Ministerul Educației și Cercetării al Republicii Moldova
Colegiul Universității Tehnice al Moldovei

RAPORTUL
STAGIULUI DE PRACTICĂ

TEMA: Dezvoltarea unei aplicații web interactive „X si O”

Elevul: **Grigoriev Artur**
Grupa: **PTPP-241**
Specialitatea: **Programarea și testarea produselor program**
Baza de practică: **Colegiul Universității Tehnice a Moldovei**

Conducătorul stagiului de practică de la
Colegiul Universității Tehnice a Moldovei
Moraru Magdalena

Chișinău 2025

REZUMAT

Lucrarea de faţă reprezintă raportul stagiului de practică efectuat la Colegiul Universităţii Tehnice a Moldovei, în cadrul specialităţii „Programarea şi testarea produselor program”. Tema raportului: „Dezvoltarea unei aplicații web interactive „X si O”. Volumul lucrării este de 22 pagini, structurate în conformitate cu cerințele academice: introducere, două capitole principale, concluzii, bibliografie şi anexă.

Capitolul 1 „Analiza domeniului de studiu” abordează: importanța aplicațiilor informatice și a jocurilor web interactive în procesul didactic şi de divertisment, analiza comparativă a unor sisteme similare de „Tic Tac Toe”, identificarea funcționalităților cheie şi definirea scopului, obiectivelor, cerinţelor ale produsului propus.

Capitolul 2 „Realizarea sistemului” prezintă: arhitectura modulară a aplicației şi descrierea codului pentru sistemul de autentificare, implementarea interfeţei de joc cu motorul Phaser.js şi logica de gestionare a tablei, alternarea turelor, animaţiile grafice, modulul de inteligenţă artificială, care combină verificări rapide de câştig/blocare şi algoritmul Minimax cu optimizare alpha-beta, prin limite de adâncime şi număr de noduri.

Principalele realizări ale proiectului:

— autentificare şi gestionare de sesiuni pentru utilizatori;
— interfaţă modernă, responsivă şi intuitivă;
— suport pentru modurile single-player (versus AI) şi multiplayer local;
— posibilitatea alegerii dimensiunii grilei de joc (3×3 sau 5×5);
— agent AI competitiv, capabil să decidă optim prin euristici şi căutare adversară.

CUPRINS

Listă de Abrevieri și Definiți

- AI (Artificial Intelligence / Inteligență Artificială): Agent software adversar ce simulează un jucător uman în modul single-player.
- Algoritmul Minimax: Algoritm AI pentru alegerea mutării optime în jocuri cu 2 jucători, evaluând consecințele presupunând joc optim advers.
- Alpha-Beta (Optimizare Alpha-Beta): Optimizare Minimax ce accelerează căutarea eliminând ramuri de joc irelevante.
- Cookie-uri (Cookies): Fișiere text mici stocate în browser pentru gestionarea sesiunilor sau preferințelor utilizator.
- CSS (Cascading Style Sheets): Limbaj pentru definirea stilului vizual (culori, fonturi, layout) al elementelor HTML.
- Funcție Euristică (Heuristic Function): Funcție AI ce estimează rapid favorabilitatea unei stări de joc, ghidând decizia.
- HTML (HyperText Markup Language): Limbaj standard de marcare pentru structurarea conținutului paginilor web.
- JavaScript (JS): Limbaj de programare pentru interactivitate web și implementarea logicii jocului în browser.
- JSON (JavaScript Object Notation): Format text standard pentru schimbul/stocarea datelor structurate (folosit pentru localStorage).
- localStorage: Mecanism de stocare persistentă a datelor (ex: autentificare) direct în browserul utilizatorului.
- Modularitate / Arhitectură Modulară: Principiu de împărțire a sistemului în componente (module) independente pentru dezvoltare/mentenanță facilă.
- Motor de Joc (Game Engine): Software specializat ce oferă funcționalități de bază (grafică, fizică, interacțiune) pentru crearea jocurilor.
- Phaser.js: Motor de joc JS utilizat în proiect pentru grafică 2D, animații și interacțiuni în jocul Tic Tac Toe.

INTRODUCERE

În lumea contemporană, tehnologiile informaționale joacă un rol esențial, influențând profund felul în care trăim, comunicăm și ne desfășurăm activitățile zilnice. Aceste tehnologii nu sunt limitate doar la anumite domenii, ci pătrund în aproape toate aspectele vieții noastre, transformând societatea într-un mod semnificativ.

Impactul tehnologiilor informaționale asupra vieții cotidiene este evident în aproape fiecare activitate pe care o realizăm. De exemplu, dispozitivele inteligente simplifică sarcinile domestice, iar digitalizarea a schimbat fundamental modul în care lucrăm și interacționăm profesional. Accesul generalizat la internet, folosirea dispozitivelor mobile a deschis noi metode de comunicare și colaborare – rețele sociale, locuri de activitate online – atât la lucru, cât și la educație. E-guvernarea, comerțul electronic, divertismentul digital reprezintă alte modalități în care tehnologia poate îmbunătăți experiența zilnică a oamenilor.

Aplicațiile software sunt instrumente fundamentale în acest proces de transformare, aducând îmbunătățiri semnificative în domenii diverse. În sănătate, aceste aplicații ajută la monitorizarea pacienților, diagnosticarea rapidă a bolilor și optimizarea gestionării dosarelor medicale, facilitând astfel accesul la servicii medicale de calitate. În educație, platformele digitale și instrumentele interactive permit accesul la resurse educaționale, adaptând procesul de învățare nevoilor individuale ale utilizatorilor și eliminând barierele geografice.

Procesul de dezvoltare al unei aplicații software implică mai multe etape bine definite. În prima fază, cea de proiectare, dezvoltatorii analizează cerințele utilizatorilor și creează prototipuri pentru a defini clar structura și funcționalitățile aplicației. Urmează etapa de programare, în care specificațiile sunt transformate în cod funcțional. Testarea riguroasă este apoi necesară pentru a asigura funcționarea corectă, securitatea și performanța aplicației. După lansare, mentenanța joacă un rol important în menținerea relevanței aplicației prin actualizări și adaptări continue la noile tehnologii și cerințe ale utilizatorilor.

În concluzie, dezvoltarea tehnologiilor informaționale și a aplicațiilor software reprezintă elemente indispensabile în progresul societății moderne, îmbunătățind calitatea vieții, eficiența economică și accesul generalizat la informații. Aceste tehnologii continuă să modeleze viitorul într-un ritm alert, oferind oportunități inovatoare și provocări captivante.

1 ANALIZA DOMENIULUI DE STUDIU

Importanța dezvoltării aplicațiilor informatice este crucială în contextul actual al digitalizării, întrucât acestea facilitează învățarea principiilor de programare, modelarea datelor și implementarea inteligenței artificiale. Aplicațiile permit integrarea interacțiunii utilizatorilor cu logica de business și gestionează atât securitatea, cât și persistența datelor.

Proiectul propus – un joc Tic Tac Toe dezvoltat cu Phaser.js [1] – se inspiră din analiza unor sisteme similare și adaugă funcționalități precum alegerea dimensiunii grilei și modul AI avansat bazat pe algoritmul Minimax cu optimizare alpha-beta.

Scopul aplicației este de a crea o experiență de joc captivantă și educativă, cu interfață modernă, scor vizibil și animații grafice, oferind atât mod single-player, cât și multiplayer local. De asemenea, aplicația asigură autentificarea utilizatorilor și este construită modular, pregătită pentru extindere și scalabilitate ulterioară.

1.1 Importanța temei

Importanța dezvoltării aplicațiilor informatice este esențială în contextul actual al transformării digitale, reprezentând un domeniu fundamental pentru înțelegerea și aplicarea principiilor programării. Proiectele software permit explorarea interacțiunii dintre diverse subdomenii, precum elaborarea algoritmilor de decizie, gestionarea stării aplicației prin structuri de date adecvate, comunicarea eficientă între interfeței utilizator și logica de business, precum și asigurarea securității datelor și a experienței utilizatorului.

În societatea contemporană, aplicațiile informatice constituie puncte de contact esențiale pentru începători în programare, facilitând înțelegerea controlului fluxului de execuție, manipularea structurilor matriciale sau a altor tipuri de date, precum și tratarea evenimentelor generate de interacțiunea utilizatorului și actualizarea dinamică a interfeței. Mai mult, aceste proiecte pot integra elemente de inteligență artificială, prin implementarea unor agenți capabili să analizeze situații complexe și să optimizeze procesul decizional, evidențiind astfel potențialul tehnologiilor avansate în dezvoltarea aplicațiilor.

Persistența datelor reprezintă un aspect crucial, iar utilizarea bazelor de date, chiar și în forme simple precum stocarea locală, pregătește tranziția către arhitecturi client-server, în care backend-ul gestionează utilizatori, statistici și istoricul operațiunilor. De asemenea, definirea clară a interfețelor web și mobile, realizată prin tehnologii standard precum HTML, CSS și JavaScript, subliniază importanța accesibilității și portabilității aplicațiilor, permițând utilizatorilor să acceseze serviciile indiferent de dispozitivul folosit.

Securitatea cibernetică, abordată la nivel conceptual, impune o atenție sporită asupra validării datelor de intrare și izolării logicii critice. Astfel, dezvoltarea aplicațiilor devine un proces complex, care combină aspecte tehnice cu responsabilitatea privind protecția datelor și integritatea experienței utilizatorului.

În concluzie, dezvoltarea aplicațiilor informatice are un caracter didactic și inovator, demonstrând că procesul nu se limitează la implementarea inițială, ci continuă prin integrarea de module avansate, interfețe sofisticate, funcționalități sociale și analize statistice ale performanțelor. Acest domeniu reprezintă o punte între teoria informatică și aplicabilitatea practică, deschizând perspective largi în cadrul transformării digitale a societății.

1.2 Sisteme similare cu proiectul realizat

În dezvoltarea aplicației TIC TAC TOE, analiza sistemelor similare este esențială pentru identificarea celor mai bune practici și direcții de îmbunătățire. Sistemele existente variază de la aplicații web simple la platforme cu interfețe grafice avansate și funcționalități extinse. Proiectul propriu, realizat cu Phaser.js, integrează elemente esențiale din aceste sisteme, oferind moduri single-player și multiplayer, precum și opțiuni de grilă variabilă și inteligență artificială competitivă.

Tabelul 1 Sisteme Similare cu proiectul realizat

Caracteristici	Tic Tac Toe Glow	Tic Tac Toe Online (BYRIL)	Google Tic Tac Toe	Proiectul propriu (Phaser.js)
Interfață	Animată, efecte vizuale atractive	Simplificată	Minimalistă	Clară, intuitivă
Dimensiune grilă	3x3, 5x5, 7x7, 11x11	3x3	3x3	3x3 și 5x5
Complexitate joc	Ridicată, strategie extinsă	Scăzută, dueluri rapide	Foarte scăzută	Medie, AI cu Minimax și alfa-beta
Moduri de joc	Mod Battle cu scoruri și runde	Dueluri rapide	Joc simplu	Single-player și multiplayer local
Experiență utilizator	Implicare pe termen lung	Repetitivă, fidelizare redusă	Accesibilitate imediată	Echilibrată, educațională

În tabelul 1 sunt prezentate funcționalitățile principale ale sistemelor similare existente. Tic Tac Toe Glow [6] oferă o interfață animată, grile variate și moduri competitive care cresc implicarea utilizatorilor. Tic Tac Toe Online (BYRIL) [7] are un design simplu și joc rapid, dar cu complexitate redusă. Google Tic Tac Toe [8] este accesibil imediat, însă foarte minimalist. Proiectul propriu combină o interfață clară, opțiuni de grilă și un AI competitiv, oferind atât mod single-player, cât și multiplayer local. Astfel, proiectul integrează și adaptează elemente cheie pentru a crea o aplicație educațională și interactivă.

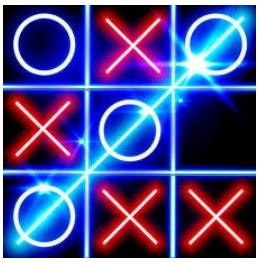

Fig 1.1 Tic Tac Toe Glow

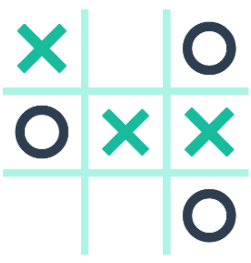

Fig 1.2 Tic Tac Toe Online (BYRIL)

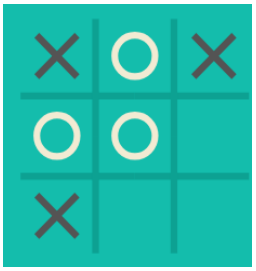

Fig 1.3 Google Tic Tac Toe

1.3 Scopul, obiectivele și cerințele sistemului

Scopul principal al sistemului „Tic Tac Toe” este de a oferi o experiență digitală interactivă și atractivă, care valorifică simplitatea mecanicii clasice a jocului „X și Zero” și elimină monotonia jocurilor simple prin implementarea unor funcționalități suplimentare, precum alegerea dimensiunii tablei de joc (de exemplu, „3x3” sau „5x5”), opțiunea de a juca împotriva unui adversar uman sau a unui agent de inteligență artificială și integrarea unei logici strategice avansate pentru modul automatizat. Sistemul răspunde astfel nevoilor de divertisment, stimulare strategică și accesibilitate, într-un cadru intuitiv.

Obiectivele sistemului vizează atât aspecte funcționale, cât și experiența utilizatorului. Sistemul trebuie să asigure o platformă robustă și intuitivă, care să permită desfășurarea jocului „X și Zero” în regim multiplayer local și single-player împotriva unui adversar controlat de inteligență artificială. Se urmărește diversificarea experienței prin setări configurabile, cum ar fi alegerea simbolului („X” sau „O”) și selecția dimensiunii tablei de joc. De asemenea, se dezvoltă o interfață vizuală coerentă și modernă, care facilitează o interacțiune rapidă și plăcută. Un alt obiectiv este implementarea unui sistem de scor vizibil și a unor animații grafice pentru mutări, pentru creșterea implicării și rejucabilității.

Sistemul trebuie să permită autentificarea utilizatorilor și selecția modului de joc printr-un meniu interactiv. Utilizatorii pot alege să joace împotriva unui alt utilizator sau a unui AI. Tabla de joc se adaptează la dimensiunile selectate, iar sistemul gestionează alternarea tururilor și validarea condițiilor de câștig. În modul single-player, strategia AI este implementată prin algoritmul Minimax cu optimizare alpha-beta, asigurând o provocare competitivă. Sistemul afișează în timp real scorurile și mesajele corespunzătoare pentru victorie, remiză sau finalul jocului.

Din perspectiva performanței, sistemul asigură un timp de răspuns redus și o redare fluidă a animațiilor. Securitatea accesului este garantată prin autentificare, realizată prin mecanisme simple bazate pe localStorage și cookie-uri, adecvate naturii academice a proiectului. Ergonomia interfeței respectă principiile accesibilității și clarității, asigurând compatibilitatea cu diverse dimensiuni de ecran prin stilizare adaptivă. Scalabilitatea este asigurată printr-o arhitectură modulară a codului, care permite extinderea ulterioară cu noi moduri de joc sau funcționalități. Sistemul oferă o experiență estetică, coerentă și intuitivă, contribuind la succesul transformării digitale în domeniul aplicațiilor recreative.

2 REALIZAREA SISTEMULUI

2.1 Descrierea la nivel de cod pe module

Mai jos se prezintă o documentare scurtă a sistemului de autentificare, care include fragmente de cod relevante, o introducere concisă și explicații succinte ale codului. Documentația este redactată într-un stil impersonal, conform normelor academice.

Această documentare descrie modul de funcționare al sistemului de autentificare pentru o aplicație web, ce gestionează înregistrarea, autentificarea prin intermediul localStorage și a cookie-urilor. Pentru claritate, sunt prezentate fragmente de cod ce ilustrează verificarea sesiunii, funcția de logout și gestionarea cookie-urilor.

La încărcarea documentului se verifică existența unei sesiuni active. Codul recuperează valoarea stocată sub cheia „debug_login_status” din localStorage și, dacă pagina curentă nu este „auth.html", iar sesiunea nu este activă, utilizatorul este redirecționat către „auth.html”.

```
document.addEventListener('DOMContentLoaded', function () {
  let debugLoginStatus;
  try {
    debugLoginStatus = JSON.parse(localStorage.getItem('debug_login_status'));
  } catch (e) {
    debugLoginStatus = localStorage.getItem('debug_login_status');
  }

  const path = window.location.pathname;
  if (path !== '/auth.html' && debugLoginStatus !== 'loggedIn') {
    window.location.href = 'http://localhost:5500/auth.html';
  }
});
```

Funcția de logout elimină sesiunea din localStorage și redirecționează utilizatorul către pagina de autentificare:

```
function logout() {
  localStorage.removeItem('debug_login_status');
  window.location.href = 'http://localhost:5500/auth.html';
}
```

Manipularea datelor în localStorage se efectuează prin funcția „saveData”, ce salvează datele în format JSON, și funcția „getData”, ce recuperează și parsează datele:

```
const saveData = (key, value) => {
  localStorage.setItem(key, JSON.stringify(value));
};

const getData = (key) => {
  let data = localStorage.getItem(key);
  try {
    return data ? JSON.parse(data) : null;
  } catch (e) {
    return null;
  }
};
```

În final, un fragment de cod actualizează interfața, ascunzând butonul de logout dacă sesiunea nu este activă:

```
document.addEventListener('DOMContentLoaded', () => {
  const logoutBtn = document.getElementById('logout-btn');
  const isLoggedIn = getData('debug_login_status') === 'loggedIn';
  if (!isLoggedIn && logoutBtn) {
    logoutBtn.style.display = 'none';
  }
});
```

Accesul la jocul X și O este permis doar utilizatorilor autentificați. Jocul oferă două moduri de funcționare: Player versus Player și Player versus AI. Structura și logica de bază a aplicației sunt prezentate în continuare, cu referire la codul de configurare inclus în Anexa A.

Inițializarea jocului presupune setarea parametrilor esențiali, precum numărul de rânduri și coloane, lungimea combinației câștigătoare și încărcarea resurselor grafice necesare (imaginile pentru X și O). Pentru reprezentarea tablei de joc se utilizează obiectul „matrix”, fiecare celulă fiind actualizată cu valoarea „x” sau „o” în funcție de mutările efectuate.

Codul de configurare stabilește dimensiunea tablei, parametrii inițiali și pregătește scena folosind motorul de joc Phaser [1]. În cadrul funcției preload sunt încărcate imaginile corespunzătoare pieselor, acestea fiind utilizate la fiecare mutare a jucătorilor.

Phaser reprezintă un motor de joc open-source, specializat pentru dezvoltarea de jocuri 2D în mediul web. Acesta oferă o gamă largă de funcționalități pentru gestionarea graficii, animațiilor, interacțiunilor și logicii de joc, facilitând astfel crearea rapidă și eficientă a aplicațiilor interactive. Platforma este bazată pe JavaScript și permite rularea jocurilor direct în browser, fără a necesita instalarea unor componente suplimentare. Printre avantajele utilizării Phaser se numără suportul extins pentru scene, sisteme de fizică, gestionarea resurselor grafice și a evenimentelor de interacțiune cu utilizatorul [1].

În cadrul aplicației, Phaser este utilizat pentru a desena tabla de joc, a gestiona plasarea pieselor și a implementa logica de interacțiune. Funcția create are rolul de a construi vizual tabla de joc și de a configura evenimentele necesare pentru interacțiunea cu utilizatorul. Dimensiunea fiecărei celule este calculată în funcție de dimensiunile totale ale tablei, iar pentru fiecare celulă se desenează un dreptunghi și se setează un eveniment de tip „pointerdown” [1]. Acest eveniment permite rularea unei funții la fiecare clic, asigurând astfel interactivitatea necesară desfășurării jocului.

Codul de mai jos ilustrează modul în care se realizează aceste operațiuni cu ajutorul funcțiilor oferite de Phaser:

```
function create() {
  const cellW = this.sys.game.config.width / cols;
  const cellH = this.sys.game.config.height / rows;
  const outer = 0x0f1425;
  const inner = 0x0066a7;
  const g = this.add.graphics();
  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      const x = c * cellW;
      const y = r * cellH;
      matrix[`${r},${c}`] = null;
      g.fillStyle(inner, 1).fillRect(x, y, cellW, cellH);
      const rect = this.add.rectangle(
        x + cellW / 2, y + cellH / 2, cellW - 8, cellH - 8, outer
      ).setInteractive();
      rect.on('pointerdown', () =>
        TakeTurn.call(this, x + cellW / 2, y + cellH / 2, r, c)
      );
    }
  }
}
```

Funcția TakeTurn reprezintă elementul central al mecanismului de interacțiune în cadrul jocului X și O, fiind apelată la fiecare acțiune de clic efectuată de utilizator asupra unei celule a tablei de joc. Această funcție gestionează atât validarea mutării, cât și actualizarea vizuală și logică a stării jocului.

La începutul execuției, funcția verifică dacă jocul este deja finalizat, prin intermediul variabilei „gameOver”. În cazul în care jocul s-a încheiat, orice acțiune suplimentară este ignorată, prevenind astfel modificarea stării finale. Ulterior, se construiește cheia celulei selectate, utilizând coordonatele de rând și coloană, și se verifică dacă această celulă este deja ocupată în obiectul matrix. Dacă celula nu este liberă, funcția se oprește, asigurând respectarea regulilor jocului.

Dacă mutarea este validă, funcția determină jucătorul curent pe baza variabilei turn și atribuie simbolul corespunzător, „x” sau „o”, în matricea de joc. Pentru actualizarea interfeței grafice, se selectează imaginea asociată jucătorului curent „ObjectX” sau „ObjectO” și se stabilește o scalare specifică pentru fiecare simbol, pentru a compensa mărimile vizual diferite ale pieselor pe tablă. Imaginea este adăugată pe poziția selectată, cu o scalare inițială de zero, permițând aplicarea unei animații de creștere prin funcția „tweens.add” din biblioteca Phaser. După actualizarea grafică, variabila turn este inversată, asigurând alternarea jucătorilor la fiecare mutare.

În final, se rulează funcția „Check” care are rolul de a evalua starea curentă a jocului pentru a determina dacă există un câștigător sau dacă partida s-a încheiat la egalitate. Această funcție parcurge întreaga matrice a tablei de joc, analizând fiecare celulă ocupată de unul dintre jucători. Pentru fiecare celulă, se verifică dacă există o secvență de simboluri identice, de lungime egală cu valoarea WIN_LEN, pe orizontală, verticală sau pe cele două diagonale principale. Direcțiile de verificare sunt definite prin vectorii de deplasare: dreapta [0,1], jos [1,0], diagonală principală [1,1] și diagonală secundară [1,-1].

Pentru fiecare jucător, funcția verifică toate pozițiile posibile de start pentru o secvență câștigătoare. Dacă, pornind dintr-o anumită celulă, se găsește o linie de WIN_LEN de simboluri consecutive în oricare dintre cele patru direcții, funcția apelează „declareWin”, marcând astfel sfârșitul jocului și afișând câștigătorul. Dacă nu se identifică nicio combinație câștigătoare și toate celulele din matrice sunt ocupate, funcția stabilește că jocul s-a încheiat la egalitate, apelând „displayWinner” cu mesajul corespunzător și setând variabila „gameOver” la valoarea „true”. Astfel, funcția „Check” asigură evaluarea corectă și completă a stării jocului după fiecare mutare. Implementarea poate fi găsită in Anexa D.

Funcția „declareWin” este apelată atunci când unul dintre jucători obține o combinație câștigătoare, conform logicii implementate în funcția „Check’. Aceasta incrementează scorul jucătorului câștigător în obiectul „scores”, apoi apelează funcția „displayWinner” pentru a afișa un mesaj corespunzător pe ecran, care include numele jucătorului și scorul actualizat. De asemenea, variabila „gameOver” este setată la valoarea `true`, blocând orice alte mutări suplimentare și marcând finalul partidei.

```
function declareWin(p){
  scores[p]++;
  displayWinner.call(this, `Player ${p} wins! Score: ${scores[p]}`);
  console.log(p, 'wins!', scores[p]);gameOver = true;}
```

Funcția „displayWinner” primește ca parametru un mesaj text și îl afișează într-o fereastră pop-up, utilizând elemente HTML [2] dedicate, „popup-cover” și „popup-text”. Aceasta asigură vizibilitatea rezultatului pentru utilizator și setează, la rândul său, variabila „gameOver” la „true”, pentru a preveni continuarea jocului după finalizare. Astfel, cele două funcții colaborează pentru a oferi o experiență de utilizare clară și coerentă la finalul fiecărei partide.

```
function displayWinner(msg){
  const cover = document.getElementById('popup-cover');
  const text  = document.getElementById('popup-text');
  cover.style.display = 'block';
  text.innerText = msg;
  gameOver = true;
}
```

Prin intermediul acestor funcții, scorurile sunt actualizate automat, iar utilizatorul primește feedback vizual imediat privind rezultatul partidei, fie că este vorba despre o victorie sau o remiză.

Modulul de inteligență artificială (AI) implementat în cadrul aplicației „X” și „O” are rolul de a asigura o experiență competitivă pentru utilizator, permițând desfășurarea partidelor în regim single-player. Acest modul utilizează o combinație de strategii euristice și algoritmi de căutare pentru a determina mutarea optimă a agentului AI, adaptându-se atât la dimensiunea tablei de joc, cât și la complexitatea situației curente.

Inteligența artificială este proiectată să analizeze starea curentă a tablei de joc și să selecteze mutarea care maximizează șansele de câștig sau, în caz contrar, să prevină victoria adversarului.

```
function aiMove(ai='o'){
  if(gameOver) return;
  let key = findImmediate(matrix, ai);            // încearcă să câștige
  if(!key) key = findImmediate(matrix, opp(ai));  // încearcă să blocheze
  // ... (Minimax dacă nu există mutări imediate)
  play(key, ai);
}
```

Algoritmul principal folosit este Minimax cu optimizare alpha-beta, completat de verificări rapide pentru mutări imediate de câștig sau blocare. Acest hibrid asigură atât eficiență, cât și un nivel ridicat de competitivitate.

```
function findImmediate(b, p) {
  for (const k in b) {
    if (b[k] != null) continue;
    const n = { ...b, [k]: p };
    if (has3Win(n, p)) return k;
  }
  return null;
}
let key = findImmediate(matrix, ai);
if (!key) key = findImmediate(matrix, opp(ai));
```

În primul rând, agentul AI verifică dacă există o mutare care îi poate aduce victoria imediat. Acest lucru se realizează prin simularea fiecărei mutări posibile și evaluarea dacă, după plasarea piesei, se obține o linie câștigătoare. Dacă o astfel de mutare este identificată, agentul AI va executa fără a mai continua analiza. Dacă nu există o mutare câștigătoare, algoritmul verifică dacă adversarul are o oportunitate de a câștiga la următoarea mutare și, dacă este cazul, blochează această posibilitate.

```
function heuristic(b, me) {
  const foe = opp(me); let v = 0;
  for (const [[r1, c1], [r2, c2], [r3, c3]] of triples) {
    const a = b[`${r1},${c1}`], d = b[`${r2},${c2}`], e = b[`${r3},${c3}`];
    const mc = (a === me) + (d === me) + (e === me);
    const fc = (a === foe) + (d === foe) + (e === foe);
    if (mc && fc) continue;
    if (mc === 2) v += 40;
    else if (mc === 1) v += 5;
    else if (fc === 2) v -= 45;
    else if (fc === 1) v -= 6;
  }
  return v;
}
```

Dacă nu există mutări imediate de câștig sau de blocare, agentul AI evaluează poziția curentă folosind o funcție euristică. Aceasta analizează toate tripletele de celule de pe tablă și acordă punctaje pentru potențialul de câștig sau pericolul iminent. De exemplu, două simboluri proprii într-un triplet fără simboluri

adverse primesc un scor pozitiv, în timp ce două simboluri adverse primesc un scor negativ, reflectând necesitatea de a bloca acea linie.

Pentru a decide mutarea optimă în situațiile în care nu există mutări imediate, agentul AI folosește algoritmul Minimax cu tăiere alpha-beta. Acest algoritm explorează recursiv toate mutările posibile, alternând între rolul de maximizator (AI) și minimizator (adversar). La fiecare nivel al arborelui de decizie, algoritmul evaluează pozițiile folosind funcția euristică și propagă scorurile înapoi pentru a determina cea mai avantajoasă mutare. Tăierea alpha-beta optimizează procesul, eliminând ramurile care nu pot influența decizia finală, reducând astfel numărul de stări analizate și îmbunătățind performanța.

```
function alphaBeta(b, d, me, maxp, a, beta, nodes) {
  if (nodes.c++ >= MM_NODES) return 0;
  if (has3Win(b, me))          return  1000 - d;
  if (has3Win(b, opp(me)))     return -1000 + d;
  if (d >= MM_DEPTH || full(b))  return heuristic(b, me);
  if (maxp) {
    let best = -Infinity;
    for (const k in b) if (!b[k]) {
      const n = clone(b); n[k] = me;
      best = Math.max(best, alphaBeta(n, d + 1, me, false, a, beta, nodes));
      a = Math.max(a, best); if (beta <= a) break;
    }
    return best;
  } else {
    let best = Infinity, o = opp(me);
    for (const k in b) if (!b[k]) {
      const n = clone(b); n[k] = o;
      best = Math.min(best, alphaBeta(n, d + 1, me, true, a, beta, nodes));
      beta = Math.min(beta, best); if (beta <= a) break;
    }
    return best;
  }}
```

Pentru a preveni consumul excesiv de resurse, algoritmul limitează atât adâncimea maximă a explorării (MM_DEPTH), cât și numărul total de noduri analizate (MM_NODES). Acest lucru asigură un timp de răspuns rezonabil, chiar și pe table de dimensiuni mai mari.

```
const MM_DEPTH = 15, MM_NODES = 60000;

if (nodes.c++ >= MM_NODES) return 0;
if (d >= MM_DEPTH || full(b)) return heuristic(b, me);
```

2.2 Testarea sistemului

Testarea produsului constituie o etapă esențială în cadrul procesului de dezvoltare software, având ca scop principal asigurarea calității și funcționalității aplicației realizate. Prin intermediul testării, se urmărește identificarea și corectarea eventualelor erori, precum și verificarea conformității cu cerințele specificate în faza de proiectare. O aplicație testată corespunzător oferă utilizatorilor finali o experiență sigură, stabilă și eficientă, reducând riscul apariției problemelor în exploatare.

Procesul de testare presupune evaluarea sistematică a tuturor componentelor aplicației, atât la nivel individual, cât și integrat, utilizând metode variate precum testarea funcțională, testarea de integrare și testarea de acceptanță. Fiecare etapă de testare are rolul de a evidenția posibilele neconformități și de a valida modul de interacțiune dintre modulele sistemului. Rezultatele obținute în urma testării permit optimizarea performanței, creșterea gradului de utilizare și asigurarea unei experiențe pozitive pentru utilizatori, contribuind astfel la succesul produsului software pe termen lung.

Fig 2.1 Panoul de logare

Pagina de logare vă permite să vă autentificați în aplicație utilizând contul existent. Introduceți adresa de email asociată contului în câmpul "Email Address". Apoi, tastați parola în câmpul "Password". Confirmați datele apăsând butonul "Login". Pentru utilizatorii noi, secțiunea "Register" este disponibilă.

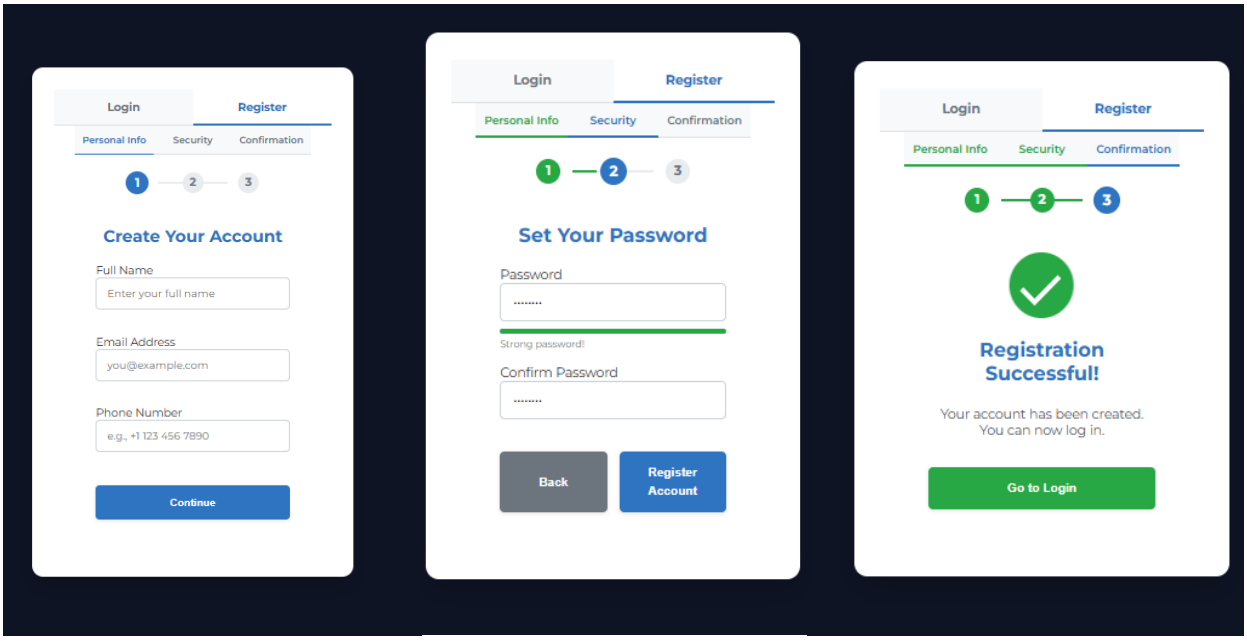

Fig 2.2 Panoul de înregistrare

Procesul de înregistrare se desfășoară în trei etape: Informații Personale, Securitate și Confirmare. Utilizatorii furnizează date de bază, setează o parolă sigură și confirmă crearea contului. Finalizarea cu succes este indicată printr-un mesaj de confirmare, iar utilizatorul p oate apoi accesa pagina de logare. Această metodă asigură o înregistrare organizată și sigură.

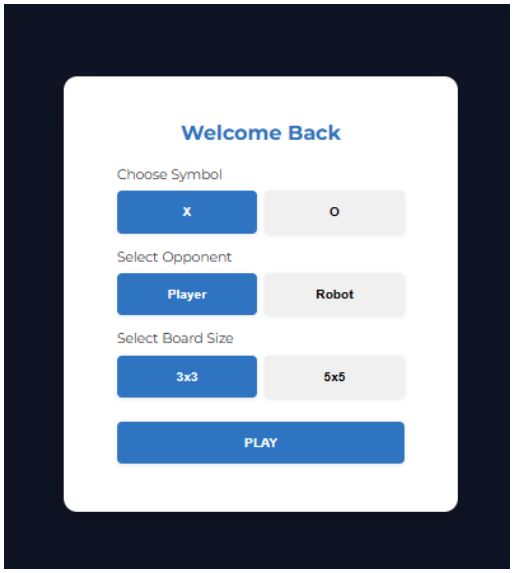

Fig 2.3 Panoul de meniu

După autentificare, utilizatorul este redirecționat către meniul principal al aplicației, unde poate selecta simbolul dorit, tipul de adversar și dimensiunea tablei de joc. În situația în care setările au fost deja configurate anterior, accesul se face direct în interfața de joc, permițând începerea imediată a unei noi partide. Astfel, experiența de utilizare este optimizată prin acces rapid la funcționalitățile principale ale aplicației.

Fig 2.4 Pagina de joc

Interfața jocului „Tic Tac Toe” este concepută pentru a fi intuitivă și ușor de utilizat. Tabla de joc este afișată central, iar scorul fiecărui jucător este vizibil în partea dreaptă. Pentru a începe o partidă, utilizatorul trebuie să facă primul click pe o celulă liberă de pe tablă; în cazul în care se joacă cu simbolul „O” împotriva agentul AI, primul click va iniția automat jocul și va permite plasarea simbolului ales. Fiecare jucător plasează alternativ simbolurile X sau O, iar scopul este de a forma o linie de trei simboluri identice pe orizontală, verticală sau diagonală. Designul minimalist și structura clară a elementelor asigură o experiență de joc plăcută și accesibilă.

Fig 2.5 Bara de navigare

Bara de navigare din partea superioară a aplicației oferă utilizatorului posibilitatea de a reveni rapid în meniul principal prin butonul „Menu” sau de a se deconecta din cont utilizând butonul „Log out”. Astfel, gestionarea sesiunii și navigarea între secțiunile aplicației se realizează într-un mod simplu și accesibil.

CONCLUZII

În cadrul stagiului de practică, a fost realizată aplicația „Tic Tac Toe”, un proiect ce a permis aplicarea cunoștințelor teoretice acumulate și dezvoltarea abilităților practice în domeniul programării și testării produselor software. Dezvoltarea acestei aplicații a urmărit atingerea obiectivelor stabilite, concentrându-se pe crearea unei experiențe de joc interactive și educative, care depășește implementările clasice ale jocului „X și Zero”.

Proiectul a implicat parcurgerea etapelor specifice ciclului de viață al dezvoltării software, începând cu analiza domeniului și a sistemelor similare, definirea cerințelor funcționale și non-funcționale, proiectarea arhitecturii și implementarea efectivă a modulelor. S-a utilizat motorul de joc Phaser.js pentru gestionarea graficii, animațiilor și interacțiunilor, demonstrând capacitatea acestuia de a facilita dezvoltarea rapidă a aplicațiilor web interactive.

Au fost implementate cu succes funcționalitățile de bază ale jocului, inclusiv gestionarea tablei de joc, alternarea turelor între jucători și validarea condițiilor de victorie sau remiză. O componentă importantă a proiectului a constat în diversificarea experienței de joc prin introducerea opțiunii de a alege dimensiunea grilei, 3x3 sau 5x5, și implementarea a două moduri de joc distincte: Player versus Player și Player versus AI.

Modulul de inteligență artificială reprezintă un punct central al realizării, fiind implementat prin utilizarea algoritmului Minimax cu optimizare alpha-beta. Această abordare asigură un nivel de dificultate competitiv și demonstrează aplicarea practică a algoritmilor de căutare și a tehnicilor euristice în dezvoltarea jocurilor. De asemenea, a fost integrat un sistem simplu de autentificare, utilizând localStorage, și un sistem de afișare a scorurilor, contribuind la rejucabilitatea aplicației.

Procesul de dezvoltare a permis consolidarea cunoștințelor referitoare la limbajul JavaScript, lucrul cu biblioteci externe precum Phaser.js, structurarea modulară a codului, gestionarea stării aplicației și implementarea interfețelor utilizator intuitive. Chiar dacă secțiunea de testare detaliată nu a fost completată în acest document, etapa de testare a fost o parte implicită a procesului de dezvoltare, asigurând funcționalitatea corectă a modulelor implementate.

În concluzie, realizarea aplicației „Tic Tac Toe” în cadrul stagiului de practică a reprezentat o experiență valoroasă, confirmând capacitatea de a transpune cerințe în soluții software funcționale și de a utiliza tehnologii moderne pentru dezvoltarea aplicațiilor web. Proiectul a îndeplinit obiectivele propuse, rezultând într-o aplicație interactivă și robustă, care servește atât ca instrument de divertisment, cât și ca exemplu practic al principiilor de programare și inteligență artificială.

BIBLIOGRAFIE

[1] Phaser Studio, "Phaser - Un cadru open source rapid, distractiv și gratuit pentru jocuri HTML5," [Online]. Disponibil: https://phaser.io/. Accesat: 9 Apr. 2025.

[2] W3Schools, "Tutorial HTML," [Online]. Disponibil: https://www.w3schools.com/html/. Accesat: 7 Apr. 2025.

[3] W3Schools, "Tutorial CSS," [Online]. Disponibil: https://www.w3schools.com/css/. Accesat: 10 Apr. 2025.

[4] Tictactoefree.com, "Reguli Joc X si 0 - Tic Tac Toe," [Online]. Disponibil: https://tictactoefree.com/ro/reguli. Accesat: 9 Apr. 2025.

[5] Wikipedia, "Minimax," [Online]. Disponibil: https://en.wikipedia.org/wiki/Minimax. Accesat: 15 Apr. 2025.

[6] Arcsys, "Tic Tac Toe Glow," [Online]. Disponibil: https://play.google.com/store/apps/details?id=com.arcsys.tictactoe&pli=1. Accesat: 10 Apr. 2025.

[7] Papergames.io, "Tic Tac Toe," [Online]. Disponibil: https://papergames.io/en/tic-tac-toe. Accesat: 10Apr. 2025.

[8] Google, "Tic Tac Toe," [Online]. Disponibil: https://www.google.com/search?q=Tic+Tac+Toe Accesat: 10 Apr. 2025.

ANEXA A

Configurarea jocului „X” și „O”

```
const config = {
type: Phaser.AUTO,
width: 600,
height: 600,
backgroundColor: '#0f1425',
parent: 'game-container',
scene: { preload, create, update }
};
const game = new Phaser.Game(config);
let cols = 3;
let rows = 3;
let WIN_LEN = 3;
let matrix = {};
let players = ['x', 'o'];
let scores = { x: 0, o: 0 };
let placedPieces = [];
let turn = true;  // true = X la mutare
let gameOver = false;

function preload() {
  this.load.image('ObjectX',
'http://127.0.0.1:5500/resources/images/ObjectX.svg');
  this.load.image('ObjectO',
'http://127.0.0.1:5500/resources/images/ObjectO.svg');
}
```

ANEXA B

Implementarea logică de joc „X” și „O”

```
  function TakeTurn(x, y, row, col) {
    if (gameOver) return;

    const key = `${row},${col}`;
    if (matrix[key] !== null) return;

    const sJSON = getCookie('GameSettings');
    if (!sJSON) { window.location.assign('http://localhost:5500/menu.html'); return;
}
    const s        = JSON.parse(sJSON);
    const vsAI     = s.PlayAgainstPlayerOr === 'off';
    const humanIsX = s.PlayAsXorO === 'on';
    const human    = humanIsX ? 'x' : 'o';
    const player   = turn ? 'x' : 'o';
    if (vsAI && player !== human) return;

    matrix[key] = player;
    const sprite = player === 'x' ? 'ObjectX' : 'ObjectO';
    const scale  = player === 'x' ? 0.7 : 0.5;

    const img = this.add.image(x, y, sprite).setScale(0);
    placedPieces.push(img);
    this.tweens.add({ targets: img, scaleX: scale, scaleY: scale,
                      ease: 'Back.easeOut', duration: 300 });

    turn = !turn;
    Check.call(this);

    const next = turn ? 'x' : 'o';
    if (vsAI && !gameOver && next !== human) aiMove(next);
  }
```

ANEXA D

Implementarea funcției „Check”

```
function Check() {
    const dirs = [[0,1],[1,0],[1,1],[1,-1]];
    for (const p of players)
      for (let r=0;r<rows;r++)
        for (let c=0;c<cols;c++) {
          if (matrix[`${r},${c}`] !== p) continue;
          for (const [dr,dc] of dirs) {
            const r2 = r + dr*(WIN_LEN-1);
            const c2 = c + dc*(WIN_LEN-1);
            if (r2<0||r2>=rows||c2<0||c2>=cols) continue;
            let ok = true;
            for (let k=1;k<WIN_LEN;k++)
              if (matrix[`${r+dr*k},${c+dc*k}`]!==p){ok=false;break;}
            if (ok){declareWin.call(this,p);return;}
          }
        }

    if (Object.values(matrix).every(v=>v!=null)) {
      displayWinner.call(this,"It's a draw!");
      console.log("It's a draw!"); gameOver=true;
    }
  }
```