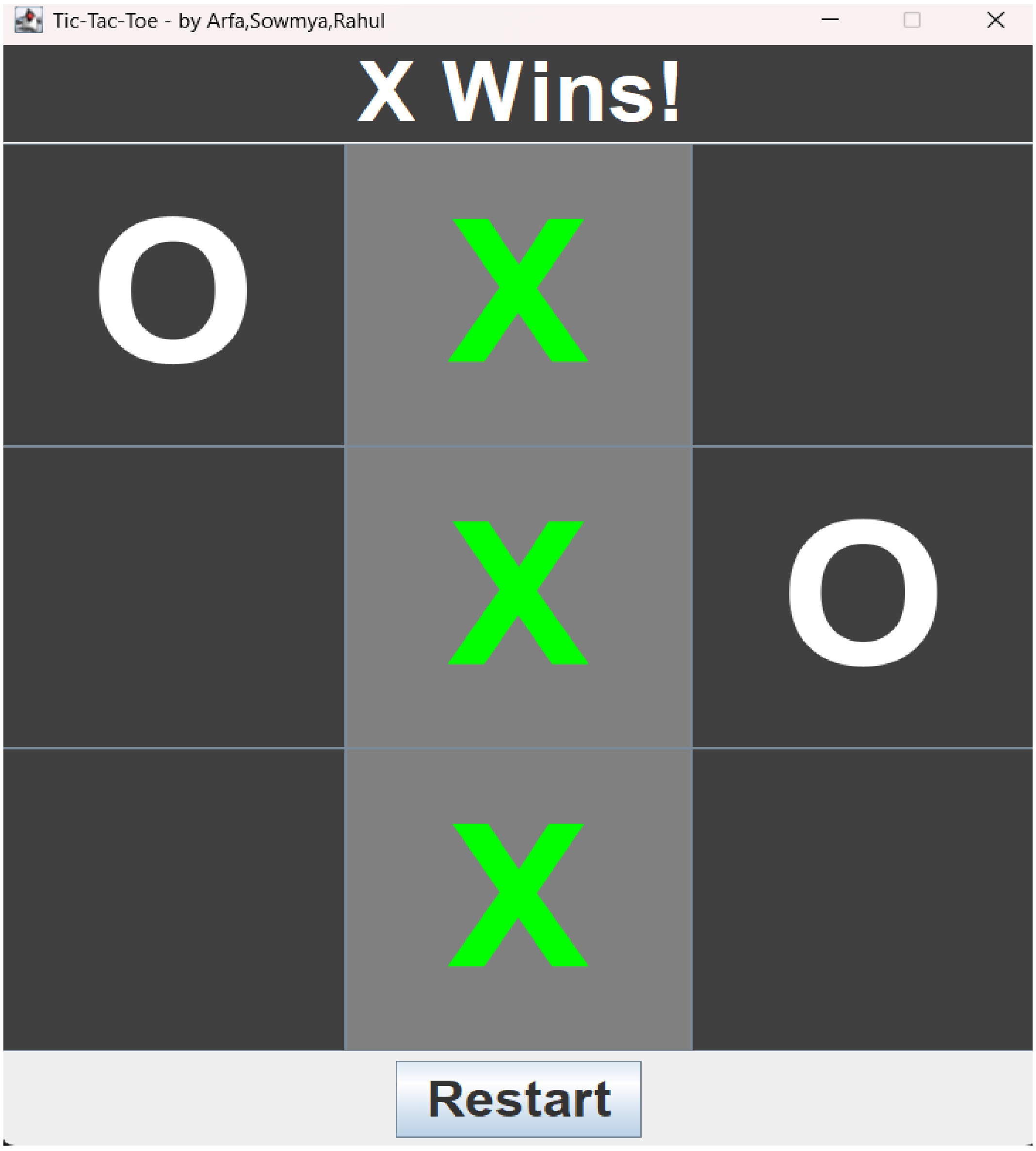
The user interface consists of:

- A title label displaying "Tic-Tac-Toe".
- A 3x3 grid for gameplay.
- A reset button to restart the game.

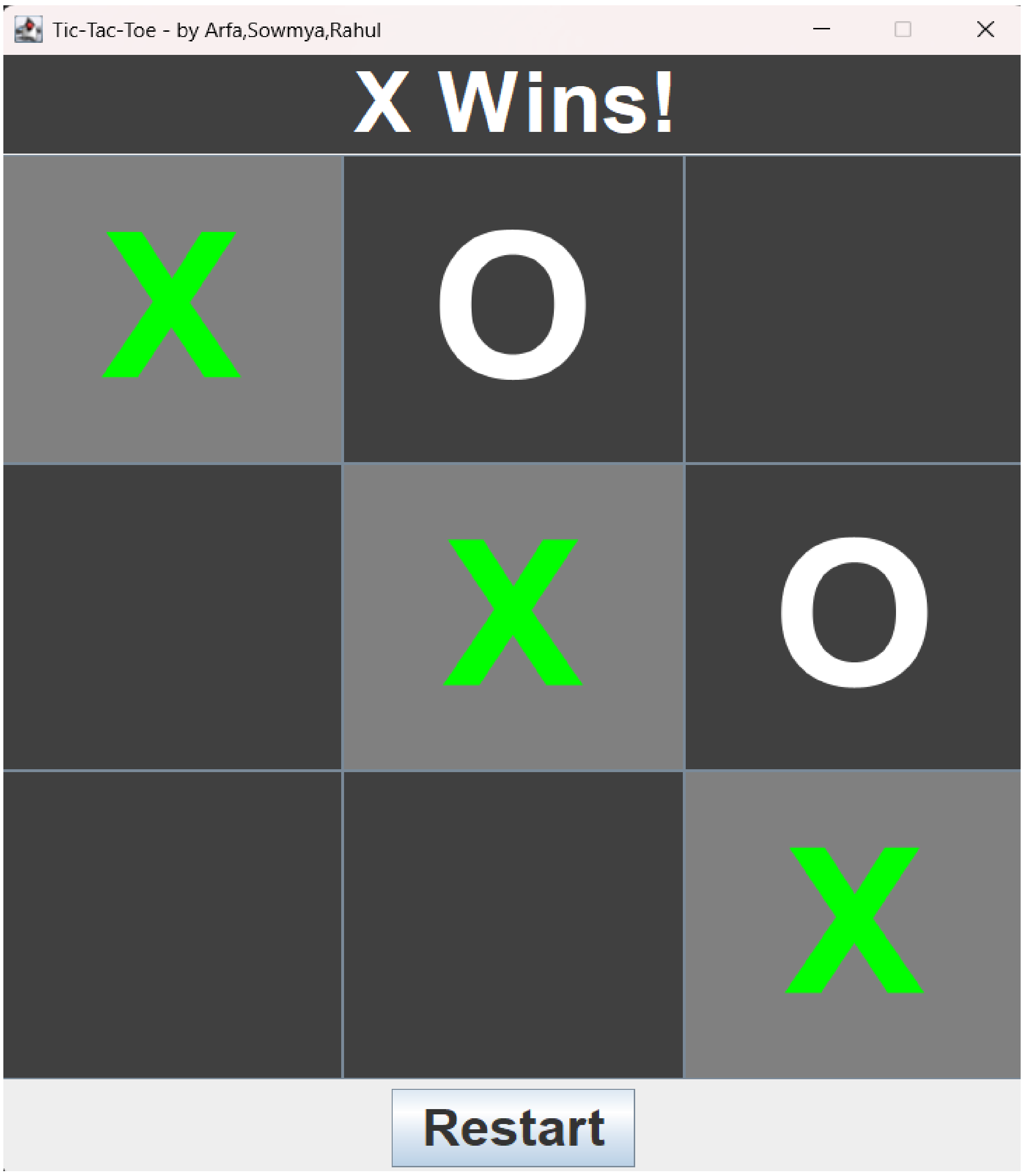
ROW WISE WIN



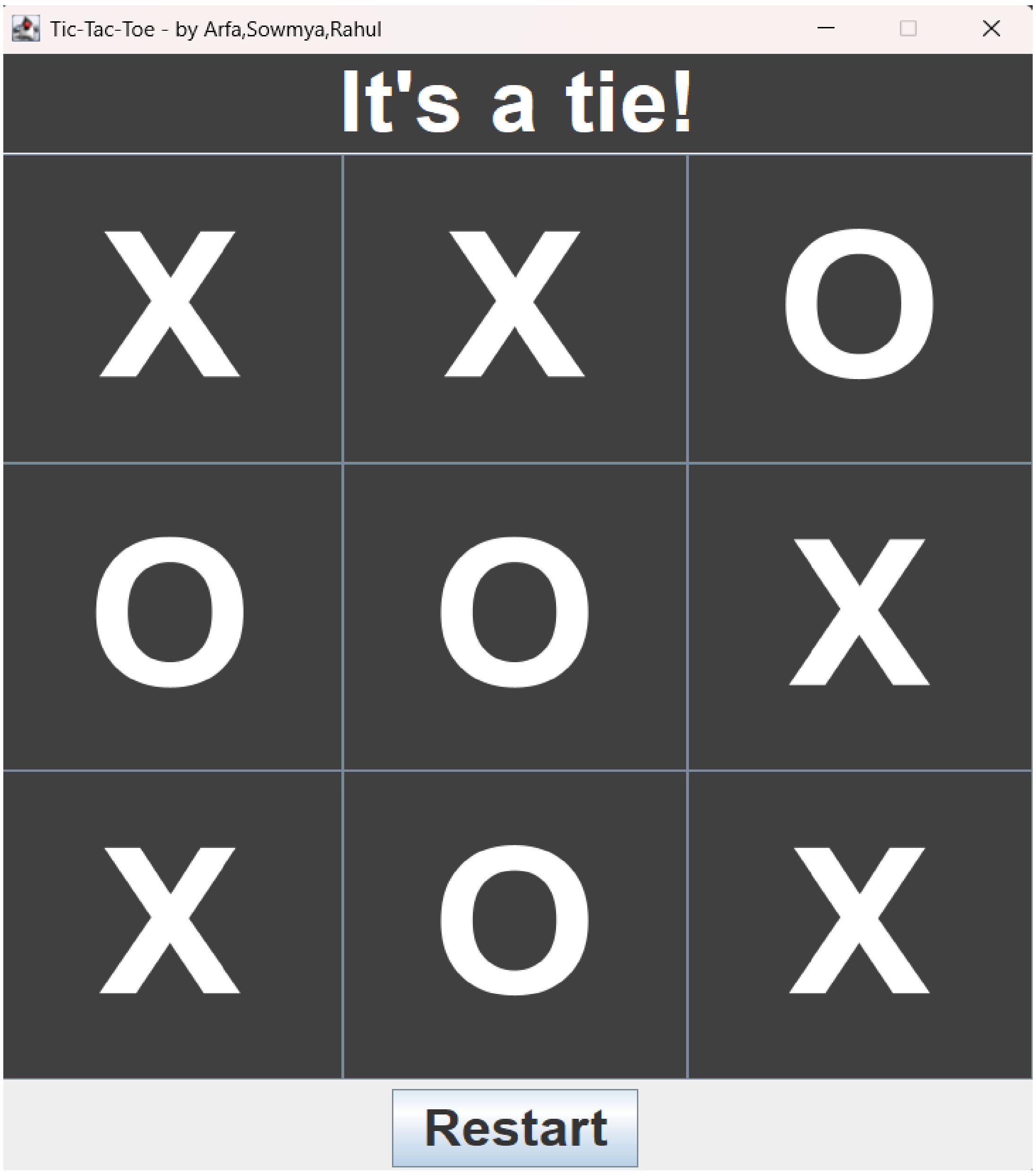
CLOUMN WISE WIN



CROSS COLUMN WIN



MATCH TIE


Tic-Tac-Toe - by Arfa,Sowmya,Rahul

It's a tie!

X	X	O
O	O	X
X	O	X

TESTING AND IMPLEMENTATION

Test Cases

Test Case	**Input**	**Expected Output**	**Status**
Player X clicks on (0,0)	X	Mark is placed	Pass
Player O clicks on (1,1)	O	Mark is placed	Pass
Player X wins	X aligns 3 in a row	"X Wins!" displayed	Pass
Board fills with no winner	Full board	"It's a tie!" displayed	Pass

CHALLENGES FACED

During the development of the Tic-Tac-Toe game using Java Swing, several challenges were encountered. These challenges spanned from technical difficulties to UI/UX-related issues. Below are the key challenges and the approaches taken to resolve them:

1. Managing GUI Event Listeners Efficiently

Handling user interactions in Java Swing requires adding event listeners for buttons. A major challenge was ensuring that every button click correctly updated the game state without causing unexpected behavior. Initially, some buttons were not responding as expected due to improper event handling. The issue was resolved by using ActionListener properly and ensuring each button was assigned a dedicated event handler.

2. Detecting Win Conditions Accurately

The game required logic to check if a player had won after each move. Implementing this logic efficiently was a challenge. Initially, the win detection algorithm checked only specific conditions, leading to incorrect results in some cases. To fix this, a more structured approach was taken by iterating through all possible winning conditions and verifying them dynamically.

3. Preventing Multiple Clicks on the Same Button

A common issue in Tic-Tac-Toe games is preventing players from clicking the same button multiple times. In early versions of the project, players could overwrite an existing move, causing inconsistencies in the game state. This was resolved by disabling buttons once they were clicked.

4. Improving UI Responsiveness

Java Swing applications sometimes appear outdated or unresponsive. Initially, the game's UI did not adjust properly on different screen resolutions. The layout was improved by using GridLayout for button arrangement and JPanel for better structuring.

5. Handling Game Restarts Smoothly

The reset functionality had to ensure that the game board was cleared and ready for a new game. Early implementations only cleared the board but did not reset internal game states, leading to incorrect turn tracking. The reset logic was refined to clear both the UI and game variables.

FUTURE ENHANCEMENTS

1. Adding an AI Opponent (Single Player Mode)

Currently, the game only supports two human players. Implementing an AI opponent using the Minimax algorithm would allow users to play against the computer, making the game more engaging.

2. Online Multiplayer Mode

The current version of the game requires both players to use the same system. A future enhancement could involve implementing network-based multiplayer functionality using Java sockets. This would allow players to compete over the internet or a local network.

3. Score Tracking System

A scoring system could be implemented to track wins, losses, and draws for each player. The scores could be stored in a local file or database to persist between game sessions.

4. Enhanced User Interface and Animations

Adding animations when a player wins, using effects for button clicks, and incorporating sound effects would enhance user experience. Implementing themes or custom board designs could also improve visual appeal.

5. Mobile-Friendly Version

The game is currently developed using Java Swing, which is best suited for desktops. Porting the game to JavaFX or Android would make it accessible on mobile devices.

By implementing these future enhancements, the Tic-Tac-Toe game can be made more interactive and widely accessible.

CONCLUSION

The development of this Java-based Tic-Tac-Toe game provided valuable insights into GUI programming, event handling, and game logic implementation. Throughout the project, several challenges were encountered, such as managing UI responsiveness, implementing win detection logic, and ensuring a smooth gaming experience.

The project successfully demonstrates key Java programming concepts, including:

- Object-oriented programming principles.
- Effective use of Java Swing for GUI development.
- Implementation of event-driven programming through ActionListener.
- Handling of user interactions and game state management.

Overall, this project has been a great learning experience and a practical demonstration of Java's capabilities in developing interactive applications. The game has room for further improvements, such as AI integration, online multiplayer, and a more advanced UI.

REFERENCES

- **Java Swing Documentation**: https://docs.oracle.com/javase/tutorial/uiswing/
- **Tic-Tac-Toe Game Rules**: https://en.wikipedia.org/wiki/Tic-tac-toe
- **Java Programming Guide by Oracle**
- **GUI Development with Swing and JavaFX**