```python
# AI makes a random move
def ai_place(board, player):
    row, col = random.choice(possibilities(board))
    board[row, col] = player
    print(f"AI (Player {player}, O) chose position: {row * 3 + col + 1}")
    return board

# Checks whether the player has three marks in a horizontal row
def row_win(board, player):
    return any(all(board[x, y] == player for y in range(len(board))) for x in range(len(board)))

# Checks whether the player has three marks in a vertical row
def col_win(board, player):
    return any(all(board[y, x] == player for y in range(len(board))) for x in range(len(board)))

# Checks whether the player has three marks in a diagonal row
def diag_win(board, player):
    return all(board[i, i] == player for i in range(len(board))) or \
           all(board[i, len(board) - 1 - i] == player for i in range(len(board)))

# Evaluates whether there is a winner or a tie
def evaluate(board):
    for player in ['X', 'O']:
        if row_win(board, player) or col_win(board, player) or diag_win(board, player):
            return player
    if np.all(board != ' '):  # Board is full
        return 'Tie'
    return None
```

```python
# importing all necessary libraries
import numpy as np
import random

# Creates an empty board
def create_board():
    return np.array([[' ' for _ in range(3)] for _ in range(3)])

# Map single input to row, col
def map_position(position):
    position -= 1  # Adjust for 0-based indexing
    return divmod(position, 3)  # Returns (row, col)

# Check for empty places on board
def possibilities(board):
    return [(i, j) for i in range(len(board)) for j in range(len(board)) if board[i][j] == ' ']

# Get human player input for the move
def user_place(board, player):
    valid_move = False
    while not valid_move:
        try:
            position = int(input(f"Player {player} (X), enter your position (1-9): "))
            if position < 1 or position > 9:
                print("Invalid input. Enter a number between 1 and 9.")
                continue
            row, col = map_position(position)
            if (row, col) in possibilities(board):
                board[row, col] = player
                valid_move = True
```

```python
# Main function to start the game
def play_game():
    print("Name : Vyom Gupta")
    print("USN  : 1BM22CS333\n")

    board, winner = create_board(), None
    print("Initial Board:")
    print(board)
    print("\nPosition mapping:")
    print("1 | 2 | 3")
    print("---------")
    print("4 | 5 | 6")
    print("---------")
    print("7 | 8 | 9")

    turn = 'X'  # Human starts as 'X'
    while winner is None:
        if turn == 'X':  # Human player's turn
            board = user_place(board, turn)
        else:  # AI's turn
            board = ai_place(board, turn)
        print("Board after move:")
        for row in board:
            print(' | '.join(row))
        winner = evaluate(board)
        turn = 'O' if turn == 'X' else 'X'
    if winner == 'Tie':
        print("It's a tie!")
    else:
        print(f"Winner is: Player {winner}!")
```

```
Name : Vyom Gupta
USN  : 1BM22CS333

Initial Board:
[[' ' ' ' ' ']
 [' ' ' ' ' ']
 [' ' ' ' ' ']]

Position mapping:
1 | 2 | 3
---------
4 | 5 | 6
---------
7 | 8 | 9
Player X (X), enter your position (1-9): 2
Board after move:
  | X |
  |   |
  |   |
AI (Player O, O) chose position: 9
Board after move:
  | X |
  |   |
  |   | O
Player X (X), enter your position (1-9): 3
Board after move:
  | X | X
  |   |
  |   | O
```

```
AI (Player O, O) chose position: 6
Board after move:
  | X | X
  |   | O
  |   | O
Player X (X), enter your position (1-9): 1
Board after move:
X | X | X
  |   | O
  |   | O
Winner is: Player X!
```