Introdução: O desafio exigido foi para que entregarmos um jogo de BlackJack(21) em html, javascript e css para embelezar o site do jogo.Nosso jogo tem as funções de começar um novo jogo, comprar uma carta e parar o jogo, tendo opções visuais das cartas, o seu naipe e a quantidade de pontos do jogador e do dealer.

Implementação: Nosso jogo foi separado em vários códigos, index.html, styles.css, main.js e também fizemos a utilização de um arquivo favicon.png, farei a explicação de cada parte do código em seguida e no final terá o exemplo do mesmo:

index.html -

Nele criamos títulos com as tags h1 e h2, separamos as partes do site com as divs para que seja possível depois editá-las pelo css e criamos três botões que aparecem na página.

styles.css -

Aqui declaramos detalhes de como o estilo do site vai ficar, como sua margem, cores, fontes, tamanho de fontes, bordas, o espaçamento interno de elementos, centralização de textos, largura de onde os elementos irão aparecer na tela,etc.

main.js -

Agora já aqui na parte do script temos várias funções que fazem com que o jogo funcione, primeiro começamos declarando as arrays e as constantes a serem utilizadas no jogo, depois vimos com uma função para criar o deck, em seguida uma para usar fórmulas matemáticas para embaralhar posições das arrays, outra para pegar os valores das cartas, outra para calcular os pontos, depois uma para fazer com que as cartas do dealer fiquem escondidas até o player decidir parar o jogo ou exceder 21 pontos(apenas o primeiro valor do dealer deve aparecer), outra para atualizar o resultado do jogo, logo depois ele inicia um novo jogo e zera a mão dos jogadores, embaralha o deck, distribui as cartas e calcula os pontos, depois ele adiciona uma função para o botão de comprar cartas, que o jogador ao clicar no mesmo e lhe entregado uma carta e calculado os seus pontos e se passar de 21 exibe uma mensagem dizendo 'Você perdeu! Sua pontuação excedeu 21.', agora temos a função do botão parar que faz com que o dealer compre cartas até ter pelo menos 17 pontos, e ao clicado ele revela as cartas e verifica quem ganhou e por fim temos a desativação dos botões para impedir que o jogador continue jogando depois que o jogo termina e também faz as interligações das funções hit e stand aos botões visuais em html “Comprar cartas” , “Parar” e “Novo Jogo”.

Depois de codificar eu adicionei e linkei um favicon para que seja exibido ao lado do nome do site, no qual é essa:

Conclusão: As maiores dificuldades que eu encontrei ao fazer esse trabalho foi em algumas partes do código, bastante na parte do css pois ele não estava querendo atualizar

o estilo do site, tive que olhar com bastante detalhe até achar o que estava errado nos códigos, que era nessa linha “<link rel="stylesheet" href="styles.css">”, invés de stylesheet estava escrito stylessheet com dois S dando conflito na hora do código ler o css, mas em um geral foi apenas isso mesmo e o eu tirei com isso foi que achei bastante divertido as integrações das funções com a criação do html e achei bastante divertido mexer no css mas especificamente nas cores, fontes e bordas de elementos internos.

Bibliografia: ChatGPT(usado para desenvolver o código e explicar detalhes do mesmo podendo possibilitar o estudo e aprendizado dos assuntos discutidos) e Google para pesquisas de significados de termos específicos.

index.html -

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jogo de Blackjack</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="shortcut icon" type="image/png" href="favicon.png">
</head>
<body>
    <div class="game-container">
        <h1>🃏 Jogo de Blackjack</h1>

        <div class="hand" id="player-hand">
            <h2>Cartas do Jogador</h2>
            <div id="player-cards" class="cards"></div>
            <p>Pontos: <span id="player-points">0</span></p>
        </div>

        <div class="hand" id="dealer-hand">
            <h2>Cartas do Dealer</h2>
            <div id="dealer-cards" class="cards"></div>
            <p>Pontos: <span id="dealer-points">0</span></p>
        </div>

        <div id="controls">
            <button id="hit">🂠 Comprar Carta</button>
            <button id="stand">🛑 Parar</button>
            <button id="new-game">🔄 Novo Jogo</button>
        </div>

        <div id="result">
            <h2 id="game-result"></h2>
```

```
        </div>
    </div>

    <script src="main.js"></script>
</body>
</html>
```

styles.css -

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #1f1f1f;
    color: #fff;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.game-container {
    background-color: #2e2e2e;
    padding: 30px;
    border-radius: 16px;
    width: 90%;
    max-width: 500px;
    text-align: center;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.6);
}

h1 {
    margin-bottom: 30px;
    font-size: 2rem;
    color: #1edd0c;
}

h2 {
    margin-bottom: 10px;
    color: #ea1cf1;
}

.hand {
    margin-bottom: 25px;
}

.cards {
```

```
    font-size: 1.5rem;
    margin: 10px 0;
    min-height: 40px;
    color: #e60e0e;
}

#controls {
    margin: 20px 0;
    display: flex;
    justify-content: center;
    gap: 10px;
    flex-wrap: wrap;
}

button {
    padding: 10px 20px;
    background-color: #4caf50;
    border: none;
    color: white;
    font-size: 1rem;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #43a047;
}

#result {
    margin-top: 20px;
    font-size: 1.2rem;
    font-weight: bold;
    color: #ffd700;
}
```

main.js -

```
let deck = [];
let playerHand = [];
let dealerHand = [];
let playerPoints = 0;
let dealerPoints = 0;

const suits = ['♠', '♥', '♦', '♣'];
const values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];
```

```
const playerCardsDiv = document.getElementById('player-cards');
const dealerCardsDiv = document.getElementById('dealer-cards');
const playerPointsSpan = document.getElementById('player-points');
const dealerPointsSpan = document.getElementById('dealer-points');
const gameResult = document.getElementById('game-result');

const hitButton = document.getElementById('hit');
const standButton = document.getElementById('stand');
const newGameButton = document.getElementById('new-game');

function createDeck() {
    deck = [];
    for (let suit of suits) {
        for (let value of values) {
            deck.push({ value, suit });
        }
    }
    deck = shuffle(deck);
}

function shuffle(deck) {
    for (let i = deck.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [deck[i], deck[j]] = [deck[j], deck[i]];
    }
    return deck;
}

function getCardValue(card) {
    if (card.value === 'A') return 11;
    if (['K', 'Q', 'J'].includes(card.value)) return 10;
    return parseInt(card.value);
}

function calculatePoints(hand) {
    let points = hand.reduce((sum, card) => sum + getCardValue(card), 0);
    let aceCount = hand.filter(card => card.value === 'A').length;

    while (points > 21 && aceCount > 0) {
        points -= 10;
        aceCount--;
    }

    return points;
}

function displayCards(revealDealer = false) {
```

```
    playerCardsDiv.innerHTML = playerHand.map(card => `${card.value}${card.suit}`).join(' ');

    if (!revealDealer) {
        dealerCardsDiv.innerHTML = `${dealerHand[0].value}${dealerHand[0].suit} 🂠`;
        dealerPointsSpan.textContent = '?';
    } else {
        dealerCardsDiv.innerHTML = dealerHand.map(card => `${card.value}${card.suit}`).join('
');
        dealerPointsSpan.textContent = dealerPoints;
    }

    playerPointsSpan.textContent = playerPoints;
}

function updateGameResult(result) {
    gameResult.textContent = result;
}

function startNewGame() {
    playerHand = [];
    dealerHand = [];
    createDeck();

    playerHand.push(deck.pop(), deck.pop());
    dealerHand.push(deck.pop(), deck.pop());

    playerPoints = calculatePoints(playerHand);
    dealerPoints = calculatePoints(dealerHand);

    displayCards();
    updateGameResult('');
}

function hit() {
    playerHand.push(deck.pop());
    playerPoints = calculatePoints(playerHand);
    displayCards();

    if (playerPoints > 21) {
        displayCards(true);
        updateGameResult('Você perdeu! Sua pontuação excedeu 21.');
        disableButtons();
    }
}

function stand() {
    while (dealerPoints < 17) {
        dealerHand.push(deck.pop());
```

```
        dealerPoints = calculatePoints(dealerHand);
    }

    displayCards(true);

    if (dealerPoints > 21) {
        updateGameResult('O dealer perdeu! Ele excedeu 21.');
    } else if (playerPoints > dealerPoints) {
        updateGameResult('Você venceu!');
    } else if (playerPoints < dealerPoints) {
        updateGameResult('Você perdeu! O dealer venceu.');
    } else {
        updateGameResult('Empate!');
    }
    disableButtons();
}

function disableButtons() {
    hitButton.disabled = true;
    standButton.disabled = true;
}

hitButton.addEventListener('click', hit);
standButton.addEventListener('click', stand);
newGameButton.addEventListener('click', () => {
    startNewGame();
    hitButton.disabled = false;
    standButton.disabled = false;
});

startNewGame();
```