# JAVA PROJECT REPORT

**ON**

**<TICTACTOE GAME>**

**SUBMITTED BY :**

1. **G NAGASAIKARTHIK 12110784 (33)**

2. **A HARSHITH 12109254 (04)**

3. **ANUGULA RAHUL 12109266 (51)**

**Submitted to**

**Dr. Om Prakash Yadav (26121)**

**Asst. Professor,**

**School of Computer Science and Engineering**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**

Project TITLE : TICTACTOE GAME

**Abstract** : Tic-tac-toe is a classic two-player game played on a 3x3 grid. Each player takes turns marking a square with their symbol (usually X or O) until one player gets three in a row, either horizontally, vertically, or diagonally. The game is simple to learn, but can be challenging to master, as players must anticipate their opponent's moves and plan their own strategy accordingly. In this abstract, we explore the rules and strategies of tic-tac-toe, and discuss various approaches to playing the game, including optimal play and heuristic-based strategies. We also examine variations of the game, such as larger boards and different win conditions, and consider how these changes affect gameplay. Tic-tac-toe is a timeless game that continues to be enjoyed by players of all ages and skill levels, and serves as an important example of simple but engaging game design.

## OBJECTIVE:

The objective of the Tic Tac Toe project is to create a game that allows two players to take turns marking cells in a 3x3 grid. The first player marks cells with an 'X' and the second player marks cells with an 'O'. The winner is the player who places three of their marks in a row, column, or diagonal. The game continues until one player wins, the board is filled up, or the players decide to end the game. The objective of the game is to get three of your symbols in a row, either horizontally, vertically, or diagonally. This project is a Java implementation of the Tic Tac Toe game using GUI components from the Swing library. We have created a very user friendly graphical user interface so that user can easily run through it without any difficulty. The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically, or diagonally - on a 3 x 3 grid.

The main objective of tic-tac-toe is to place three of your marks (either X or O) in a row, either vertically, horizontally, or diagonally, on a 3x3 grid. Players take turns placing their marks on the board until one player achieves three in a row, or all nine squares are filled without either player achieving three in a row, resulting in a tie. Tic-tac-toe is a popular beginner programming project because it is easy to understand and implement, but still requires logical thinking and decision-making.

.

## Description:

Tic-tac-toe is a classic two-player game where each player takes turns marking a 3x3 grid with their symbol (usually X or O) until one player gets three in a row (horizontally, vertically, or diagonally) or all the spaces are filled. The game is often used to introduce basic game theory concepts and strategies. In programming, tic-tac-toe is a popular game to implement as a beginner project as it can be done with simple logic and basic coding skills.

## Introduction:

The Tic Tac Toe project is implemented in Java using the Swing GUI toolkit. It uses the Model-View-Controller (MVC) architecture to separate the game logic from the user interface. The **TicTacToe** class represents the game model, which includes the state of the game board and the logic for checking for a win or a draw. The user interface is implemented using **JButton** objects arranged in a 3x3 grid. The ActionListener interface is used to handle button clicks and update the game state accordingly. The J**Label** class is used to display messages to the user, such as whose turn it is and who won the game.

## Future Scope:

**Keyboard functions will be added.

**We want to design more complex boards for the game in future

**AI opponent:** One of the most interesting enhancements would be to create an AI opponent for the game. This would require developing a machine learning model that can learn to play Tic Tac Toe and make strategic moves against the human player. This could make the game more challenging and engaging.

**Multiplayer functionality:** Another enhancement would be to add multiplayer functionality, so that two people can play against each other on different devices. This would require developing a networking system to allow players to connect and communicate with each other in real time.

**Improved user interface:** The user interface for the game could be improved to make it more visually appealing and user-friendly. This could include adding animations and sound effects, as well as improving the design of the game board and menu system

**Difficulty levels**: Adding different difficulty levels to the game could make it more interesting for players of different skill levels. For example, the game could have an easy mode where the AI opponent makes random moves, and a hard mode where the AI opponent is more strategic and difficult to beat.

**Score tracking:** Keeping track of scores and displaying them on the screen could add a competitive element to the game, encouraging players to try to beat their high scores and compete against each other.

**Board size customization**: Tic Tac Toe traditionally has a 3x3 board, but allowing the user to choose a different board size, like 4x4 or 5x5, could create more complex gameplay.

## System Requirement (Hardware / Software):

The system requirements for a Tic Tac Toe project can vary depending on the platform and java  programming language being used. However, here are some general hardware and software requirements:

**Hardware Requirements:**

A computer or mobile device capable of running the operating system required by the programming language or platform being used
Sufficient processing power and memory to run the application smoothly
**Software Requirements:**

An integrated development environment (IDE) or code editor that supports the chosen programming language (such as Visual Studio Code for JavaScript or Eclipse for Java)

The necessary libraries and dependencies for the programming language being used (such as React for JavaScript or JavaFX for Java)
A version control system such as Git, to manage and track changes to the codebase
A web server or hosting platform, if the project is being developed for the web
In addition to these requirements, it is recommended to follow best practices for software development, including regular backups and testing, documentation, and adherence to coding standards and guidelines.

## Libraries and Methods Used:

The Tic Tac Toe project uses the following Java libraries and methods:

### Libraries

- **java.awt.*:** This library provides classes for creating and manipulating graphical user interfaces (GUIs).
- **java.awt.event.*:** This library provides classes and interfaces for handling events, such as button clicks, in GUIs.
- **javax.swing.***: This library provides a set of GUI components that are built on top of the AWT toolkit.

### Methods

- **JFrame()**: This is a constructor for the JFrame class that creates a new window for the game board.
- **JButton()**: This is a constructor for the JButton class that creates a new button object.
- **addActionListener**(): This method adds an action listener to a button object to handle button clicks.
- **setLayout()**: This method sets the layout manager for a panel object to determine how its components are arranged.
- **add()**: This method adds a component to a container, such as a panel or a frame.

- **getSource():** This method returns the source of an event, such as the button that was clicked.
- **setText()**: This method sets the text of a button or label object.
- **setBackground()**: This method sets the background color of a button object.
- **setEnabled():** This method enables or disables a button object.
- **equals()**: This method checks if two objects are equal.
- **setSize():** This method sets the size of a window or a component.
- **setVisible():** This method shows or hides a window or a component.
- **public TicTacToe()**: Constructor method for the TicTacToe class. It initializes the game board with nine buttons and sets the layout of the main panel and message label.
- **public void actionPerformed(ActionEvent e)**: This method is called when a button on the game board is clicked. It handles the logic for placing X's and O's on the board, checking for a win or a draw, and updating the message label.
- **private boolean checkForWin()**: This method checks if there is a winner by calling the checkRow() method with all the possible winning combinations.
- **private boolean checkRow(int a, int b, int c)**: This method checks if the three buttons at indices a, b, and c have the same non-empty text. It is called by the checkForWin() method with all possible winning combinations.
- **private boolean checkForDraw()**: This method checks if the game has ended in a draw by checking if all the buttons on the board have been filled with X's or O's.
- **private void resetBoard()**: This method resets the game board by clearing all the button text and background color, enabling all the buttons, setting the turn to X's, and updating the message label.

## GUI :

The GUI of this project consists of a JFrame window with a title of "Tic Tac Toe". The window contains a main panel with a 3x3 grid of JButtons representing the game board. Each button has an ActionListener attached to it that triggers the actionPerformed() method when it is clicked. The main panel is placed at the center of the window using the BorderLayout.CENTER constant.

At the top of the window is a JLabel named "message" that displays the turn of the current player and any game-ending messages such as a win or a draw. The JLabel is placed at the top of the window using the BorderLayout.NORTH constant.

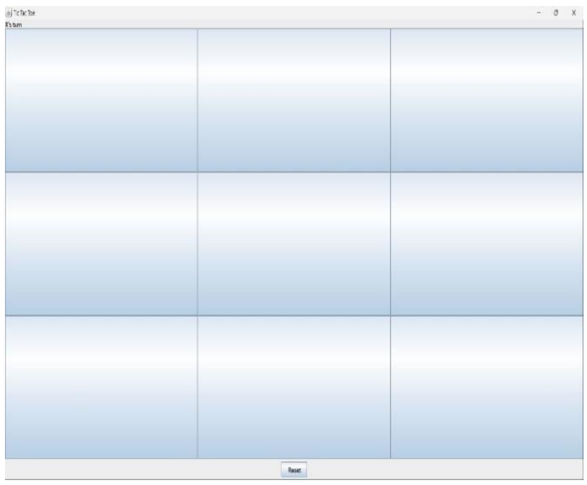
At the bottom of the window is a JPanel named "buttonPanel" that contains a single JButton named "Reset". The JButton has an ActionListener attached to it that triggers the resetBoard() method when it is clicked. The JPanel is placed at the bottom of the window using the BorderLayout.SOUTH constant.

## USE OF GUI:

The Tic Tac Toe project uses the Swing GUI toolkit to create the user interface for the game. The user interface consists of a window with a 3x3 grid of buttons that represent the cells of the game board. The buttons are arranged in a JPanel object with a GridLayout layout manager. Each button has an action listener attached to it to handle button clicks. When a button is clicked, the action listener checks if the button is empty and updates the game state accordingly. If a player wins or the game ends in a draw, the action listener disables all the buttons and displays a message to the user.

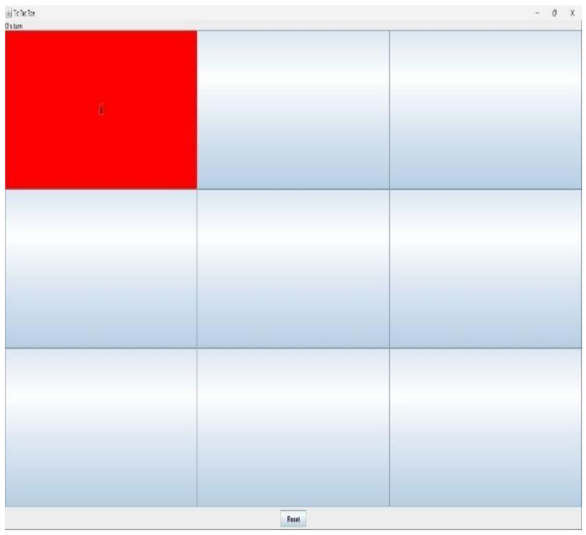
## STEPS OF GUI INTERFACE:

Step1:



- After compiling the code this is the interface that'll appear as Output

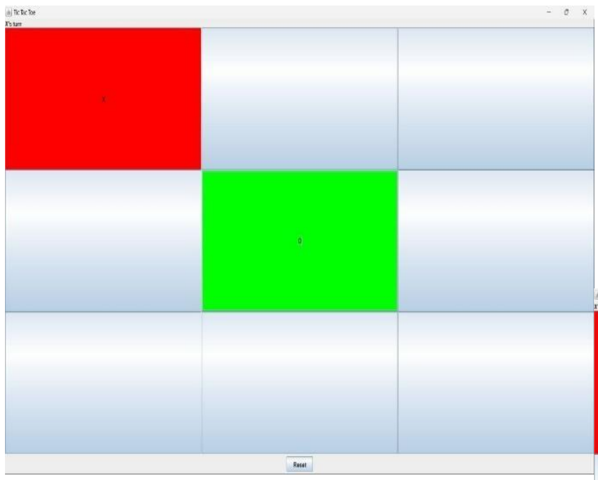
Step2:



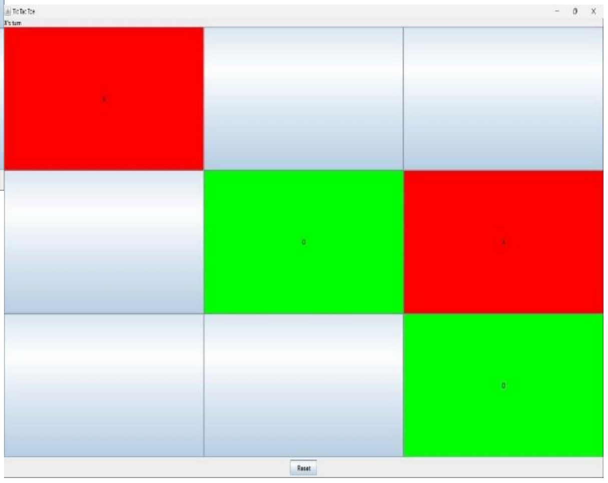
- The first player has to select a cell in output 3*3 grid with an 'x'

Step3:

**Then the second has to select a cell with an 'o'

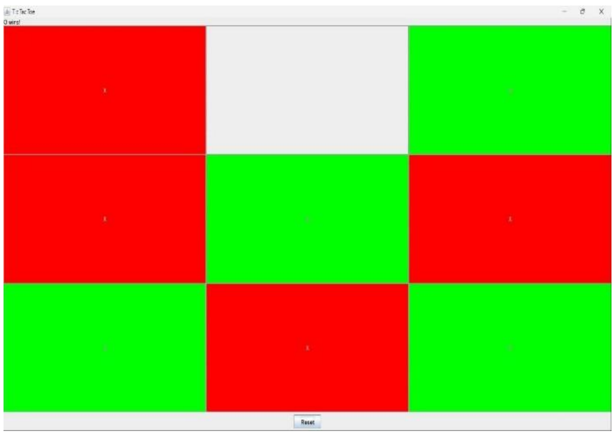


Step4:



- The game continues until one player wins, the board fills up or the players decide to end the game

Step5:


Tic Tac Toe

O wins!

In case if any player won or the game is draw then they can start a new game by clicking on ‘reset’ button

**APPENDIX:**

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

public class TicTacToe implements ActionListener {

   private JFrame frame = new JFrame("Tic Tac Toe");
private JButton[] buttons = new JButton[9];     private
JLabel message = new JLabel("X's turn");     private
boolean xTurn = true;

   public TicTacToe() {       JPanel panel =
new JPanel();       panel.setLayout(new
GridLayout(3, 3));

      for (int i = 0; i < buttons.length; i++) {
buttons[i] = new JButton();
```

```
buttons[i].addActionListener(this);
panel.add(buttons[i]);
    }

      JPanel buttonPanel = new JPanel();     JButton
resetButton = new JButton("Reset");
      resetButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  resetBoard();
            }
      });
      buttonPanel.add(resetButton);

    frame.add(panel, BorderLayout.CENTER);
      frame.add(buttonPanel, BorderLayout.SOUTH);
frame.add(message, BorderLayout.NORTH);
frame.setSize(300, 300);
    frame.setVisible(true);
  }

  public void actionPerformed(ActionEvent e) {
    JButton button = (JButton)e.getSource();

    if (button.getText().equals("")) {
if (xTurn) {              button.setText("X");
button.setBackground(Color.RED);
message.setText("O's turn");
```

```
            } else {                button.setText("O");
button.setBackground(Color.GREEN);
message.setText("X's turn");
            }

            if (checkForWin()) {
                message.setText(button.getText() + " wins!");
for (JButton b : buttons) {
b.setEnabled(false);
                }
            } else if (checkForDraw()) {
message.setText("It's a draw!");
            } else {
xTurn = !xTurn;
            }
        }
    }

    private boolean checkForWin() {
        return checkRow(0, 1, 2) || checkRow(3, 4, 5) || checkRow(6, 7, 8) ||
checkRow(0, 3, 6) || checkRow(1, 4, 7) || checkRow(2, 5, 8) ||              checkRow(0,
4, 8) || checkRow(2, 4, 6);
    }

    private boolean checkRow(int a, int b, int c) {
        return buttons[a].getText().equals(buttons[b].getText()) &&
buttons[b].getText().equals(buttons[c].getText()) &&
               !buttons[a].getText().equals("");
    }
```

```
   private boolean checkForDraw() {
for (JButton b : buttons) {            if
(b.getText().equals("")) {
return false;
          }
      }

      return true;
   }

 private void resetBoard() {
      for (int i = 0; i < buttons.length; i++) {
buttons[i].setText("");
buttons[i].setBackground(null);
buttons[i].setEnabled(true);
      }
      xTurn = true;
      message.setText("X's turn");
   }

   public static void main(String[] args) {
new TicTacToe();
   }
}
```

## Conclusion:

In conclusion, we successfully implemented the Tic Tac Toe game using Java's Swing GUI toolkit and the MVC design pattern. The game features a 3x3 grid of buttons that players can click on to place either an "X" or an "O". The game ends when a player wins by getting three in a row, either horizontally, vertically, or diagonally, or when the game ends in a draw. The TicTacToe class implements the ActionListener interface to handle button click events and updates the game state accordingly. Overall, this project demonstrates the use of eventdriven programming and graphical user interfaces in Java to create a simple game. It could be expanded upon by adding more features, such as a scoring system or different game modes, to make it more challenging and engaging for the players.

## REFERENCES:

.Books: Fundamentals of Java

Lectures attend : short term application course on java ,NPTEL

THANK YOU 😊