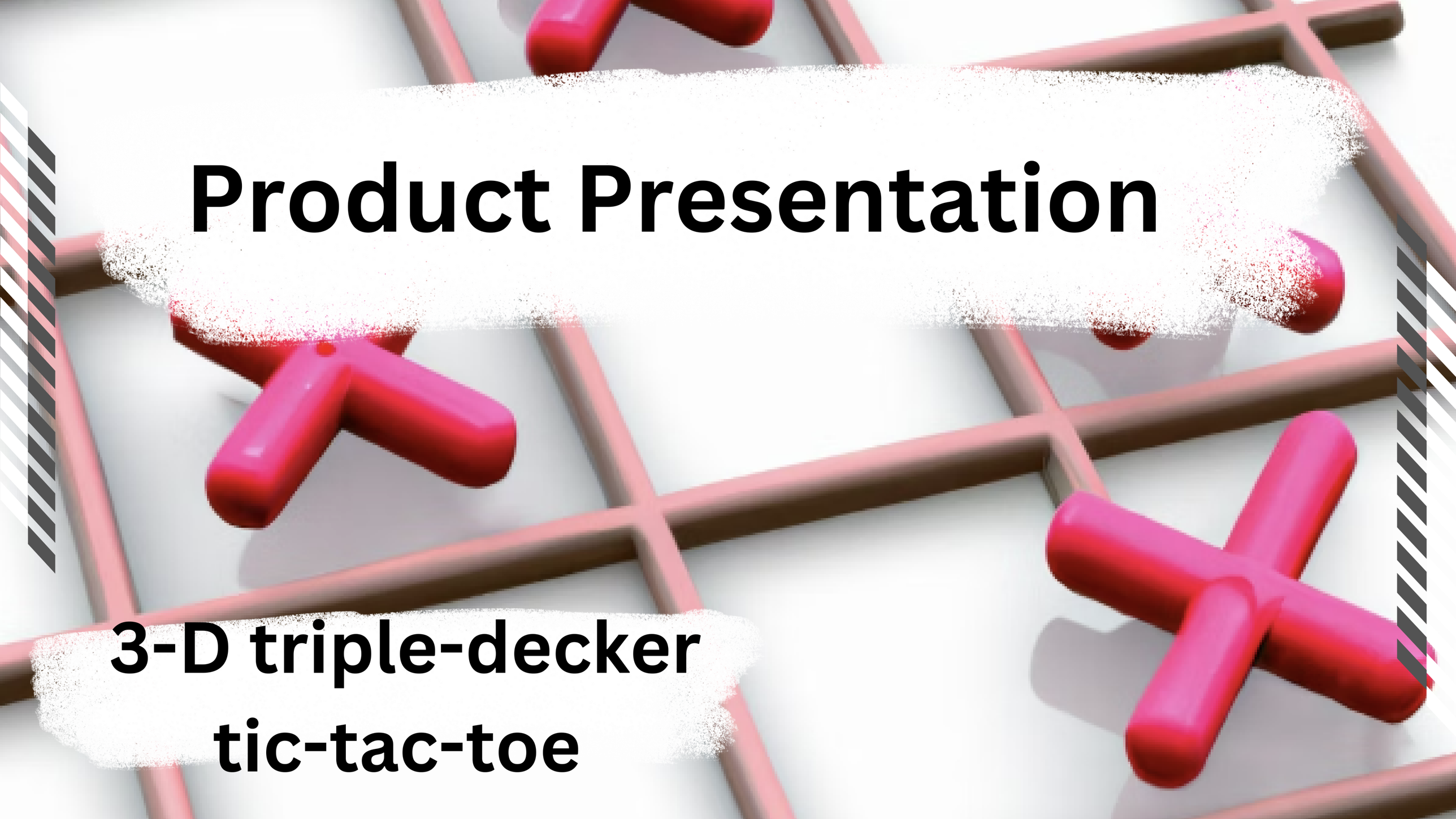
Product Presentation



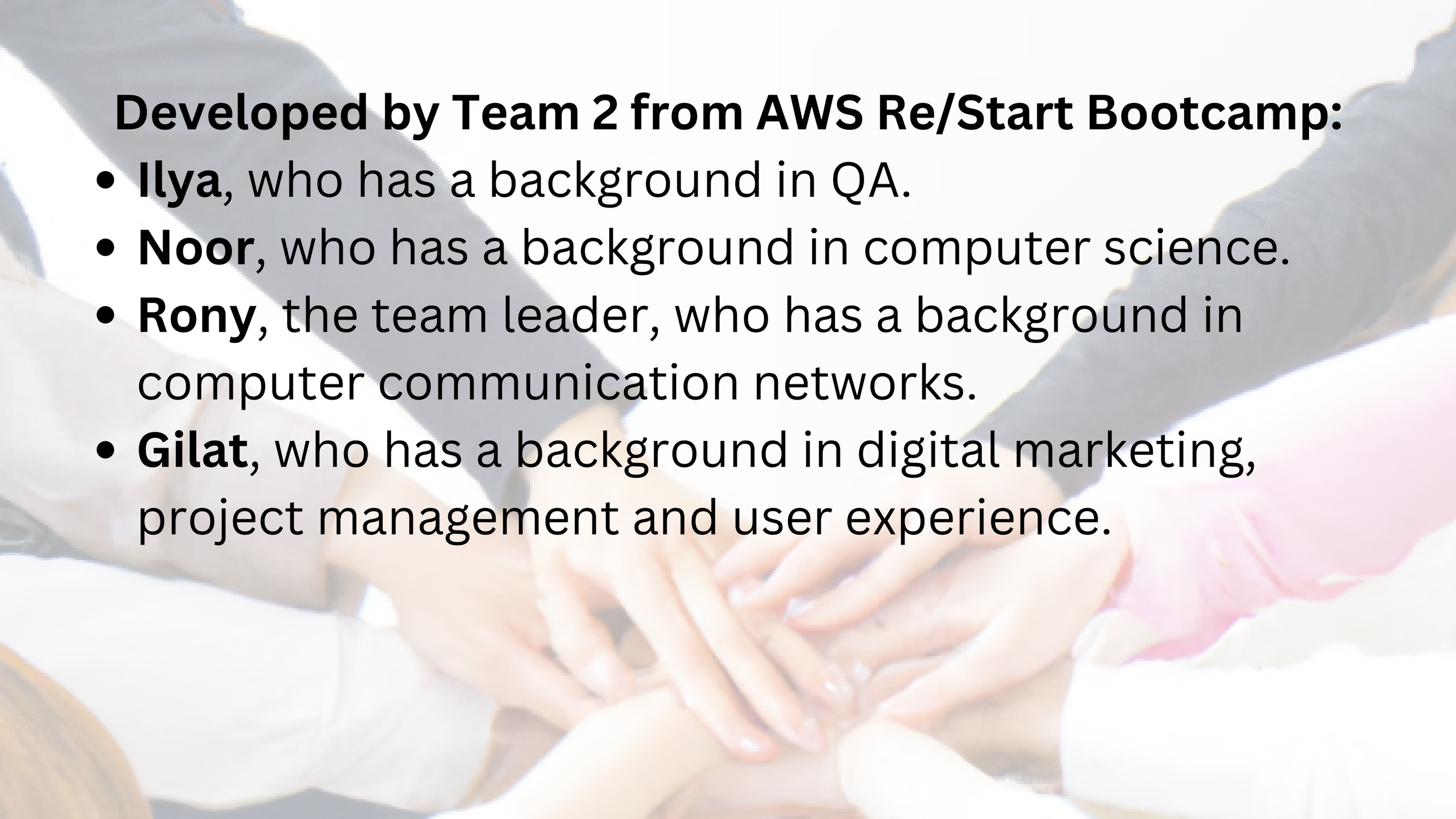
3-D triple-decker tic-tac-toe

the product:

This is a 3D version of the classic Tic-Tac-Toe game.
The 3D board made up of three layers of Tic-Tac-Toe grids stacked on top of each other.

<u>A New Dimension of Strateg:</u>

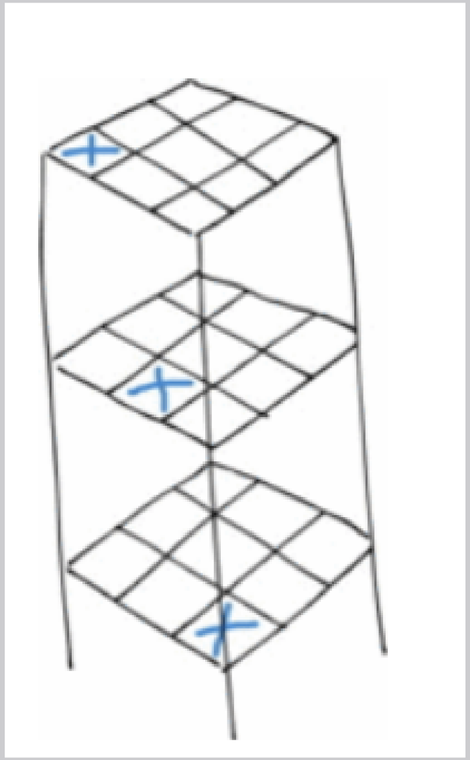
Adds an extra dimension to the game, making it more challenging and requiring players to think strategically about their moves in all three layers.



Developed by Team 2 from AWS Re/Start Bootcamp:

- **Ilya**, who has a background in QA.
- **Noor**, who has a background in computer science.
- **Rony**, the team leader, who has a background in computer communication networks.
- **Gilat**, who has a background in digital marketing, project management and user experience.

Together, we bring a diverse set of skills and perspectives to the table.



CHECK THIS OUT

The customer's requirements

1. Each player enters a name
2. The first player will be chosen randomly
3. The current player will be asked to enter his move
4. The game board will be drawn
5. Evaluation of the board for a win will be checked
6. If the game wasn’t ended go to step 3
7. When the game is over the winner's name will be presented, and a question about a new game will be asked

Winnable scenarios in the game:

The GIF below illustrates

```
y
import random
#https://www.youtube.com/watch?v=maKaugOIhUA
def two_players():
    print('Welcome to the 3D World of Tic-Tac-Toe with Team 2!')
    player1 = input('Player 1, please enter your name: ')
    flag = True
    while flag:
        if player1 != "":
            print("Greetings, {}! Get ready to test your strategic thinking and skill!".format(player1
            flag = False
        else:
            player1 = input('Player 1, please enter your name: ')


    player2 = input('Player 2, please enter your name: ')
    flag1 = True
    while flag1:
        if player2 != "":
```

The program starts by importing the "**random**" module, which is a built-in module in Python that provides functions for generating random numbers. In this program, the "random" module is used to randomly choose the starting player.

The program then defines several functions that are used to create the game board, draw the board, get the player's move, check if a player has won the game, and play the game. These functions are defined using the "**def**" keyword, whichh is used to define a function in Python.

- The "**two_players**" function prompts the players to enter their names and returns their names as a tuple.
- The "**choose_starting_player**" function randomly chooses the starting player using the "**random.choice**" function from the "**random**" module. It takes the names of the two players as arguments and returns the name of the starting player

```
def choose_starting_player():
    #randomly chooses the starting player using the "random.choice" function from the "random" module.
    player1, player2 = two_players()
    starting_player = random.choice([player1, player2])
    unchoose=player2
    if starting_player==player2 :
        unchoose=player1
    else:
        unchoose=player2
    print("now we will make a random choice")
    print("The starting player is {}!".format(starting_player))
    return starting_player,unchoose



def create_board():
    #create a 3D list to represent the game board
    board = []
    for i in range(3):
        layer = []
        for j in range(3):
```

- The "**create_board**" function creates the game board as a "3D" list using loops.
- The "**draw_board**" function draws the current state of the board using loops and conditional statements to check if a square is empty or has been filled by a player.It takes the game board and the names of the two players as arguments.

```python
def draw_board(board,player1,player2):
    #draws the current state of the board using loops and conditional to check if a square is empty or n
    for layer in board:
        print('-' * 25)
        for row in layer:
            # print vertical bar
            print('|', end=' ')
            for square in row:
                if square == player1: # player 1 move
                    print('X', end=' ')
                elif square == player2: # player 2 move
                    print('O', end=' ')
                else: # empty square
                    print('.', end=' ')
                print('|', end=' ')
            print()
    print('-' * 25) #print final horizontal line of hyphens
```

- The "**get_move**" function prompts the player to enter their move and checks if the move is valid.It takes the current player and the game board as arguments and returns the player's move as a tuple.
- The "**check_win**" function checks if a player has won the game by checking for horizontal, vertical, diagonal, and 3D wins using loops and conditional statements.

```python
def get_move(player, board):
    #prompts the player to enter their move and checks if the move is valid
    while True:
        try:
            # get the player's move as a string
            move = input(f"Player {player}, enter your move (layer,row,column): ")
            # split the move string into layer, row, and column
            layer, row, col = map(int, move.split(','))
            # check if the move is valid
            if board[layer][row][col] == 0:
                return layer, row, col
            else:
                print("That square is already taken. Try again.")
        except ValueError:
            print("Invalid input. Try again.")
```

```python
    # check for horizontal wins
    for layer in board:
        for row in layer:
            if row.count(player) == 3:
                return True
    # check for vertical wins
    for layer in board:
        for col in range(3):
            if layer[0][col] == player and layer[1][col] == player and layer[2][col] == player :
                return True
    # check for diagonal wins
    if layer[0][0] == player and layer[1][1] == player and layer[2][2] == player:
        return True
    if layer[0][2] == player and layer[1][1] == player and layer[2][0] == player:
        return True
    if board[0][0][0] == player and board[1][1][1] == player and board[2][2][2] == player:
        return True
    if board[0][0][2] == player and board[1][1][1] == player and board[2][2][0] == player:
        return True
    if board[0][2][0] == player and board[1][1][1] == player and board[2][0][2] == player:
        return True
    if board[0][2][2] == player and board[1][1][1] == player and board[2][0][0] == player:
```

- The "**play_game**" function uses the other functions to play a game of 3D Tic-Tac-Toe.

1. It starts by creating the game board and randomly choosing the starting player.
2. It then enters a loop that continues until there is a winner or the board is full.
3. In each iteration of the loop, the function draws the current state of the board, prompts the current player to enter their move, updates the board with the player's move, and checks if the player has won the game.
4. If a player wins the game, the function prints a message indicating which player has won.
5. If the board is full and there is no winner, the function prints a message indicating that the game is a tie.

Improvements

- The user must enter a name, it cannot be left blank
- This game invites the user to challenge a brain game by encouraging positive words
- The user can play again after the game is over by choose Y/N.

Instruction

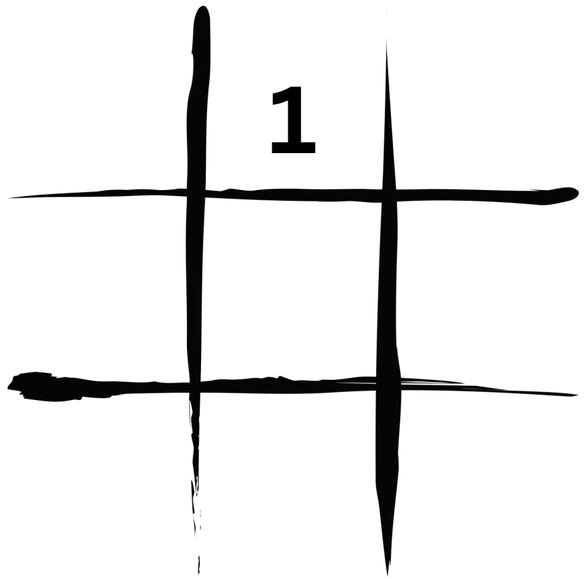
When it's your turn, you will be prompted to enter your move in the format **"layer, row, column".**

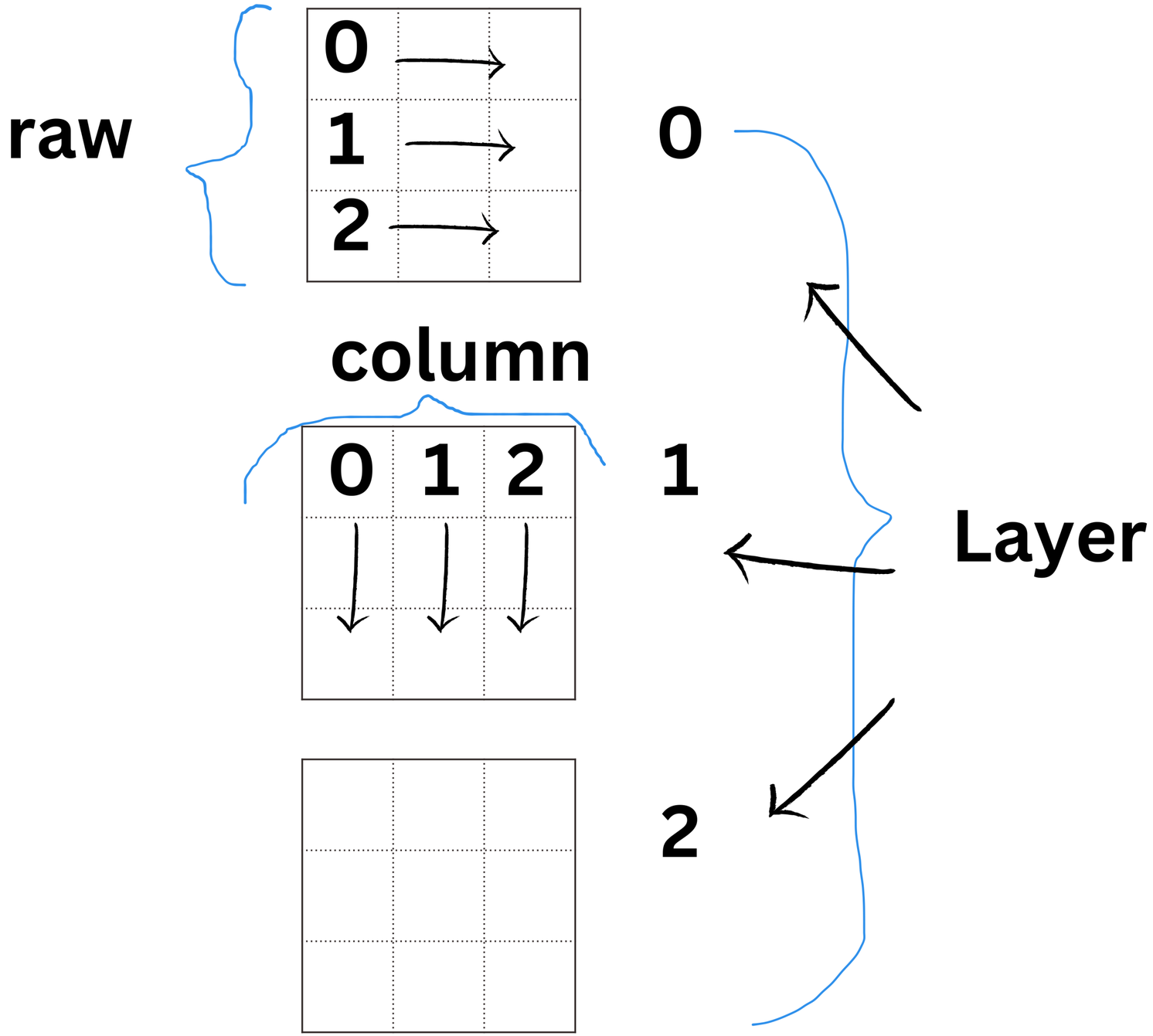
The "**layer**" refers to the layer of the board where you want to place your mark. You can choose a number between 0 and 2.

The "**row**" refers to the row on the layer where you want to place your mark. You can choose a number between 0 and 2.

The "**column**" refers to the column on the layer where you want to place your mark. You can choose a number between 0 and 2.

For example, if you want to place your mark in the middle square of the top layer, you would enter "0, 1, 1".




raw
0
1
2
column
0 1 2
0
1
2
Layer


GitHub