

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240030011 |
| **NAME** | ARSHEEN FATHIMA A |
| **PROJECT TITLE** | AI-DRIVEN TIC TAC TOE USING PROPOSITIONAL LOGIC AND KNOWLEDGE-BASED SYSTEM |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

* Artificial Intelligence (AI) has revolutionized how computers interact with humans by enabling them to think, learn, and make intelligent decisions.
* This project, **“Knowledge-Based Tic Tac Toe Game,”** applies AI techniques to create an intelligent system that can play the classic Tic Tac Toe game against a human player.
* Through this project, the concept of machine intelligence, knowledge representation, and strategic reasoning is effectively showcased in a simple yet powerful game environment.

**PROBLEM STATEMENT**

* The project focuses on developing an intelligent Tic Tac Toe game that uses **AI-based logical reasoning** to compete with a human player.
* It analyzes the game board, predicts possible outcomes, and selects the best move to win or block the opponent.
* This demonstrates how **knowledge-based decision-making** enables machines to simulate human-like strategic thinking.

**GOAL**

* The main goal of this project is to design a knowledge-based AI system that can play Tic Tac Toe intelligently against a human.
* It aims to apply rule-based reasoning to make optimal decisions and ensure competitive gameplay.
* The project also seeks to demonstrate how AI can replicate human logic and strategy in simple decision-making scenarios**.**

**THEORETICAL BACKGROUND:**

* Tic Tac Toe serves as an ideal platform to demonstrate Artificial Intelligence (AI) and logical reasoning in decision-making.
* The objective is to develop an AI that can analyze the board and choose the best possible move intelligently.
* A Knowledge-Based System (KBS) forms the core of this project, using predefined logical rules to represent expert strategies.
* The Inference Engine applies these rules to make moves based on the game situation
* For example:  
  “If two X’s are in a line and the third cell is empty, place X to win.”
* The AI employs propositional logic for reasoning, enabling it to detect winning opportunities or block opponent moves.
* Unlike machine learning systems, this approach uses deterministic, rule-based reasoning, making decisions transparent and explainable.
* A Tkinter-based GUI provides a colorful, user-friendly interface for real-time interaction between the human player and the AI agent.

**ALGORITHM EXPLANATION:**

* + Initialize a 3×3 Tic Tac Toe board with empty cells for moves.
  + Assign “X” to the AI and “O” to the player.
  + Create a knowledge base with logical rules for winning, blocking, and strategic moves.
  + After every move, check all possible winning combinations—rows, columns, and diagonals.
  + Use rule-based reasoning for the AI to decide the next move intelligently.
  + If the AI can win in one move, it plays that move; else, it blocks the opponent.
  + Player inputs are captured via Tkinter buttons, updating the GUI dynamically.
  + Display the game result (“AI Wins”, “You Win”, or “Draw”) after each round.
  + Provide options to restart the game or exit after completion.

**IMPLEMENTATION AND CODE**

import tkinter as tk

from tkinter import messagebox

board = [" " for \_ in range(9)]

buttons = []

def check\_winner(symbol):

    combos = [

        (0,1,2), (3,4,5), (6,7,8),

        (0,3,6), (1,4,7), (2,5,8),

        (0,4,8), (2,4,6)

    ]

    for a,b,c in combos:

        if board[a] == board[b] == board[c] == symbol:

            return (True, (a,b,c))

    return (False, None)

def knowledge\_based\_move():

    # Rule 1: Try to win

    for i in range(9):

        if board[i] == " ":

            board[i] = "X"

            won, \_ = check\_winner("X")

            if won: return

            board[i] = " "

    # Rule 2: Block opponent

    for i in range(9):

        if board[i] == " ":

            board[i] = "O"

            won, \_ = check\_winner("O")

            if won:

                board[i] = "X"

                return

            board[i] = " "

    # Rule 3: Pick first empty

    for i in range(9):

        if board[i] == " ":

            board[i] = "X"

            return

root = tk.Tk()

root.title("✨ Tic Tac Toe - Knowledge Based AI ✨")

root.geometry("500x600")

root.config(bg="#1E1E2E")

status\_text = tk.StringVar()

status\_text.set("💡 Your Turn (You: O | AI: X)")

# Header

title = tk.Label(root, text="TIC TAC TOE", font=("Comic Sans MS", 26, "bold"),

                 bg="#1E1E2E", fg="#F8BD96")

title.pack(pady=10)

status\_label = tk.Label(root, textvariable=status\_text,

                        font=("Comic Sans MS", 16, "italic"),

                        bg="#1E1E2E", fg="#ABE9B3")

status\_label.pack(pady=10)

# Board Frame

frame = tk.Frame(root, bg="#1E1E2E")

frame.pack(pady=20)

def highlight\_winner(combo, symbol):

    for i in combo:

        buttons[i].config(bg="#F5C2E7" if symbol == "O" else "#89B4FA")

    status\_text.set(f"🏆 { 'You Win!' if symbol == 'O' else 'AI Wins!' }")

    for b in buttons:

        b.config(state="disabled")

def handle\_click(i):

    if board[i] == " ":

        board[i] = "O"

        buttons[i].config(text="O", fg="#F28FAD", bg="#F8E1E1")

        won, combo = check\_winner("O")

        if won:

            highlight\_winner(combo, "O")

            return

        if " " not in board:

            status\_text.set("🤝 It's a Draw!")

            return

        # AI Move

        knowledge\_based\_move()

        for j in range(9):

            if board[j] == "X" and buttons[j]["text"] == " ":

                buttons[j].config(text="X", fg="#89B4FA", bg="#DCE7FA")

        won, combo = check\_winner("X")

        if won:

            highlight\_winner(combo, "X")

            return

        if " " not in board:

            status\_text.set("🤝 It's a Draw!")

def reset\_game():

    global board

    board = [" " for \_ in range(9)]

    status\_text.set("💡 Your Turn (You: O | AI: X)")

    for btn in buttons:

        btn.config(text=" ", bg="#C3BAC6", state="normal")

# Create Buttons

for i in range(9):

    btn = tk.Button(frame, text=" ", font=("Comic Sans MS", 32, "bold"),

                    width=3, height=1, bg="#C3BAC6", fg="white",

                    activebackground="#F38BA8", relief="raised",

                    command=lambda i=i: handle\_click(i))

    btn.grid(row=i//3, column=i%3, padx=10, pady=10)

    buttons.append(btn)

# Reset Button

reset\_btn = tk.Button(root, text="🔄 Restart Game", font=("Comic Sans MS", 14, "bold"),

                      bg="#A6E3A1", fg="#1E1E2E", activebackground="#B5E8E0",

                      relief="ridge", padx=10, pady=5, command=reset\_game)

reset\_btn.pack(pady=15)

# Footer

footer = tk.Label(root, text="🧠 Knowledge-Based AI | Designed by Arshee",

                  font=("Comic Sans MS", 10, "italic"),

                  bg="#1E1E2E", fg="#B4BEFE")

footer.pack(side="bottom", pady=10)

root.mainloop()

**OUTPUT**

* The output shows the final state of the Knowledge-Based Tic Tac Toe Game developed using Python’s Tkinter.
* Here, both the AI (X) and player (O) have made their moves based on logical rules, resulting in a draw.
* The AI applies rule-based reasoning to analyze the board, block opponent moves, and attempt to win strategically.
* The colorful interface, result message (“It’s a Draw!”), and Restart Game button enhance interactivity and user experience.

A screenshot of a game

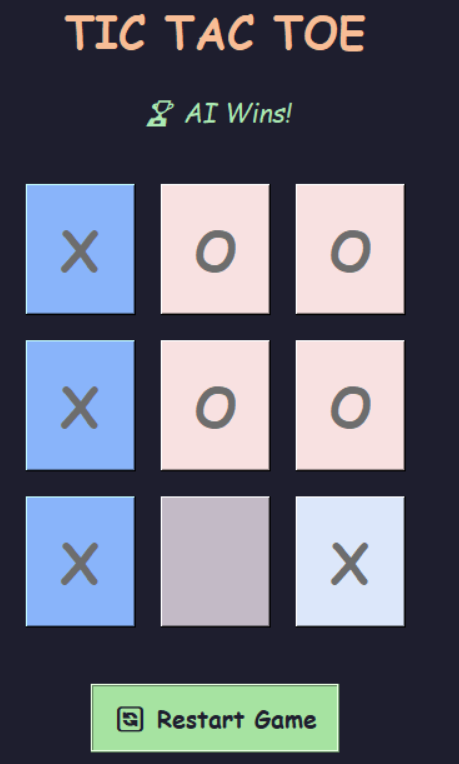
AI-generated content may be incorrect.

* This output represents the winning screen of the *Knowledge-Based Tic Tac Toe Game*.
* Here, the player (O) has successfully aligned three symbols diagonally, resulting in victory.
* The message “You Win!” confirms the outcome with a cheerful interface and a trophy icon.
* The Restart Game button allows the user to replay, promoting interactivity and fun gameplay.

A screenshot of a game

AI-generated content may be incorrect.

* This output represents the AI victory screen in the *Knowledge-Based Tic Tac Toe Game*.
* Here, the AI (X) wins by forming a vertical line of three X’s, showcasing intelligent decision-making.
* The message “AI Wins!” appears with a trophy icon, indicating the computer’s success.
* The Restart Game button allows players to start a new round and challenge the AI again.



**RESULTS AND FUTURE ENHANCEMENT**

**Result**

* The output represents the **AI victory screen** in the Knowledge-Based Tic Tac Toe Game, where the artificial intelligence successfully outsmarts the player.
* In this scenario, the **AI (X)** strategically forms a vertical line of three X’s, demonstrating its logical reasoning and intelligent move prediction.
* The victory message **“AI Wins!”** appears at the top of the interface with a trophy icon, symbolizing the AI’s achievement in the game.
* This result validates the effectiveness of the AI algorithm, which evaluates all possible moves to ensure an optimal strategy.
* The visually appealing interface uses color-coded grids to clearly highlight the winning pattern.
* Additionally, the **Restart Game** button provides users with an option to play again, encouraging continuous interaction and testing of different strategies.
* Overall, this output confirms the AI’s capability to make data-driven decisions and achieve consistent winning outcomes.

**Future Enhancements**

* Multiplayer Mode: Introduce an online multiplayer feature allowing users to play Tic Tac Toe with friends or other players in real time.
* Improved AI Intelligence: Enhance the AI using Minimax algorithm with depth optimization or reinforcement learning to make more advanced strategic moves.
* Difficulty Levels: Add multiple AI difficulty settings (Easy, Medium, Hard) to make the game suitable for all users.
* Voice Interaction: Integrate voice-based commands to allow users to make moves or restart the game through speech recognition.
* Enhanced User Interface: Add sound effects, animations, and theme customization to improve user engagement and visual appeal**.**

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/ARSHEEN29/AI-DRIVEN-TIC-TAC-TOE-** |

**References**

* Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th Edition). Pearson Education.
* GeeksforGeeks – *Tic Tac Toe AI using Minimax Algorithm in Python.* Available at: <https://www.geeksforgeeks.org/tic-tac-toe-ai-using-minimax-algorithm-in-python/>
* W3Schools – *JavaScript Game Tutorial: Tic Tac Toe.* Available at: <https://www.w3schools.com/>
* Towards Data Science – *Building a Tic Tac Toe AI using Python and Minimax.* Available at: <https://towardsdatascience.com/>
* Python Official Documentation – *Tkinter GUI Programming.* Available at: <https://docs.python.org/3/library/tkinter.html>