## horizontal line



Tic-Tac-Toe

28.01.2024

**─**

|  |  |
| --- | --- |
| Team Members | Assigned Roles |
| Abinav Harsha | Project lead |
| Y S Pradyumna | Associate lead |

# Overview

The Tic-Tac-Toe game is a console-based application developed in Python. It provides two modes of play: 1) Against the Program (computer), and 2) Against a Friend. Players can choose to play against the computer or challenge a friend in a classic Tic-Tac-Toe duel. The game incorporates a coin flip mechanism to decide who goes first, adding an element of randomness and fairness.

# Goals

1. Implement a coin flip to determine the starting player, creating a fair and engaging experience.
2. Develop the core game logic for both single-player (against the computer) and two-player modes.
3. Enhance the user experience with clear prompts, messages, and visual representations of the game board.

# Specifications

|  |  |
| --- | --- |
| Team Member | Task Assigned |
| Abinav Harsha | * Designing the code * Adding the feature of coin flip * Documentation |
| Y S Pradyumna | * Designing the code * Testing the code with all possible situations * Modifying the code to make it error free |

**DEADLINE FOR PROJECT COMPLETION:** 24th January, 2024

# Milestones

## Coin Flip Mechanism

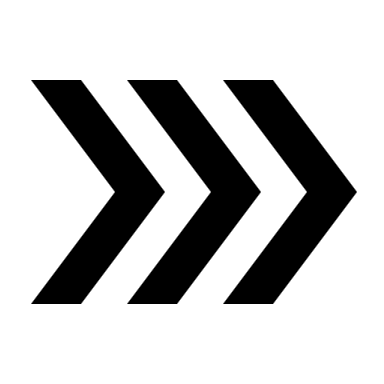
* Implement a coin flip to determine the starting player, providing an element of randomness.

## User Input and Mode Selection

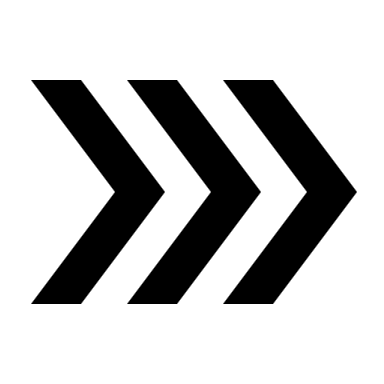
* Allow users to input their names and choose between playing against the computer or a friend.

**Timeline**

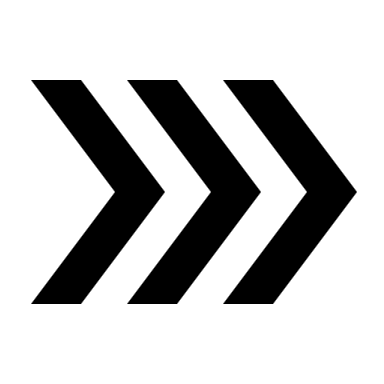
**23nd Jan – Come up with an idea**



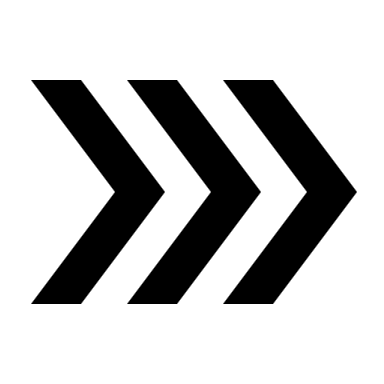
**24th Jan – Develop the core and game logic**



**25th Jan – Implementing single player mode**



**26th Jan –Testing and Refinement**



**30th Jan - Documentation**

**Challenges Faced:**

* Ensuring robust input validation to handle various scenarios and prevent unexpected errors.
* Incorporating the coin flip mechanism seamlessly into the user interface and game flow.

Learning Outcomes:

* **Programming Logic:**
  + Implementing the main game loop, conditional statements, and loops to control the game's flow.
* **User Input Handling:**
  + Handle user input effectively, including input validation to ensure the program can handle a variety of scenarios

# Future Scope:

* + Transform the game into a graphical user interface (GUI) for a more visually appealing and interactive interface.
  + Expand the game to support online multiplayer or local multiplayer on different devices.
  + Implement a scoring system to track and display players' overall performance and achievements.
  + Allow users to customize game settings, such as board size, symbols, and themes.

# Code:

import random  
  
def print\_board(board):  
 for i, row in enumerate(board):  
 print(" | ".join(row))  
 if i < 2:  
 print("-" \* 9)  
  
def check\_winner(board, player):  
 # Check rows, columns, and diagonals for a win  
 for i in range(3):  
 if all(cell == player for cell in board[i]) or all(board[j][i] == player for j in range(3)):  
 return True  
 if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3)):  
 return True  
 return False  
  
def is\_board\_full(board):  
 return all(all(cell != ' ' for cell in row) for row in board)  
  
def player\_turn(board, player):  
 while True:  
 try:  
 position = int(input(f"Enter the position (1-9) for {player}: "))  
 if 1 <= position <= 9:  
 row, col = divmod(position - 1, 3)  
 if board[row][col] == ' ':  
 board[row][col] = player  
 break  
 else:  
 print("Position already taken. Try again.")  
 else:  
 print("Invalid position. Please enter a number between 1 and 9.")  
 except ValueError:  
 print("Invalid input. Please enter a number.")  
  
def computer\_turn(board, player):  
 print(f"{player}'s turn:")  
 while True:  
 # Generate a random position for the computer's move  
 position = random.randint(1, 9)  
 row, col = divmod(position - 1, 3)  
  
 if board[row][col] == ' ':  
 board[row][col] = player  
 print(f"{player} chooses position {position}.")  
 break  
  
def coin\_flip():  
 return random.choice(['Heads', 'Tails'])  
  
def decide\_first\_player(player1, player2):  
 while True:  
 decision = input(f"{player1}, choose 'Heads' or 'Tails' for the coin flip: ").capitalize()  
 if decision in ['Heads', 'Tails']:  
 break  
 else:  
 print("Invalid input. Please choose 'Heads' or 'Tails'.")  
  
 flip\_result = coin\_flip()  
 print(f"The result of the coin flip is: {flip\_result}")  
  
 if decision == flip\_result:  
 print(f"{player1} won the toss and will go first!")  
 return player1, player2  
 else:  
 print(f"{player2} won the toss and will go first!")  
 return player2, player1  
  
def main():  
 print("Welcome to Tic-Tac-Toe!")  
  
 while True:  
 mode = input("Choose a mode (1: Against the Program, 2: Against a Friend): ")  
  
 if mode == '1':  
 player\_name = input("Enter your name: ")  
 computer\_name = "Computer"  
  
 start\_player, second\_player = decide\_first\_player(player\_name, computer\_name)  
  
 board = [[' ' for \_ in range(3)] for \_ in range(3)]  
 print\_board(board)  
  
 while True:  
 player\_turn(board, 'X')  
 print\_board(board)  
  
 if check\_winner(board, 'X'):  
 print(f"Congratulations! {start\_player} wins!")  
 break  
 elif is\_board\_full(board):  
 print("It's a tie!")  
 break  
  
 computer\_turn(board, 'O')  
 print\_board(board)  
  
 if check\_winner(board, 'O'):  
 print(f"{computer\_name} wins. Better luck next time, {start\_player}!")  
 break  
 elif is\_board\_full(board):  
 print("It's a tie!")  
 break  
  
 elif mode == '2':  
 player1 = input("Enter name for Player 1: ")  
 player2 = input("Enter name for Player 2: ")  
  
 start\_player, second\_player = decide\_first\_player(player1, player2)  
 print(f"{start\_player} will go first.")  
  
 board = [[' ' for \_ in range(3)] for \_ in range(3)]  
 print\_board(board)  
  
 while True:  
 player\_turn(board, 'X')  
 print\_board(board)  
  
 if check\_winner(board, 'X'):  
 print(f"Congratulations! {start\_player} wins!")  
 break  
 elif is\_board\_full(board):  
 print("It's a tie!")  
 break  
  
 player\_turn(board, 'O')  
 print\_board(board)  
  
 if check\_winner(board, 'O'):  
 print(f"Congratulations! {second\_player} wins!")  
 break  
 elif is\_board\_full(board):  
 print("It's a tie!")  
 break  
  
 else:  
 print("Invalid choice. Please choose 1 or 2.")  
  
 play\_again = input("Do you want to play again? (yes/no): ").lower()  
 if play\_again != 'yes':  
 break  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()