Videopoker

Autor: Chirca Rares-Ciprian, 331AA

An: 3

Cuprins:

1.Introducere

Am ales acest proiect datorita faptului ca am vrut sa vad cum functioneaza un joc si cum se poate interactiona cu o interfata pentru a efectua diferite actiuni, iar jocul de poker este unul dintre preferatele mele. Aceasta aplicatie a fost o experienta noua pentru ca a trebuit sa combin notiuni pe care deja la dobandisem, cu elemente pe care a trebuit sa le invat pe parcurs si sa invat cum sa le combin,

In prima etapa, cea de documentare, pentru a putea realiza acest proiect a trebuit sa aleg ce limbaj de programare sa folosesc pentru partea de backend, iar datorita faptului ca imi doream sa invat Python[1], m-am gandit ca aceasta este o oportunitate perfecta. Pentru interconectare am folosit framework-ul Flask[2](scris tot in Python), iar pentru partea de frontend am folosit Javascript[3], HTML[4] si CSS[5]. Toate acestea au fost folosite cu ajutorul IDE-urile celor de la JetBrains PyCharm[6] si WebStorm[7].

Initial, a trebuit sa caut implementari aseamanatoare[8] pentru partea de backend pentru a intelege cum ar trebui sa functioneze acest joc, iar pentru partea de interfata m-am gandit cum mi-as dori sa arate design-ul si am incercat sa ajung cat mai aproape de acesta.

Am avut ca obiectiv principal realizarea unui joc functional de Videopoker, iar pentru a realiza acest lucru a trebuit sa implementez urmatoarele feature-uri.

- Crearea unui pachet de carti si a unui jucator caruia sa ii fie impartite cartile
- Implementarea functionalitatii de a castiga sau a pierde o mana in functie de cartile avute la final
- Implementarea unui sistem de puncte corelat cu subpunctul anterior
- Posibilitatea de a schimba un numar de carti date initial prin intermediul interfetei
- Posibilitatea de a juca o noua mana prin apasarea unui buton

2. Prezentare pe scurt a suportului tehnic, scurta trecere in revista a unor realizari similar sustinuta prin referinte bibliografice

Referitor la partea tehnica voi prezenta mai pe larg limbajul de programare folosit, framework-urile si IDE-ul. Programul reprezinta o aplicatie full-stack, imbinand logica jocului din backend cu functionalitatile oferite de frontend.

Python este un limbaj de programare high-level, interpretat, de uz general care are ca filosofie de proiectare lizibilitatea codului prin utilizarea unei identari semnificative. Acesta a fost proiectat de Guido van Rossum in anul 1991.

Flask este un framework web scris in Python, care suporta extensii care pot adauga caracteristici aplicatiei ca si cum acestea ar fi implementate pe Flask. Pinterest si Linkedin sunt bazate pe acest framework.

JavaScript este un limbaj de programare orientat pe obiect bazat pe conceptul prototipurilor. Este folosit in special pentru introducerea unor functionalitati in paginile web, fiind dat faptul ca browser-ul ruleaza codul JavaScript din aceste pagini. A fost initial dezvoltat de catre Brendan Eich, avand initial numele de Mocha, iar apoi LiveScript.

HTML este limbajul standard pentru documentele concepute cu scopul de a fi afisate intr-un browser web. HTML descrie structura unei pagini web din punct de vedere semantic și inițial a inclus indicii pentru aspectul documentului. Elementele HTML sunt elemente de baza ale paginilor, iar acestea sunt delimitate de etichete folosint paranteze unghiulare.

In ceea ce priveste mediul in care a fost scris codul, am folosit doua dintre IDE-urile celor de la JetBrains, primul fiind PyCharm, datorita faptului ca ofera o serie de functionalitati deja implementate care pot fi folosite mult mai usor si compatibilitate cu multe framework-uril existente, lucru care confora utilizatorului o usurinta in scrierea codului si in implementarea oricarei solutii. Acelasi lucru este valabil si pentru WebStorm, care este de un real ajutor in fluiditate si rapiditate in ceea ce priveste flow-ul de munca pentru partea de frontend.

Jocul de Videopoker[9] a fost prima oara inventat in cazionuri si este bazat pe un joc de poker cu cinci carti care pot fi schimbate maxim o data pentru a incerca sa realizezi o combinatie de carti care sa plateasca cat mai bine. Acesta era des jucat pe console computerizate in anii 1970-1980. Pentru fiecare mana noua jucata se mizeaza o suma, care poate fi pierduta in cazul unei combinatii de carti necastigatoare sau poate fi multiplicata in functie de cat de buna este mana jucatorului.

3. Prezentarea tehnica a etapei de realizare / implementare

Dupa ce am facut documentarea initiala, a trebuit sa aleg cum va arata interfata, insa nu am avut un model pentru ca implementarile asemanatoare erau folosite fara interfata, neavand aceasta parte de frontend.

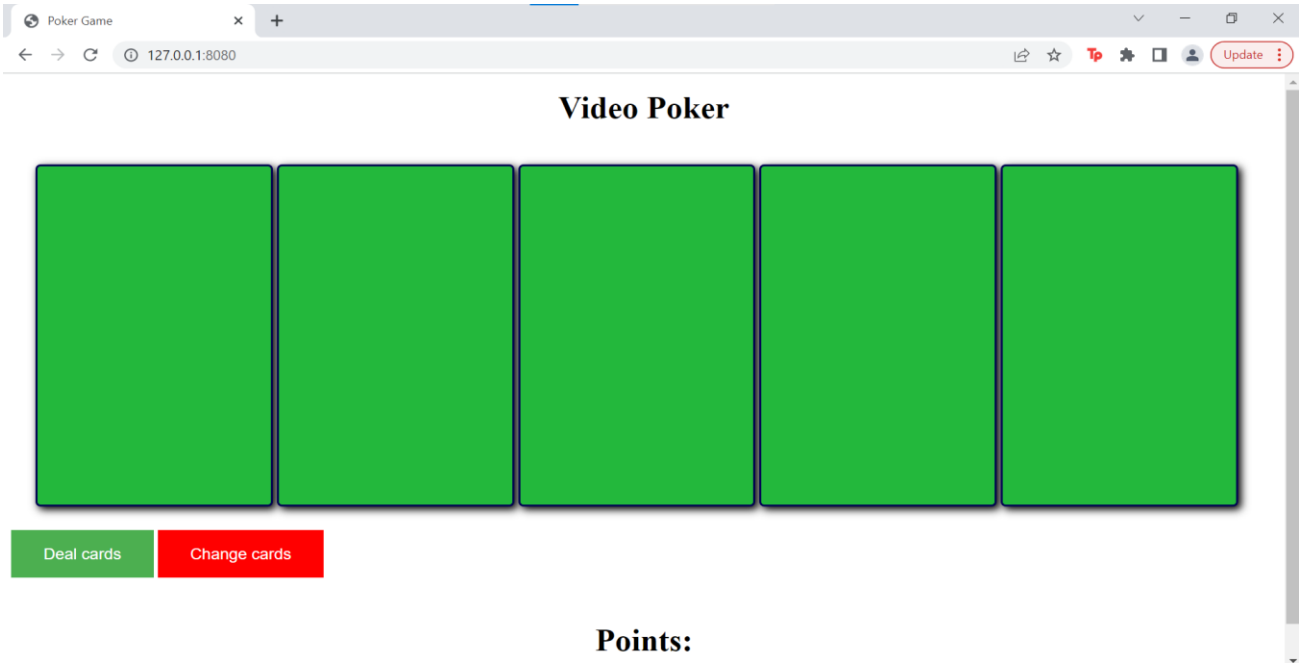

Fig. 1: Pagina initiala inainte de inceperea jocului

Dupa ce am vazut cum arata videopoker-ul implementat in realitate, am venit cu aceasta idee de interfata simplista, dar intuitiva. Pentru a putea fi accesibila, va trebui sa pornim modulele necesare pentru backend si frontend. Primul pas este pornirea backend-ului din terminal cu ajutorul comenzii “flask run”.

```
PS C:\Users\40763\PycharmProjects\PokerPrjct\backend> flask run
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

Fig. 2: Porinrea back-endului corespunzator aplicatiei

Privind partea de frontend, aceasta va fi pornita tot din terminal cu ajutorul comenzii “http-server”.

```
PS C:\Users\40763\PycharmProjects\PokerPrjct\frontend> http-server
Starting up http-server, serving ./

http-server version: 14.1.1

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
(Use `node --trace-deprecation ...` to show where the warning was created)
```

Fig. 3: Pornirea frontend-ului corespunzator aplicatiei

Arhitectura acestei parti de frontend a proiectului consta in impartirea in sectiuni specializate in realizarea unor tipuri diferite de task-uri. Astfel, vom avea un fisier de JavaScript(.js), un fisier HTML(.html) si unul CSS(.css), prin care vom putea defini design-ul, dar si modul in care se va comporta o componenta.

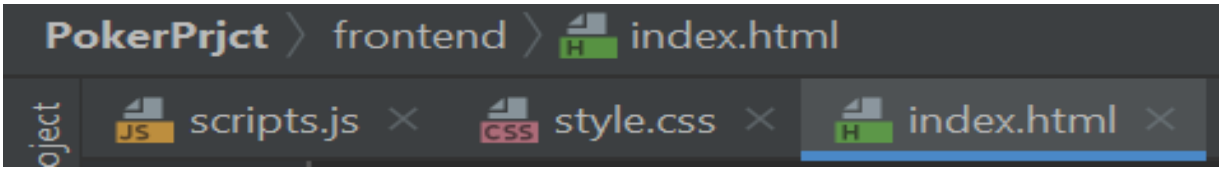

Fig. 4: Structura frontend a proiectului

Python dispune de un web framework numit Flask, care permite conectarea backend-ului cu frontend-ul, care prin ajutorul unor request-uri HTTP ne ajuta sa folosim datele trimise de user prin intermediul interfetei si sa le putem prelucra.

Pe pagina principala gasim cele cinci carti, construite ca niste butoane, care prin apasare isi vor schinba starea si infatisarea in carti marcate. Totodata, vor fi prezente si alte doua butoane: cel de “Deal cards” si cel de “Change cards”, alaturi de un paragraf care ne va prezenta intotdeauna numarul actual de puncte pe care il are jucatorul.

In cele ce urmeaza, voi expune tot procesul prin care se realizeaza jocul, mana cu mana. Initial, am creat fiecare carte dintre cele 52 avand un rang si un simbol. Aceste carti vor constitui pachetul de joc, care va fi amestecat inainte fiecarei noi maini si va ajuta la impartirea cartilor necesare. Vom avea totodata si o clasa numita Player, care va avea ca atribut mana sa actuala si anumite metode prin care se va adauga o carte sau se va arunca o carte din mana.

Cream un server local prin intermediul framework-ului Flask, prin care vom trimite la acea adresa cele cinci carti initiale care vor fi impartite jucatorului, dupa ce s-a apasat butonul de “Deal cards”.

```
@app.route("/cards/")
def deal_cards():
    nr_of_cards = int(request.args.get("number"))
    which_deal = request.args.get("deal")
```

Fig. 5: Modul prin care se impart cartile cu un request HTTP

Apoi asociem unui buton creat in HTML o functie in JavaScript care prin apasare se va duce la acest URL si va lua cartile de acolo, carti care au fost initial pasate ca un JSON. Ulterior, va trebui sa asociem fiecarui dintre cele cinci butoane cate o carte din acest JSON, care a fost impartit in functie si vom asocia fiecareia un fundal si culoarea scrisului. In aceasta pasam si doua argumente request-ului: cards=5 care ne spune ca avem nevoie sa luam doar primele cinci carti de deasupra pachetului si ca modul de impartire deal=first_deal pentru a putea seta punctele initiale la 1000.

Vom face un alt request HTTP care ne va pasa punctele actuale si in momentul in care apasam “Deal cards” ne va scadea un numar de puncte pe care il stabilim noi. Vom avea nevoie de aceasta functie si pentru a actualiza punctele la finalul mainii de joc, in cazul in care mana este castigatoare, sa adaugam punctele respective pentru acea mana.

```python
@app.route("/points")
def get_points():
    global POINTS
    mode = request.args.get("mode")
    hand_cost = 20

    if mode == "new_deal":
        POINTS -= hand_cost
    else:
        pass
    return str(POINTS)
```

Fig. 6: Modul prin care acutalizam punctele prin request HTTP

Aceste puncte vor trebui actualizate si in interfata, prin div-ul din HTML care va fi corespondent functiei din JavaScript specifica, care va face un fetch la URL-ul /points si va face display punctelor pe interfata.

```javascript
async function get_points(mode){
    const resp = await fetch( input: api_url + "points?mode=" + mode);
    let points = await resp.json();
    document.getElementById( elementId: "points").innerHTML = points.toString()
}
```

Fig.7 : Functia JavaScript prin care se actualizeaza punctele in interfata

Pentru a schimba cartile, jucatorul va trebui sa apese pe fiecare dintre carteile pe care vrea sa le schimbe, iar la final va apasa pe butonul “Change cards”. In momentul, in care se apasa pe o carte se cheama o functie, care ne apeleaza primul request HTTP (“/cards/”) cu parametrul cards egal cu numarul de carti care vrem sa fie schimbate. In acest fel se iau din acelasi pachet din care au fost imparite primele carti, urmatoarele x carti (x = numarul de carti schimbate). Totodata va fi pasata si pozitia cartii schimbate, pentru a sti ulterior cu ce pozitie vom inlocui cartea veche. In cazul in care nu vom vrea sa schimbam nicio carte, va fi pasat inapoi JSON-ul initial pentru interpretarea cartilor in vederea verificarii unui posibil castig, altfel va fi pasat JSON-ul cu cartile finale.

```html
<div class = "button-container">
    <button id="card0" type="button"  class="button-fancy" onclick="turn_card('card0', 'Card', 'blue', 'red')"></button>
    <button id="card1" type="button"  class="button-fancy" onclick="turn_card('card1', 'Card', 'blue', 'red')"></button>
    <button id="card2" type="button"  class="button-fancy" onclick="turn_card('card2', 'Card', 'blue', 'red')"></button>
    <button id="card3" type="button"  class="button-fancy" onclick="turn_card('card3', 'Card', 'blue', 'red')"></button>
    <button id="card4" type="button"  class="button-fancy" onclick="turn_card('card4', 'Card', 'blue', 'red')"></button>
</div>
```

Fig. 8: Butoanele HTML corespunzatoare cartilor

```
async function change_old_cards() {
    let changed_cards = []
    const btns = document.querySelectorAll( selectors: '.button-fancy')
    btns.forEach( callbackFn: function(btn : Element ){
        if(btn.style.backgroundColor === "blue")
            changed_cards += btn.id[4]
    })
    let final_cards = {}
    if(changed_cards.length) {
        for (let i = 0; i < changed_cards.length; i++) {
            const response = await fetch( input: api_url + "cards?number=1&deal=another_deal&pos=" + changed_cards[i]);
            let data = await response.json();
            if (i === changed_cards.length - 1) {
                final_cards = data;
            }
            document.getElementById( elementId: "card" + changed_cards[i]).innerHTML = data[changed_cards[i]];
            document.getElementById( elementId: "card" + changed_cards[i]).style.backgroundColor = "white";
            document.getElementById( elementId: "card" + changed_cards[i]).style.color = "black";
        }
    }else{
        const response = await fetch( input: api_url + "cards?number=1&pos=0");
        final_cards = await response.json();
    }
```

Fig. 9: Functie pentru schimbarea cartilor initiale

Dupa ce ni s-au intors cartile finale, va trebui sa verificam daca acestea reprezinta una dintre combinatiile castigatoare (perechi, Full House, chinta, culoare etc). In functie de rezultatul mainii, ni se va intoarce un mesaj care va aparea pe interfata (“Two Pairs”, “Better luck next time”, “Straight” etc).

```
@app.route("/win")
def winning_hand():
    final_cards = request.args.get("finalCards").split(',')
    msg = won_hand(final_cards)
    return msg
```

Fig. 10: Request HTTP care ne intoarce cartile finale si face POST la URL-ul /win cu mesajul corespunzator

```
    def flush(self):
        suits = []
        for card in self.cards:
            suits.append(card.suit)
        if len(set(suits)) == 1:
            return True
        return False

    def pairs(self):
        pairs = []
        vals = get_vals(self.cards)
        for value in vals:
            if vals.count(value) == 2 and value not in pairs:
                pairs.append(value)
        return pairs

    def four_of_a_kind(self):
        vals = get_vals(self.cards)
        for value in vals:
            if vals.count(value) == 4:
                return True
        return False
```

Fig. 11: Definirea unora dintre modurile de a castiga o mana de poker

4. Prezentare mod de utilizare, interactiune cu utilizatorul, configurare

Aplicatia are un mod intuitiv si simplu de a juca pentru fiecare jucator. Initial pe pagina vor aparea doar doua butoane: “Deal cards” si “Change cards”. Dupa ce vom apasa pe primul ne vor aparea cartile noastre.

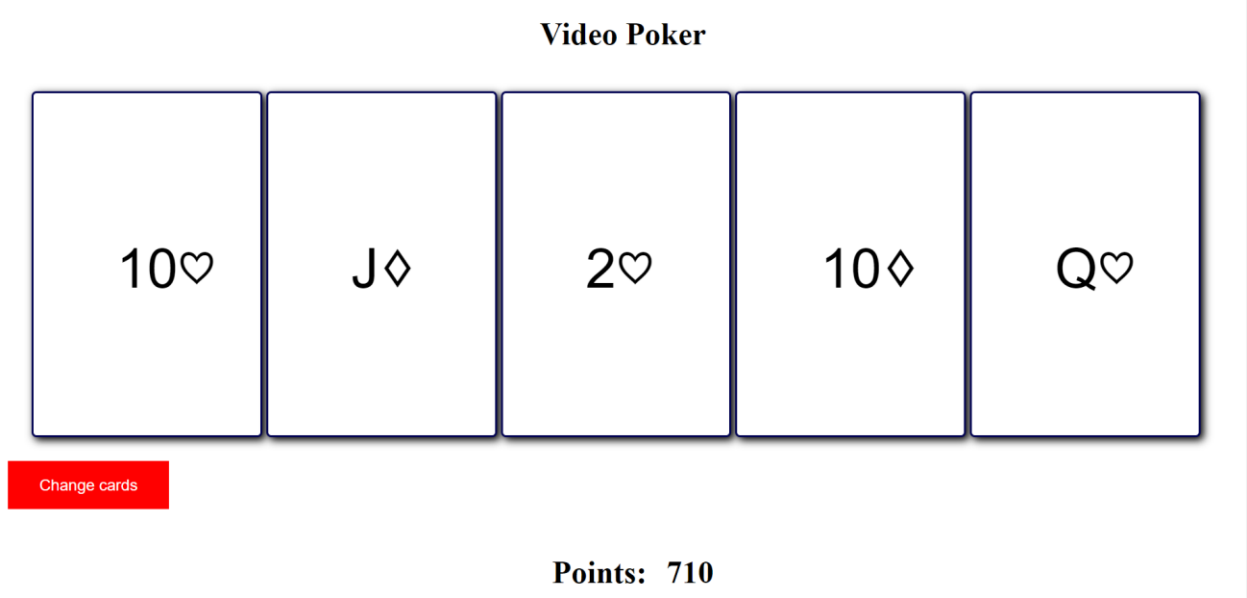

Fig. 12: Cartile oferite initial care pot fi schimbate

Pentru a face totul mai intuitiv, dupa ce s-au dat primele carti, singurul buton dispoibil va fi doar cel de “Change cards”, iar ulterior momentului schimbarii cartilor va fi disponibil doar cel de “Deal cards”. Schimbarea cartilor de joc se face efectiv prin apasarea pe cartea/cartile respective, iar apoi pe butonul rosu.

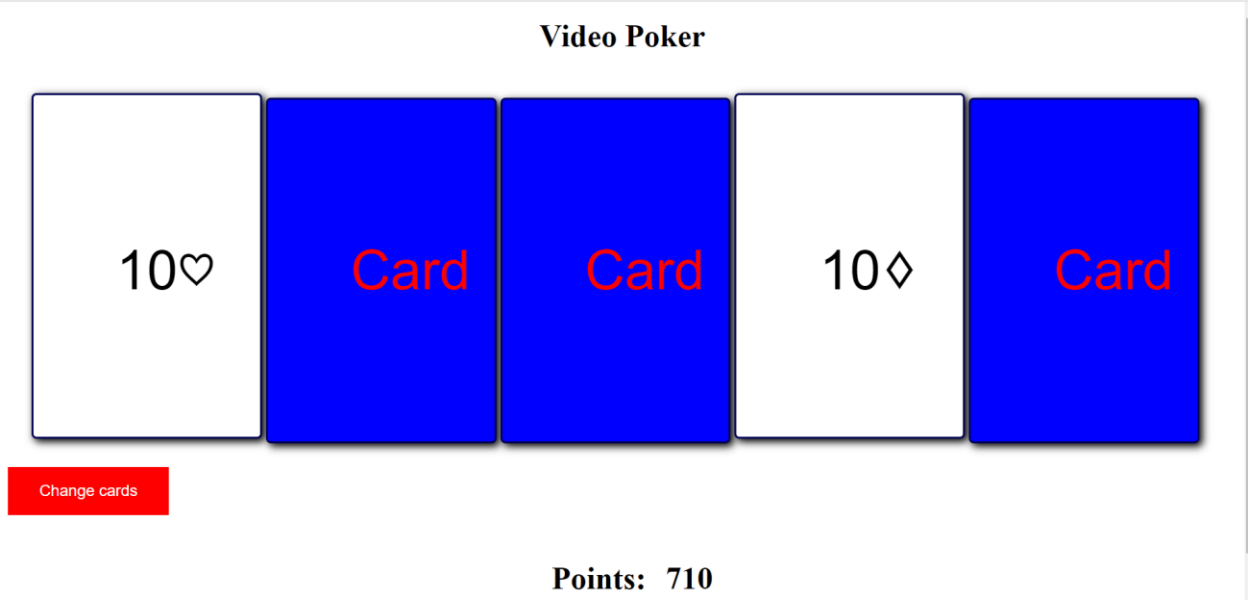

Fig. 13: Schimbarea cartilor pe care nu le dorim

Apoi, vom primi noile carti insotit de un mesaj care ne va spune daca am castigat, moment in care ne vor fi adaugate punctele corespunzatoare castigului respectiv, sau daca am pierdut, moment in care vom putea juca o noua mana. Acest lucru se poate repeta pana atat timp cat numarul punctelor este mai mare decat 0.

5. Concluzii

Aplicatia realizata are doar scop recreativ si simuleaza un joc real de videopoker care poate sa ii ofere jucatorului o experienta foarte asemanatoare cu cea reala, dar la un nivel putin mai simplist.

Din cauza faptului ca aplicatia este rulata intr-un mediu local, nu putem estima cu precizie performanta si rapiditatea cu care aceasta poate rula intr-un alt mediu, astfel acestea vor depinde foarte mult de setup-ul pe care il detine utilizatorul.

In concluzie, parerea mea este ca aplicatia este implementata intr-un mod placut si simplist reusind sa ofere jucatorului o experienta relaxanta si placuta. Aceasta dispune de toate componentele care se regasesc si intr-un joc real si de toate modalitatile de a castiga.

6. Referinte Bibliografice

[1] - https://pythonbasics.org/

[2] - https://flask.palletsprojects.com/en/2.2.x/

[3] – https://www.javascript.com/

[4] - https://en.wikipedia.org/wiki/HTML

[5] - https://en.wikipedia.org/wiki/CSS

[6] - https://www.jetbrains.com/pycharm/

[7] - https://www.jetbrains.com/webstorm/

[8] - https://github.com/annaymj/Python-Code/blob/master/Poker.py

[9] - https://en.wikipedia.org/wiki/Video_poker