# Documentație Proiect ”X și O”

Studenţi Participanţi:
Albu Andreea-Cristiana – grupa 232
Chelaru Gabriela – grupa 232
Druţă Cati – grupa 232
Roșu Diana-Michesa – grupa 235

# Cuprins

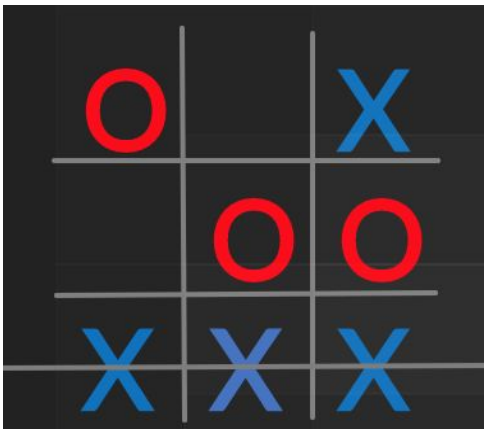
Capitolul I Numărul Paginii

1. Introducere ..... 3
2. Analiza Sistemului..... 4
3. Descrierea Software..... 6
4. User Stories..... 9
5. Descrierea Proiectului ..... 10
6. Circulația informației ..... 12
7. Desfășurare Joc ..... 14
8. Bugs and Fixes ..... 15
9. Listarea programului ..... 16
10. Capturi de ecran ..... 21

# Introducere

**X şi O** (engleză tic-tac-toe) este un joc ce presupune participarea a doi jucători, unul reprezentat prin simbolul ”X” şi unul prin simbolul ”O”. Jocul presupune marcarea de fiecare jucător a unei casete dintr-un tabel de 3 linii şi 3 coloane. Jocul este câştigat de primul jucător care reuşeşte să marcheze 3 casete consecutive, fie pe verticala, orizontală sau verticală. Există şi cazul în care putem determina cine este câştigătorul, această situaţie apare atunci când în urma completării întregului tabel, nu există trei simboluri consecutive.

În modelul reprezentat mai jos, câştigătorul este primul jucător, jucătorul ”X”:



**X şi O** este considerat a fi cel mai vechi joc din istorie. Deşi la început pare un joc pueril, destinat copiilor, X şi O este mult mai mult de atât, este un joc de logică şi strategie, ajutţnd la dezvoltarea gândirii, putând fi considerat chiar un joc didactic. Dacă e să ne legăm de istoria lui, se presupune că îşi are originile în Antichitate, fiind jucat în special jucat de cetăţenii Imperiului Roman în jurul secolului I î.Hr.. Pe de altă parte s-a emis şi o teorie conform căreia, acesta şi-ar avea originile în Egiptul Antic. În anul 1952 a apărut prima versiune a acestui joc, versiune care purta numele de **OXO**. Deşi jocul pare unul relativ simplu, acesta îmbină noţiuni de combinatorică, aranjamente şi permutări şi mai ales, după cum am afirmat în propoziţiile anterioare, o gândire logică.

# Analiza Sistemului

INTRODUCERE:

Analiza poate fi definită prin “dezmembrarea” unui întreg pentru a-i afla natura, funcţionalităţile. Designul este definit pentru a face schiţe preliminare, pentru a schiţa un tipar sau o direcţie înspre care să se planifice. Analiza şi designul sistemelor poate fi caracterizata drept un set de tehnici si procese, un ansamblu de interese, o cultura şi o orientare intelectuala.

Sarcinile din cadrul fazei de analiza a sistemului includ următoarele:

· înţelegerea aplicaţiei
· planificarea proiectului
· studii ale pieţei
· analiza costurilor in raport cu beneficiile
· soluţii alternative
· supervizare, instalare, mentenanţa a sistemului

Acest sistem permite userului sa joace de un număr nelimitat de ori. Întâi se face designul claselor care ţine evidenţa structurii tablei, a selecţiei pixelilor, a mutării la click, a modului de joc etc. În funcţie de mutările jucate de jucători, proiectul va afişa rezultatul in format cat mai atractiv.

Sistemul deja existent e un joc simplu ce se joaca pe hârtie, cu pixul, între doi oameni. Se fac nouă pătrate pe hârtie şi se plasează X-uri şi 0-uri până la deciderea câştigătorului. Procesul se repetă de fiecare dată. Faptul că de fiecare dată trebuie redesenată tabla îngreunează procesul. Dacă se greşeşte inputul, întreg procesul trebuie repetat. Astfel, experienţa utilizatorilor este îngreunată.

SISTEMUL PROPUS

Pentru a minimaliza impedimentele sistemului existent, sistemul propus a fost dezvoltat. Acest proiect îşi propune să reducă hârtia necesară şi să genereze rezultate valide din perspectiva jucătorului. Sistemul oferă cea mai bună interfaţă grafică.

AVANTAJELE SISTEMULUI PROPUS

Jocul a fost făcut mai prietenos pentru utilizatori datorită folosirii unei interfețe grafice.
Userul poate juca oricâte jocuri.
Este de încredere, cu rezultate precise.
Este un bun exerciţiu pentru minte pentru oamenii de orice vârsta.

# Descrierea Software

## Aplicațiile Android și Kotlin

Limbajul de programare Kotlin a devenit din ce în ce mai popular în ultimii ani - şi nu este de mirare. Limbajul încă foarte tânăr este complex, conţine multe caracteristici moderne și este superior colegilor mai în vârstă din punct de vedere al agilităţii în domeniu.

Limbajul Kotlin se bazează pe tehnologii din ecosistemul Java.

Kotlin este un limbaj ce are scopul de a îmbunătăţi conteptele din limbajul de programare Java dar menţinând intercompatibilitate cu acesta, asta însemnând că se poate adauga cod Kotlin într-un proiect existent Java.

Dezvoltarea limbajului Kotlin a început în 2010, în anul 2016 acesta a ajuns la versiunea stabilă 1.0 și în 2019 a ajuns limbajul recomandat de Google pentru dezvoltare Android.

În prezent cu Kotlin putem crea orice este posibil cu Java (aplicaţii Android, desktop, server, consola etc) dar şi aplicaţii iOS, Web (frontend) şi aplicaţii native (generare de cod assembly).

Dezvoltarea de software se poate face utilizând sisteme de operare: Windows, macOS, Linux, FreeBSD şi alte sisteme Unix-like.

În iulie 2011, JetBrains a prezentat Project Kotlin, un limbaj nou pentru JVM. Conducerea JetBrains, Dmitry Jemerov, a spus că majoritatea limbilor străine nu au caracteristicile pe care le căutau, cu excepţia Scala. Cu toate acestea, el a menţionat timpul de compilare lent al Scala ca fiind o deficienţă. Unul dintre obiectivele declarate de Kotlin este să compilezi cât mai rapid Java. În februarie 2012, JetBrains a deschis proiectul sub licenţa Apache 2.

Numele provine din insula Kotlin, lângă Sankt Petersburg. Andey Breslav a menţionat că echipa a decis să o numească după o insulă la fel cum Java a fost numită după insula indoneziană Java (deşi limbajul de programare Java a fost numit probabil după cafea).

JetBrains speră că noua limbă va conduce vânzările Intellij IDEA.

Kotlin v1.0 a fost lansat pe 15 februarie 2016. Aceasta este considerată a fi prima versiune oficială stabilă, iar JetBrains s-a angajat să compatibilizeze pe termen lung, pe termen lung, începând cu această versiune.
La Google I/O 2017, Google a anunţat asistenţă de primă clasă pentru Kotlin pe Android.
Kotlin v1.2 a fost lansat pe 28 noiembrie 2017. La distribuirea codului JVM şi a platformelor JavaScript a fost adăugată recent această versiune (din versiunea 1.3, programarea multiplataformă este experimentală ). Demonstraţia completă a fost făcută cu noul plugin Kotlin / JS Gradle.
Kotlin v1.3 a fost lansat pe 29 octombrie 2018, aducând coroutine pentru programare asincronă.

La 7 mai 2019, Google a anunţat că limbajul de programare Kotlin este acum limba preferată pentru dezvoltatorii de aplicaţii Android.

Conducătorul dezvoltării, Andrey Breslav, a spus că Kotlin este proiectat să fie un limbaj orientat pe obiecte de rezistenţă industrială şi un „limbaj mai bun" decât Java, dar să fie în continuare complet interoperabil cu codul Java, permiţând companiilor să facă o trecere treptată de la Java la Kotlin.

Semicolonii sunt opţionale ca terminator de instrucţiuni ; în majoritatea cazurilor, o linie nouă este suficientă pentru ca compilatorul să deducă că declaraţia s-a încheiat.

Declaraţiile de variabile Kotlin şi listele de parametri au tipul de date vin după numele variabilei (şi cu un separator de colon ), similar cu Pascal şi TypeScript .

Variabilele din Kotlin pot fi numai de citire, declarate cu cuvântul cheie val , sau mutabile, declarate cu cuvântul cheie var.

Membrii clasei sunt publici în mod implicit, iar clasele în sine sunt finale implicit, ceea ce înseamnă că crearea unei clase derivate este dezactivată, cu excepţia cazului în care clasa de bază este declarată cu cuvântul cheie deschis .

În plus faţă de clasele şi metodele (numite funcţii de membru în Kotlin) de programare orientată pe obiecte, Kotlin acceptă şi programarea procedurală cu utilizarea funcţiilor . Funcţiile Kotlin (şi constructorii) acceptă argumente implicite , liste de argumente cu lungime variabilă , argumente numite şi supraîncărcare prin semnătură unică. Funcţiile membrilor de clasă sunt virtuale, adică trimise în funcţie de tipul de rulare a obiectului la care sunt apelate.

Kotlin 1.3 adaugă suport (experimental) pentru contracte (inspirat de designul lui Eiffel prin contract paradigma de programare).

**Stil de funcţionare**

Kotlin relaxează restricţia Java, permiţând variabileleor statice să existe şi în exteriorul unui corp de clasă. Obiectele şi funcţiile statice se pot defini şi la nivelul superior al pachetului, nefiind nevoie de un nivel de clasă redundant.

Pentru compatibilitatea cu Java, Kotlin oferă o JvmNameadnotare care specifică un nume de clasă folosit atunci când pachetul este vizualizat dintr-un proiect Java. De exemplu @ file:JvmName ("JavaClassName").

# User stories , backlog creation

1. Sunt elev şi aţ vrea ca aplicaţia să fie uşor de jucat

2. Aş dori ca aplicaţia să aibă un aspect plăcut

3. Ca şi student aş dori ca atunci când deschid aplicaţia să nu aibă un sunet strident şi eventual să aibă un buton de mute pentru acesta

4. Mi-aş dori ca aplicaţia să nu fie complicată, ci uşor de accesat

5. Lucrez şi aş vrea ca aplicaţia să fie rapidă şi eficientă

6. Nu mi-aş dori să aibă un grad de dificultate mare

7. Ca User, aş vrea să aibă meniul simplu, să nu mă încurc în butoane

8. Aş vrea să pot alege nivelul de dificultate

9. Nu mi-ar face plăcere ca aceasta să fie plină de reclame.

Personaje: Vlad este elev la Scoala Gimnaziala nr 3 din Bucureşti. El este mai retras şi îşi petrece timpul din pauze pe telefon. Îşi doreşte să înceapă jocuri pe care le poate termina rapid, înainte de următoarea oră.

Ana-Maria, studentă la ASE. Se plictiseşte repede la cursuri şi vrea să îşi umple timpul cu un joc uşor şi rapid care să o ţină antrenată.

Marina, pensionară. Îi plac aplicaţiile simple, cu o interfaţă plăcută deoarece culorile aprinse şi butoanele excesive o deranjează.

Dragoş, este angajat la banca şi s-a săturat de reclamele care îi luau din timp pentru a se juca. Totodată, nu ar vrea ca nivelul de dificultate să fie prea scăzut. Îşi doreşte un joc care să-l stimuleze intelectual.

Petrică, angajat la benzinarie. Face naveta cu metroul şi s-a săturat să deschidă aplicaţii cu sunet strident, ar vrea să existe un buton de mute.

# Descrierea Proiectului

DEFINIREA PROBLEMEI

Proiectul constă în dezvoltarea şi implementarea unui program ce simulează un joc de X şi O, între un utilizator şi calculator. Acest produs elimină partea clasică a jocului, transformă tradiţionalul joc X şi O, cel jucat pe hârtie, astfel se elimină necesitatea utilizării unui pix, a unui creion a foilor pe care altfel, ar trebui să le porţi mereu cu tine. Sistemul este capabil să genereze sute de jocuri X şi O, fără întreruperi şi fără apariţia altor probleme.

PRIVIRE DE ANSAMBLU ASUPRA PROIECTULUI

Acest proiect este divizat în câteva clase şi fişiere, apelate ulterior în clase. Fiecare clasă deşi diferită are componente vitale pentru o bună funcţionare a proiectului.

DESCRIEREA PROECTULUI

- ImageView

Atunci când deschidem aplicaţia, în partea de sus putem observa numele aplicaţiei create, un logo. Partea aceasta de program se ocupă de afişarea titlului

- TextView

În urma terminării jocului, atunci când funcţia IsGameOver nu mai găseşte celule libere, iar HasComputerWon sau HasPlayerWon trebuie să declare un câştigător, este afişat un text pentru a informa utilizatorul cu privire la câştigarea sau pierderea jocului. Această funcţie este îndeplinită de secvenţa respectivă de cod.

- HasComputerWon & HasPlayerWon

Funcţiile acestea se concentrează pe a verifica dacă jucătorul, respectiv calculatorul, câştigă, returnând rezultatul atunci când este apelată. Cu alte cuvinte, verifică toate combinaţiile posibile prin care se poate câştiga jocul, se

verifică diagonalele, liniile şi coloanele pentru a vedea dacă există trei X-uri sau O-uri consecutive.

- IsGameOver

După cum sugerează şi numele, verifică dacă jocul a ajuns la o finalitate. Jocul se termină atunci când unul dintre jucători câştigă sau când este remiză, nu câştigă nimeni. Funcţia verifică dacă mai sunt celule disponibile pe tablă pentru a fi completate, atunci când nu găseşte nimic, înseamnă că jocul s-a terminat, fără un câştigător.

- Board & GridLayout

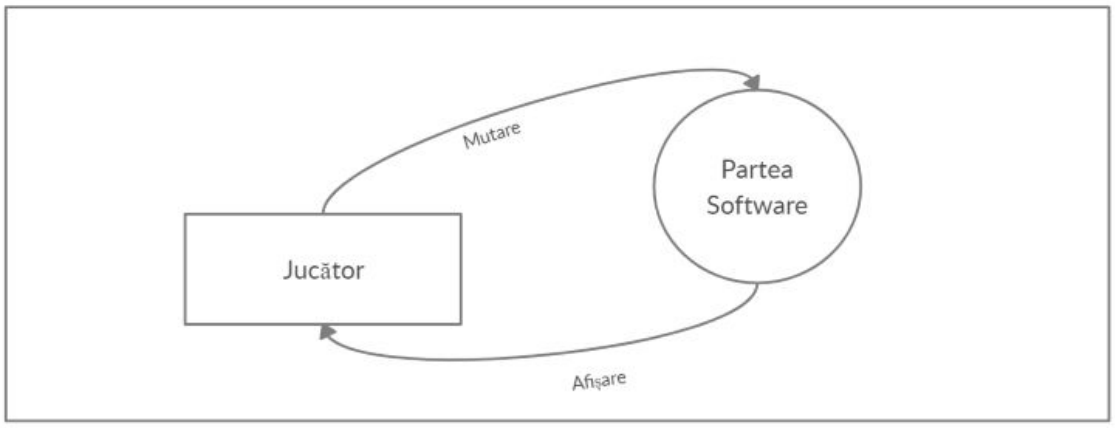
Afişarea tablei de joc se face sub forma unui tabel de 3x3, fiind reţinută într-un array bidimensional.

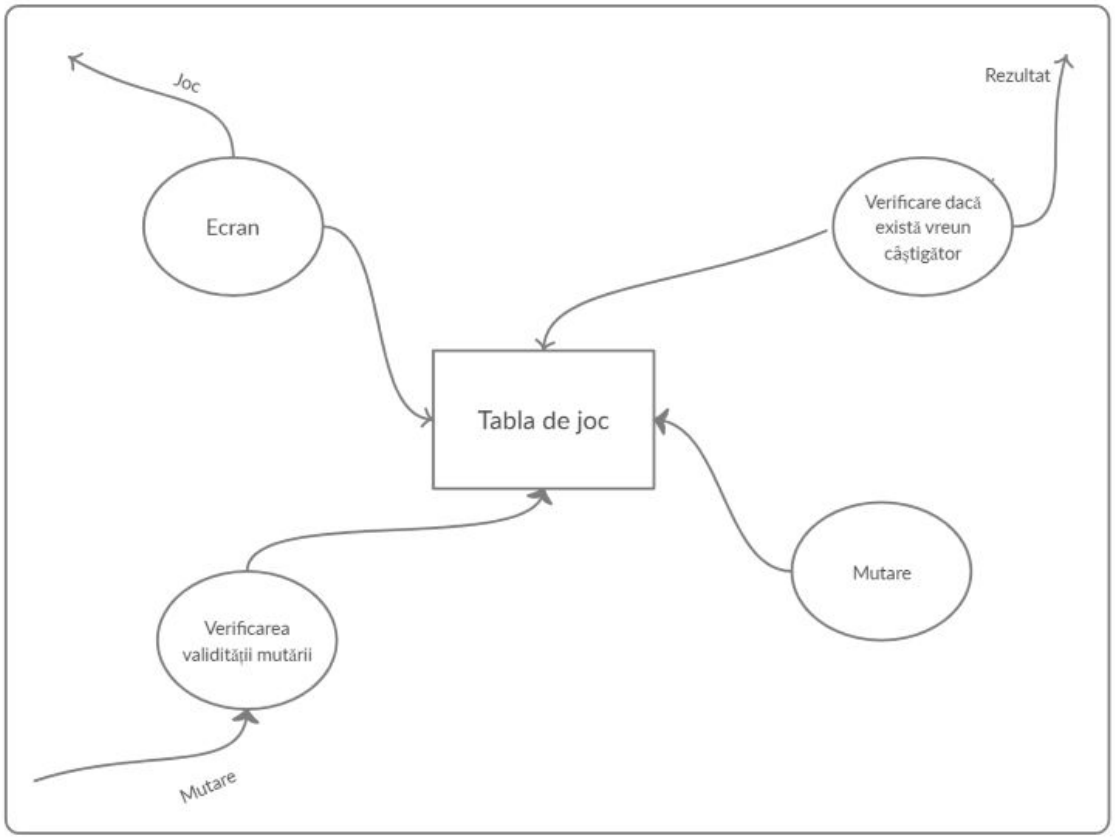
- Button

Butonul de Restart, care apare în jocul paginii, oferind utilizatorului şansa de a reîncepe un joc.

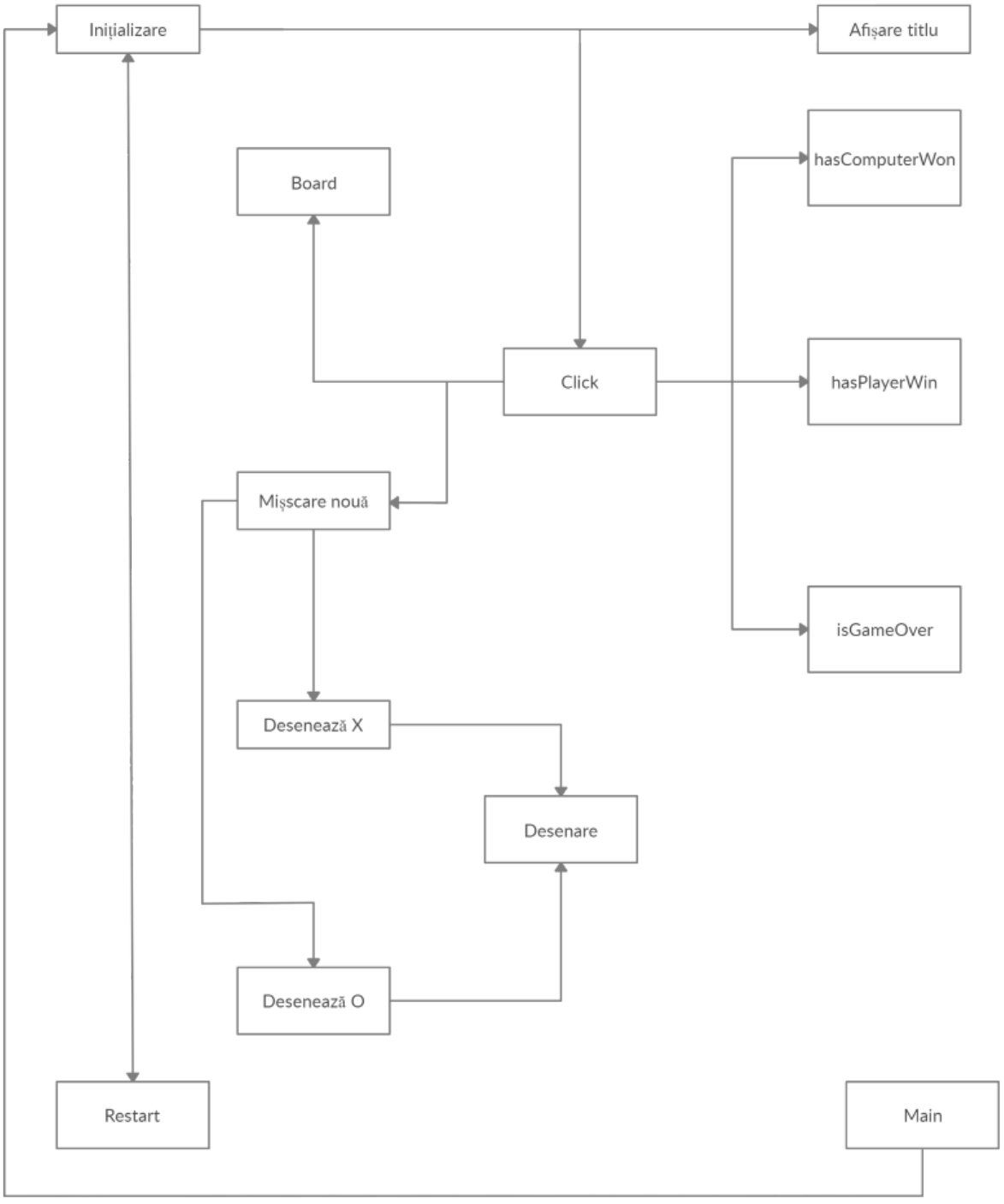
# Desfășurare Joc

Circularea Informațiilor și comenzilor

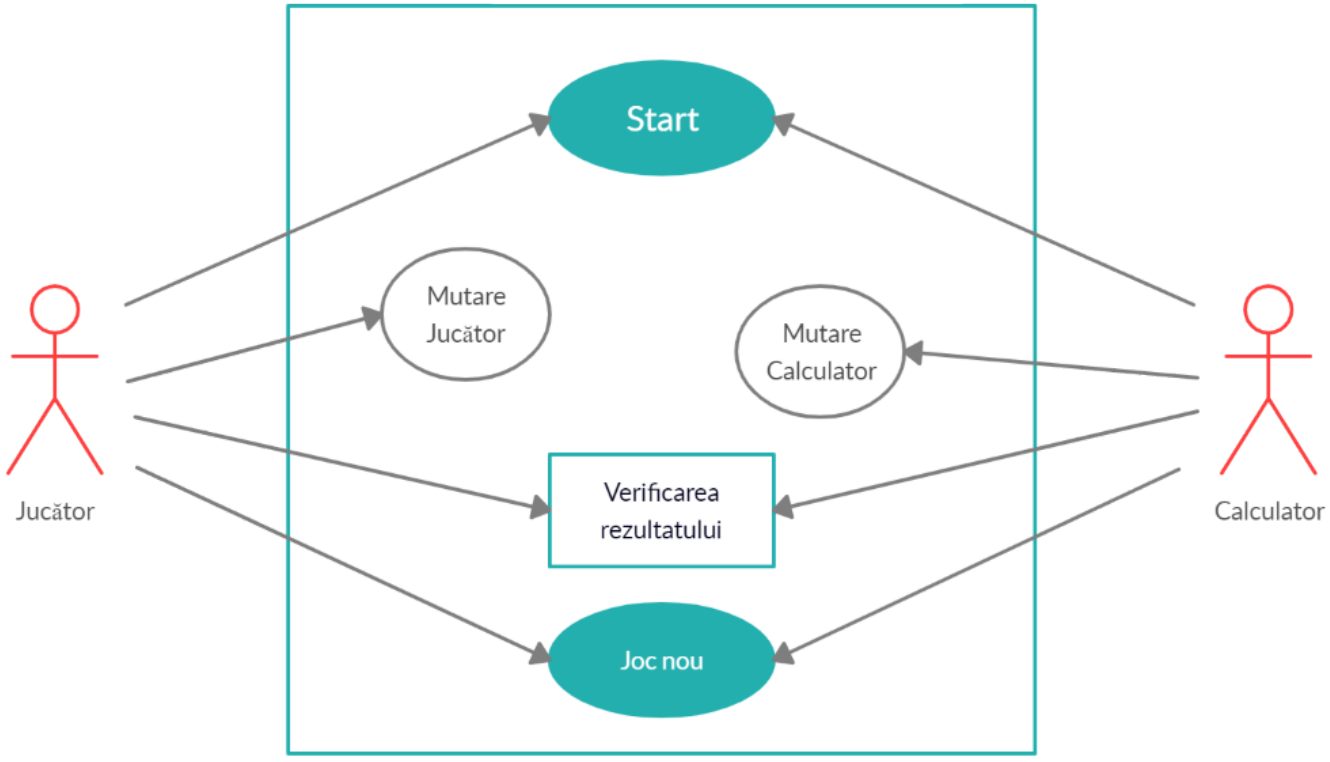


# Circularea Informației



Desfășurare Joc



# Bugs and Fixes

Un dezavantaj ce i-ar putea îndepărta pe utilizatori de produs după un anumit punct este imposibilitatea de a câştiga jocul împotriva sistemului, cel mai bun rezultat pentru aceştia fiind o remiză. O perspectiva de dezvoltare este astfel ideea de a lăsa, în mod intenţionat, utilizatorul să câştige într-un anumit procent din jocuri. Astfel sistemul devine mai atractiv şi utilizatorii sunt scutiţi de frustrarea de a nu reuşi niciodată să câştige.

Pentru a implementa această idee, e nevoie să schimbăm algoritmul Minimax, care decide fiecare mutare a calculatorului, adăugându-i în mod intenţionat un bug ce să permită ca acesta să fie uneori înfrânt. Pentru algoritmul clasic, la fiecare pas sunt generate toate scenariile posibile şi este aleasă mutarea care duce la cele mai favorabile. Pentru a îndepărta problema cu care ne confruntam, algoritmul clasic ar trebui înlocuit cu o formă a sa care sa aleagă, după un număr de paşi aleator, unul dintre scenariile mai puţin optimiste. Un alt mod de a gestiona problema ar fi ca, după acelaşi număr de paşi ales de fiecare dată printr-o funcţie aliatoare, calculatorul să nu mai ţină cont de rezultatul obţinut în urma aplicării algoritmului şi să facă o mutare într-o căsuţa libera aleatoare. Astfel, se oferă jucătorului posibilitatea de a câştiga.

# LISTAREA PROGRAMULUI

- Board

```
package com.example.xo1st

class Board {
    companion object {
        const val PLAYER = "X"
        const val COMPUTER = "O"
    }

    val board = Array( size: 3) { arrayOfNulls<String>( size: 3) }

    val availableCells: List<Cell>
        get() {
            val cells = mutableListOf<Cell>()
            for (i in board.indices) {
                for (j in board.indices) {
                    if (board[i][j].isNullOrEmpty()) {
                        cells.add(Cell(i, j))
                    }
                }
            }
            return cells
        }


    val isGameOver: Boolean
        get() = hasComputerWon() || hasPlayerWon() || availableCells.isEmpty()


    fun hasComputerWon(): Boolean {
        if (board[0][0] == board[1][1] &&
            board[0][0] == board[2][2] &&
            board[0][0] == COMPUTER ||
            board[0][2] == board[1][1] &&
            board[0][2] == board[2][0] &&
            board[0][2] == COMPUTER
        ) {
            return true
        }

        for (i in board.indices) {
            if (
                board[i][0] == board[i][1] &&
                board[i][0] == board[i][2] &&
                board[i][0] == COMPUTER ||
```

```
                board[0][i] == board[1][i] &&
                board[0][i] == board[2][i] &&
                board[0][i] == COMPUTER
            ) {
                return true
            }
        }

        return false
    }

    fun hasPlayerWon(): Boolean {

        if (board[0][0] == board[1][1] &&
            board[0][0] == board[2][2] &&
            board[0][0] == PLAYER ||
            board[0][2] == board[1][1] &&
            board[0][2] == board[2][0] &&
            board[0][2] == PLAYER
        ) {
            return true
        }

        for (i in board.indices) {
            if (
                board[i][0] == board[i][1] &&
                board[i][0] == board[i][2] &&
                board[i][0] == PLAYER ||
                board[0][i] == board[1][i] &&
                board[0][i] == board[2][i] &&
                board[0][i] == PLAYER
            ) {
                return true
            }
        }

        return false
    }


    var computersMove: Cell? = null

```

```
88
89      fun minimax(depth: Int, player: String): Int {
90          if (hasComputerWon()) return +1
91          if (hasPlayerWon()) return -1
92
93          if (availableCells.isEmpty()) return 0
94
95          var min = Integer.MAX_VALUE
96          var max = Integer.MIN_VALUE
97
98          for (i in availableCells.indices) {
99              val cell = availableCells[i]
100             if (player == COMPUTER) {
101                 placeMove(cell, COMPUTER)
102                 val currentScore = minimax( depth: depth + 1, PLAYER)
103                 max = Math.max(currentScore, max)
104
105                 if (currentScore >= 0) {
106                     if (depth == 0) computersMove = cell
107                 }
108
109                 if (currentScore == 1) {
110                     board[cell.i][cell.j] = ""
111                     break
112                 }
113
114                 if (i == availableCells.size - 1 && max < 0) {
115                     if (depth == 0) computersMove = cell
116                 }
117
118             } else if (player == PLAYER) {
119                 placeMove(cell, PLAYER)
120                 val currentScore = minimax( depth: depth + 1, COMPUTER)
121                 min = Math.min(currentScore, min)
122
123                 if (min == -1) {
124                     board[cell.i][cell.j] = ""
125                     break
126                 }
127             }
128             board[cell.i][cell.j] = ""
129         }
130
131         return if (player == COMPUTER) max else min
```

```
132     }
133
134     fun placeMove(cell: Cell, player: String) {
135         board[cell.i][cell.j] = player
136     }
137
138 }
```

- Cell

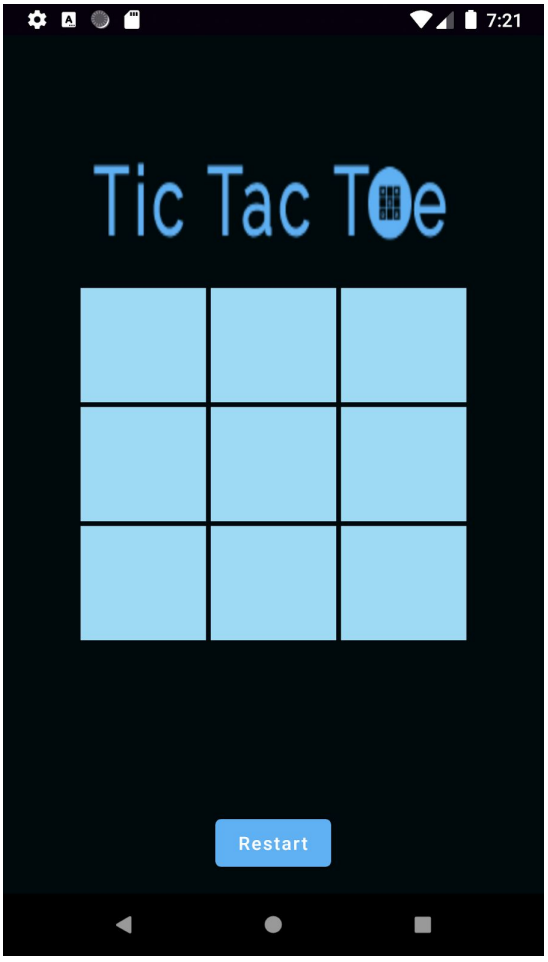
```
1  package com.example.xo1st
2
3  class Cell(val i : Int, val j : Int)
```

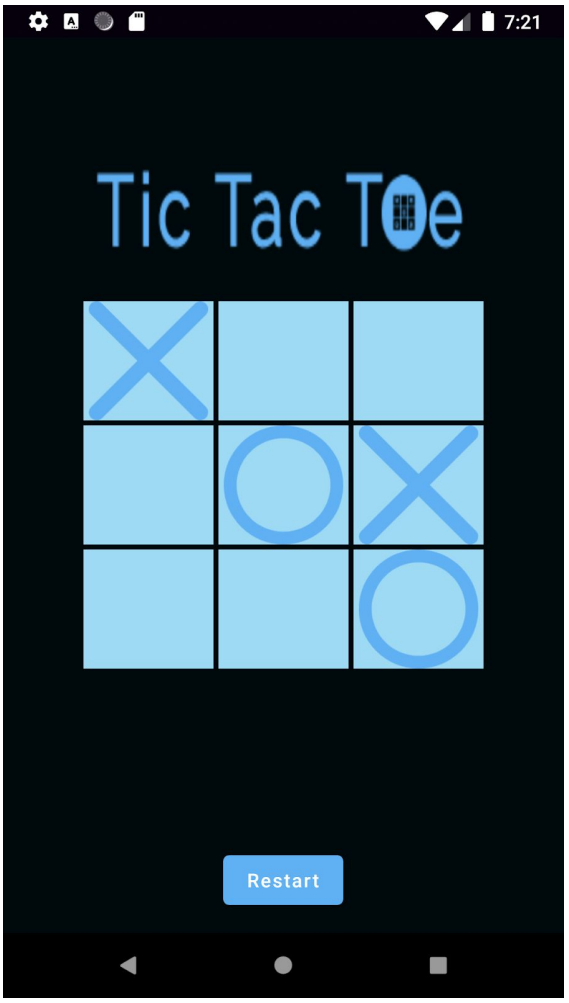
- MainActivity

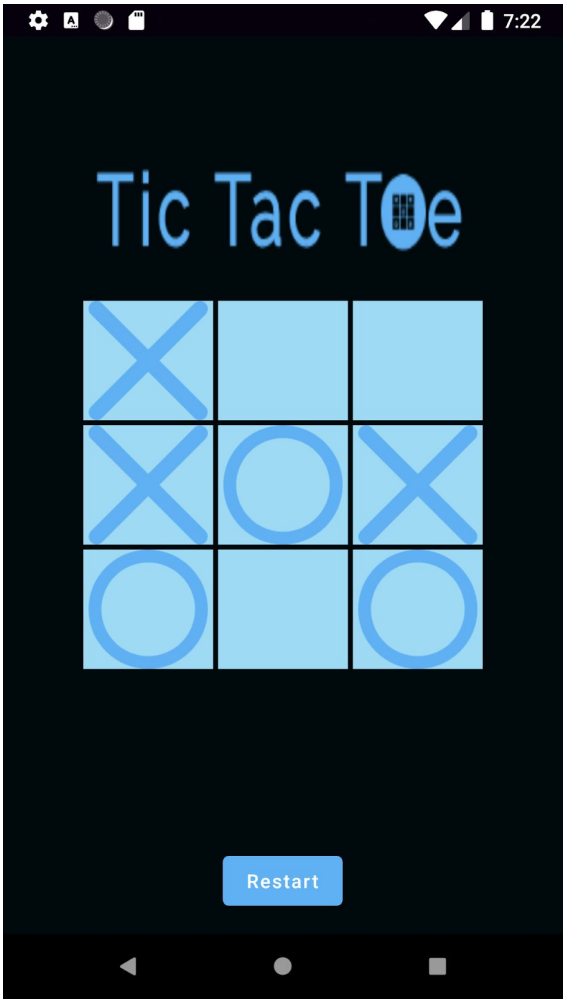
```
51      private fun loadBoard() {
52          for (i in boardCells.indices) {
53              for (j in boardCells.indices) {
54                  boardCells[i][j] = ImageView(this)
55                  boardCells[i][j]?.layoutParams = GridLayout.LayoutParams().apply {
56                      rowSpec = GridLayout.spec(i)
57                      columnSpec = GridLayout.spec(j)
58                      width = 250
59                      height = 230
60                      bottomMargin = 5
61                      topMargin = 5
62                      leftMargin = 5
63                      rightMargin = 5
64                  }
65                  boardCells[i][j]?.setBackgroundColor(ContextCompat.getColor(this, R.color.colorPrimary))
66                  boardCells[i][j]?.setOnClickListener(CellClickListener(i, j))
67                  layout_board.addView(boardCells[i][j])
68              }
69          }
70      }
71
72      inner class CellClickListener(
73          private val i: Int,
74          private val j: Int
75      ) : View.OnClickListener {
76
77          override fun onClick(p0: View?) {
78
79              if (!board.isGameOver) {
80                  val cell = Cell(i, j)
81                  board.placeMove(cell, Board.PLAYER)
82                  board.minimax(0, Board.COMPUTER)
83                  board.computersMove?.let {
84                      board.placeMove(it, Board.COMPUTER)
85                  }
86                  mapBoardToUi()
87              }
88              when {
89                  board.hasComputerWon() -> text_view_result.text = "Computer Won"
90                  board.hasPlayerWon() -> text_view_result.text = "You Won"
91                  board.isGameOver -> text_view_result.text = "Game Tied"
92              }
93          }
94      }
95  }
```

# SCREEN SHOTS

- Computer Won

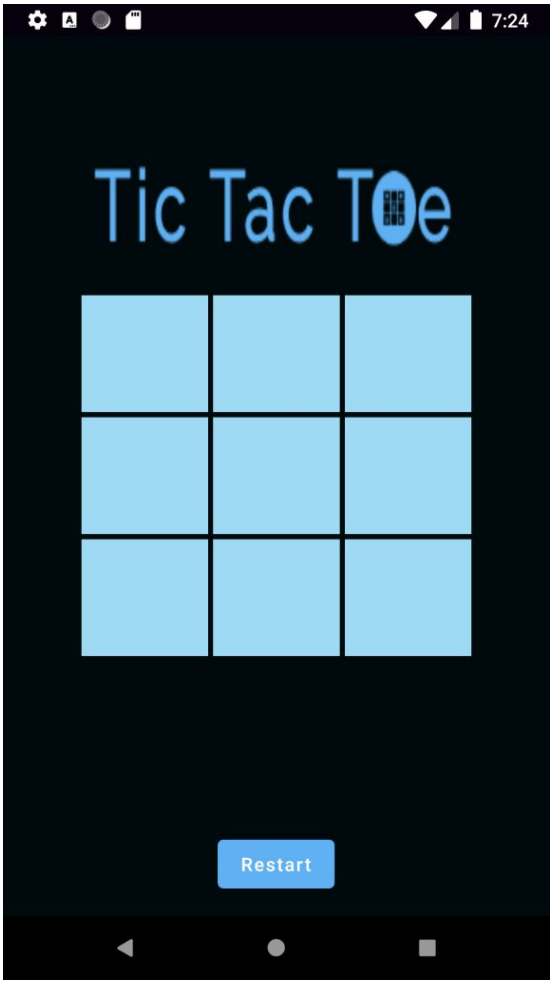


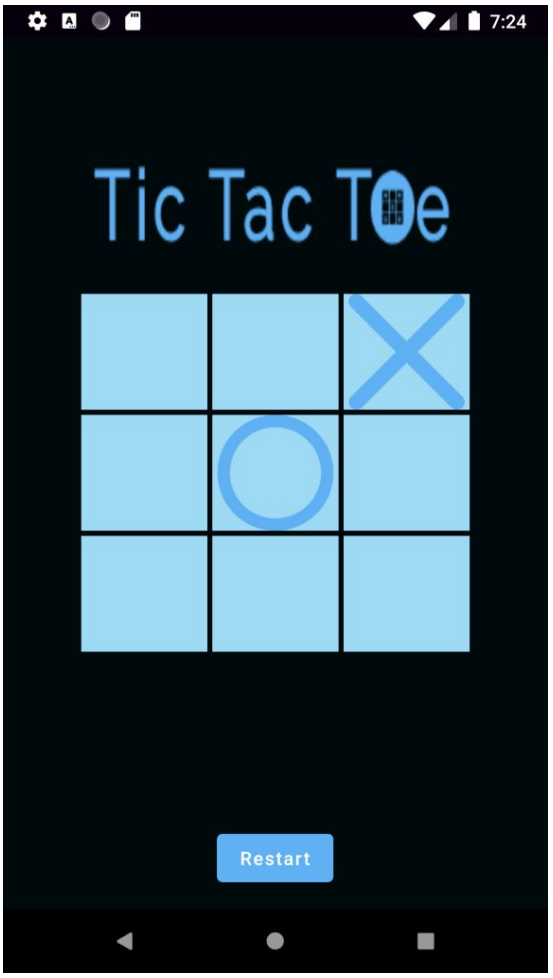


7:21
Tic Tac Toe
Restart

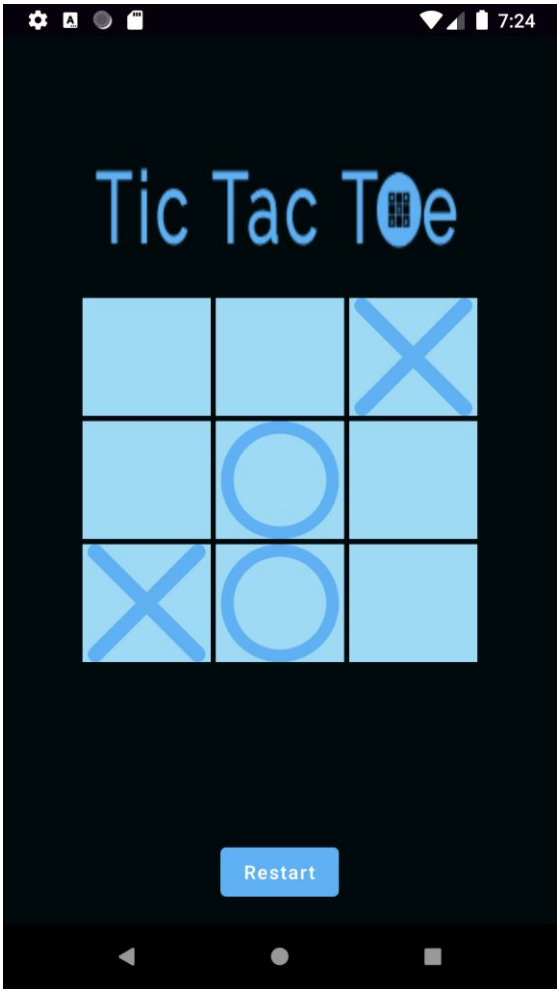

7:22
Tic Tac Toe
Restart

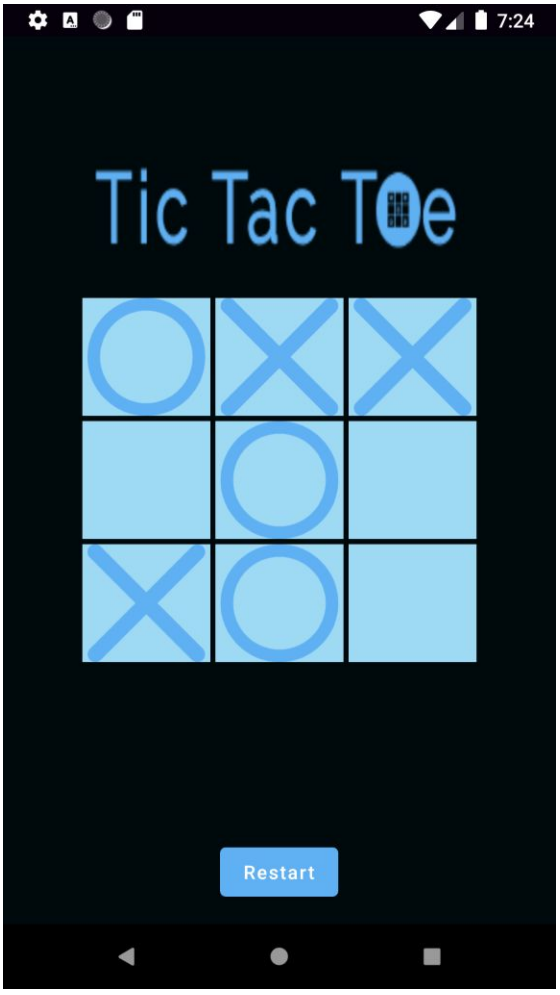

7:22
Tic Tac Toe
Computer Won
Restart

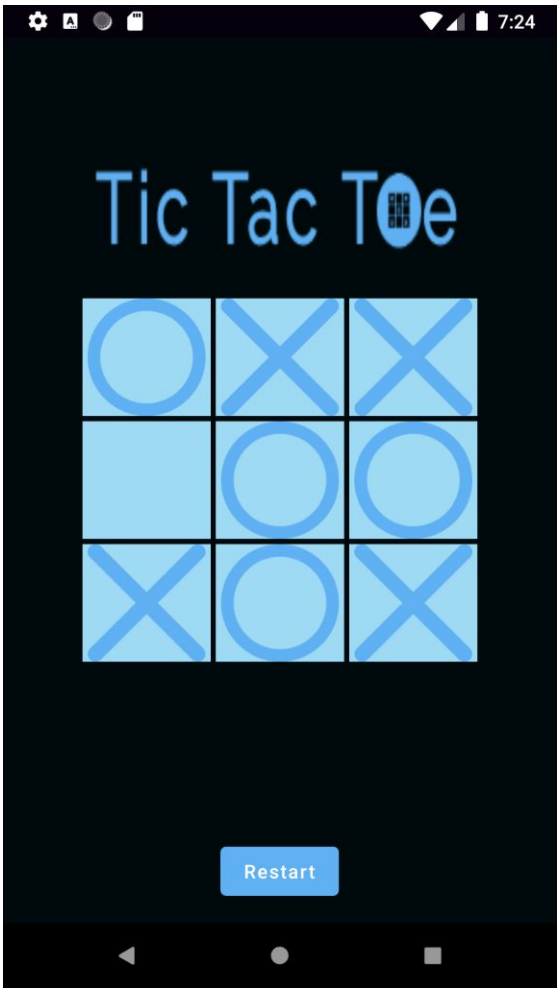
- You Won

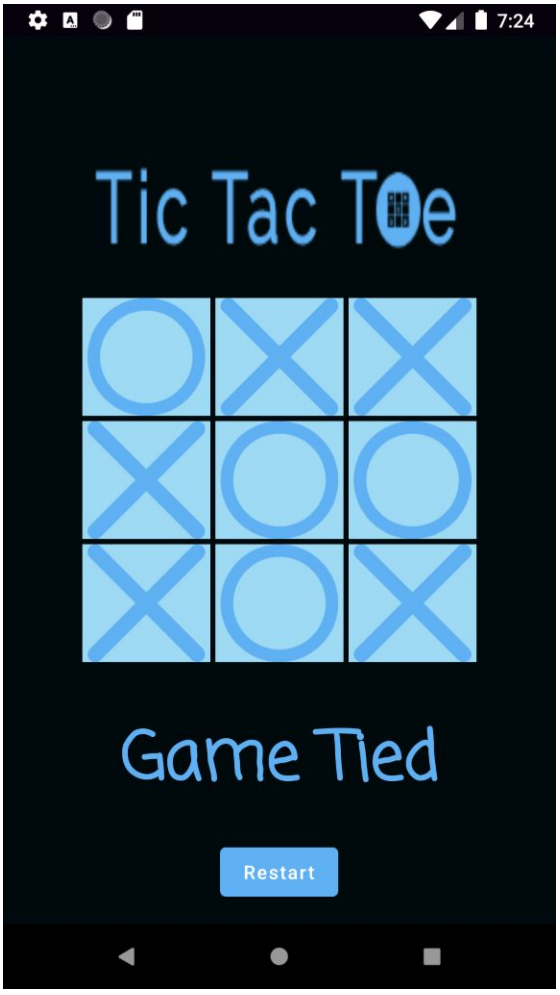



7:24
Tic Tac Toe
Restart


7:24
Tic Tac Toe
Restart


7:24
Tic Tac Toe
Restart


7:24
Tic Tac Toe
Game Tied
Restart