# **Practical – 1**

**Aim :**

1. Write a JS arrow function to find missing number in an Array.
2. Write a JS arrow function to find second largest and second smallest number from given array.
3. **Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

var arr = [1, 2, 4, 5, 6, 8, 10];

const findMissingElement = (array) => {

for (let i = 0; i < array.length; i++) {

if (array[i] !== i + 1) {

return i + 1;

}

}

return null;

};

console.log(arr);

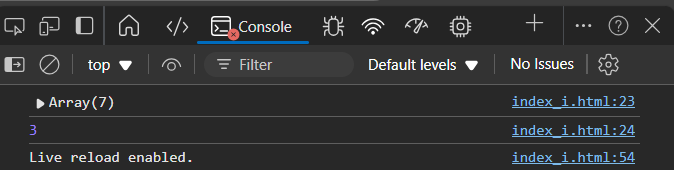
console.log(findMissingElement(arr));

</script>

</body>

</html>

**Output:**

  
***Figure I:*** *Missing number in Array*

1. **Code:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        var sort = (arr) => {

            for (var i = 0; i < arr.length; i++) {

                var min = i;

                for (var j = i + 1; j < arr.length; j++) {

                    if (arr[j] < arr[min]) {

                        min = j;

                    }

                }

                if (arr[min] < arr[i]) {

                    var temp = arr[min];

                    arr[min] = arr[i];

                    arr[i] = temp;

                }

            }

        };

        var array = [10, 8, 3, 6, 5, 1, 2];

        console.log(`Unsorted Array: ${array}`);

        sort(array);

        console.log(`Sorted Array: ${array}`);

        if (array.length >= 2) {

            console.log(`2nd Smallest Element: ${array[1]}`);

            console.log(`2nd Largest Element: ${array[array.length - 2]}`);

        } else {

            console.log("Array does not have enough elements to determine the 2nd smallest and 2nd largest.");

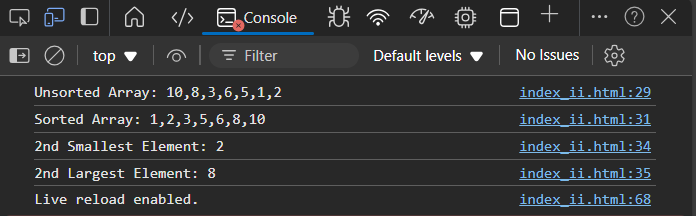
        }

    </script>

</body>

</html>

**Output:**

  
***Figure II:*** *Second largest and second smallest number from given array.*

# **Practical – 2**

**Aim :** Build a “Guess the Number Game” using JavaScript.

**Code:**

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Number Guessing Game</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="game-container">

        <div class="game-header">

            <svg class="icon target-icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"

                fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

                <circle cx="12" cy="12" r="10" />

                <circle cx="12" cy="12" r="6" />

                <circle cx="12" cy="12" r="2" />

            </svg>

            <h1>Number Guessing Game</h1>

        </div>

        <div class="game-content">

            <div class="game-stats">

                <div class="timer">

                    <svg class="icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"

                        fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"

                        stroke-linejoin="round">

                        <circle cx="12" cy="12" r="10" />

                        <polyline points="12 6 12 12 16 14" />

                    </svg>

                    <span id="timer">30s</span>

                </div>

                <div class="attempts">

                    Attempts: <span id="attempts">5</span>/5

                </div>

            </div>

            <div id="feedback" class="feedback">

                Guess a number between 1 and 100!

            </div>

            <div class="input-group">

                <input type="number" id="guess" placeholder="Enter your guess" min="1" max="100">

            </div>

            <div class="button-group">

                <button id="submit" class="btn btn-primary">

                    <svg class="icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"

                        fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"

                        stroke-linejoin="round">

                        <circle cx="12" cy="12" r="10" />

                        <circle cx="12" cy="12" r="6" />

                        <circle cx="12" cy="12" r="2" />

                    </svg>

                    Submit Guess

                </button>

                <div class="secondary-buttons">

                    <button id="reset" class="btn btn-success hidden">

                        <svg class="icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"

                            fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"

                            stroke-linejoin="round">

                            <path

                                d="M21.5 2v6h-6M2.5 22v-6h6M2 11.5a10 10 0 0 1 18.8-4.3M22 12.5a10 10 0 0 1-18.8 4.3" />

                        </svg>

                        New Game

                    </button>

                    <button id="quit" class="btn btn-danger">

                        <svg class="icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24"

                            fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round"

                            stroke-linejoin="round">

                            <circle cx="12" cy="12" r="10" />

                            <path d="m15 9-6 6" />

                            <path d="m9 9 6 6" />

                        </svg>

                        Quit

                    </button>

                </div>

            </div>

        </div>

        <div id="winner" class="winner hidden">

            <svg class="icon" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none"

                stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">

                <path d="M8.21 13.89 7 23l5-3 5 3-1.21-9.11" />

                <path d="M15 7a2 2 0 1 0 0-4 2 2 0 0 0 0 4Z" />

                <path d="M9 7a2 2 0 1 0 0-4 2 2 0 0 0 0 4Z" />

                <path d="M12 7h0" />

                <path d="M12 7h0" />

            </svg>

            <span>Winner!</span>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

**Style.css**

/\* Base styles \*/

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: system-ui, -apple-system, sans-serif;

    min-height: 100vh;

    display: flex;

    align-items: center;

    justify-content: center;

    background: linear-gradient(135deg, #e0e7ff 0%, #f3e8ff 100%);

    padding: 1rem;

}

/\* Game container \*/

.game-container {

    background: white;

    border-radius: 1rem;

    box-shadow: 0 10px 25px -5px rgba(0, 0, 0, 0.1);

    width: 100%;

    max-width: 28rem;

    padding: 2rem;

}

/\* Header \*/

.game-header {

    display: flex;

    align-items: center;

    justify-content: center;

    margin-bottom: 1.5rem;

}

.game-header h1 {

    font-size: 1.5rem;

    font-weight: bold;

    color: #1f2937;

    margin-left: 0.5rem;

}

/\* Game content \*/

.game-content {

    display: flex;

    flex-direction: column;

    gap: 1.5rem;

}

/\* Game stats \*/

.game-stats {

    display: flex;

    justify-content: space-between;

    align-items: center;

}

.timer {

    display: flex;

    align-items: center;

    color: #f97316;

    font-weight: 600;

}

.attempts {

    color: #7c3aed;

    font-weight: 600;

}

/\* Feedback message \*/

.feedback {

    padding: 1rem;

    border-radius: 0.5rem;

    text-align: center;

    color: white;

    background-color: #3b82f6;

    transition: background-color 0.3s ease;

}

.feedback.success {

    background-color: #22c55e;

}

.feedback.error {

    background-color: #ef4444;

}

/\* Input group \*/

.input-group {

    margin-bottom: 1rem;

}

input[type="number"] {

    width: 100%;

    padding: 0.75rem 1rem;

    border: 1px solid #d1d5db;

    border-radius: 0.5rem;

    font-size: 1rem;

    transition: all 0.2s ease;

}

input[type="number"]:focus {

    outline: none;

    border-color: #6366f1;

    box-shadow: 0 0 0 3px rgba(99, 102, 241, 0.2);

}

input[type="number"]:disabled {

    background-color: #f3f4f6;

    cursor: not-allowed;

}

/\* Buttons \*/

.button-group {

    display: flex;

    flex-direction: column;

    gap: 1rem;

}

.secondary-buttons {

    display: grid;

    grid-template-columns: 1fr 1fr;

    gap: 1rem;

}

.btn {

    display: flex;

    align-items: center;

    justify-content: center;

    padding: 0.75rem 1rem;

    border: none;

    border-radius: 0.5rem;

    font-size: 1rem;

    font-weight: 500;

    cursor: pointer;

    transition: background-color 0.2s ease;

}

.btn:disabled {

    opacity: 0.5;

    cursor: not-allowed;

}

.btn .icon {

    margin-right: 0.5rem;

}

.btn-primary {

    background-color: #4f46e5;

    color: white;

}

.btn-primary:hover:not(:disabled) {

    background-color: #4338ca;

}

.btn-success {

    background-color: #22c55e;

    color: white;

}

.btn-success:hover:not(:disabled) {

    background-color: #16a34a;

}

.btn-danger {

    background-color: #ef4444;

    color: white;

}

.btn-danger:hover:not(:disabled) {

    background-color: #dc2626;

}

/\* Icons \*/

.icon {

    width: 1.25rem;

    height: 1.25rem;

}

.target-icon {

    width: 2rem;

    height: 2rem;

    color: #4f46e5;

}

/\* Winner section \*/

.winner {

    display: flex;

    align-items: center;

    justify-content: center;

    margin-top: 1.5rem;

    color: #22c55e;

    font-size: 1.25rem;

    font-weight: 600;

}

.winner .icon {

    margin-right: 0.5rem;

    width: 2rem;

    height: 2rem;

}

/\* Utility classes \*/

.hidden { display: none; }

**Script.js**

class NumberGuessingGame {

    constructor() {

        this.randomNumber = 0;

        this.attempts = 5;

        this.timeLeft = 30;

        this.gameStatus = 'playing'; // 'playing', 'won', 'lost'

        this.timer = null;

        // DOM elements

        this.elements = {

            guess: document.getElementById('guess'),

            feedback: document.getElementById('feedback'),

            timer: document.getElementById('timer'),

            attempts: document.getElementById('attempts'),

            submit: document.getElementById('submit'),

            reset: document.getElementById('reset'),

            quit: document.getElementById('quit'),

            winner: document.getElementById('winner')

        };

        // Bind event listeners

        this.elements.submit.addEventListener('click', () => this.handleGuess());

        this.elements.reset.addEventListener('click', () => this.resetGame());

        this.elements.quit.addEventListener('click', () => this.quitGame());

        this.elements.guess.addEventListener('keypress', (e) => {

            if (e.key === 'Enter') this.handleGuess();

        });

        // Initialize game

        this.resetGame();

    }

    generateRandomNumber() {

        // Generate a whole number between 1 and 100

        const number = Math.floor(Math.random() \* 100) + 1;

        this.randomNumber = number;

        // Log both the raw calculation and the final number

        console.log('Random Number Generation:');

        console.log('Raw calculation:', Math.random() \* 100);

        console.log('Final whole number:', number);

    }

    startTimer() {

        clearInterval(this.timer);

        this.timeLeft = 30;

        this.updateTimerDisplay();

        this.timer = setInterval(() => {

            this.timeLeft--;

            this.updateTimerDisplay();

            if (this.timeLeft <= 0) {

                this.endGame('lost', "Time's up! Game over!");

            }

        }, 1000);

    }

    updateTimerDisplay() {

        this.elements.timer.textContent = `${this.timeLeft}s`;

    }

    updateFeedback(message, status = '') {

        this.elements.feedback.textContent = message;

        this.elements.feedback.className = 'feedback ' + status;

    }

    handleGuess() {

        const guess = this.elements.guess.value.trim(); // Remove extra spaces

        // Check if the guess is a valid number and within the range

        if (!/^\d+$/.test(guess) || parseInt(guess) < 1 || parseInt(guess) > 100) {

            this.updateFeedback('Please enter a valid number between 1 and 100.', 'error');

            return;

        }

        // Convert the guess to a number

        const guessNum = parseInt(guess);

        this.attempts--;

        this.elements.attempts.textContent = this.attempts;

        // Log the comparison

        console.log('Current guess:', guessNum);

        console.log('Target number:', this.randomNumber);

        if (guessNum === this.randomNumber) {

            this.endGame('won', `Congratulations! You guessed it in ${5 - this.attempts} attempts!`);

            return;

        }

        if (this.attempts <= 0) {

            this.endGame('lost', `Game over! The number was ${this.randomNumber}.`);

            return;

        }

        // Show reset button after the first invalid guess

        if (this.attempts < 5) {

            this.elements.reset.classList.remove('hidden');

        }

        this.updateFeedback(

            guessNum < this.randomNumber ? 'Your guess is too low.' : 'Your guess is too high.',

            'error'

        );

    }

    endGame(status, message) {

        this.gameStatus = status;

        clearInterval(this.timer);

        this.elements.guess.disabled = true;

        this.elements.submit.disabled = true;

        this.elements.reset.classList.remove('hidden');

        if (status === 'won') {

            this.elements.winner.classList.remove('hidden');

            this.updateFeedback(message, 'success');

        } else {

            this.updateFeedback(message, 'error');

        }

    }

    resetGame() {

        this.generateRandomNumber();

        this.attempts = 5;

        this.gameStatus = 'playing';

        // Reset DOM

        this.elements.attempts.textContent = this.attempts;

        this.elements.guess.value = '';

        this.elements.guess.disabled = false;

        this.elements.submit.disabled = false;

        this.elements.reset.classList.add('hidden');

        this.elements.winner.classList.add('hidden');

        this.updateFeedback('Guess a number between 1 and 100!');

        // Start timer

        this.startTimer();

    }

    quitGame() {

        this.endGame('lost', 'You quit the game!');

    }

}

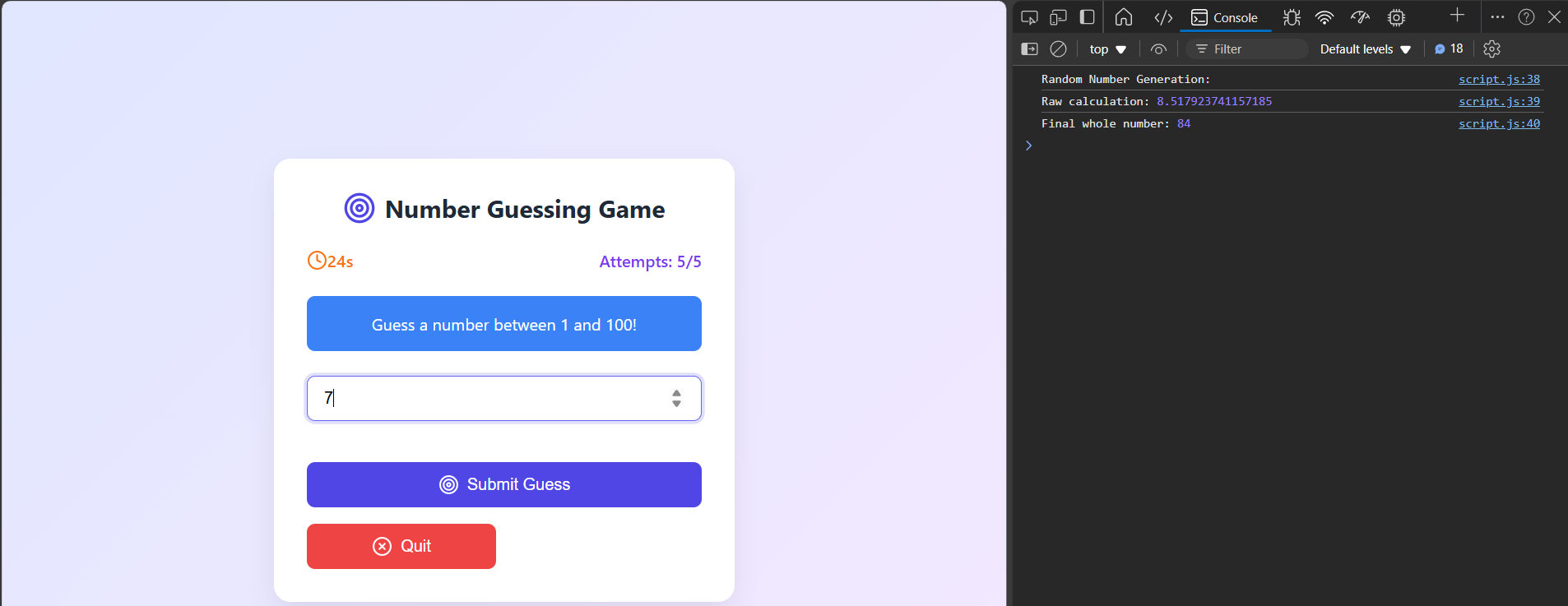
// Initialize game when DOM is loaded

document.addEventListener('DOMContentLoaded', () => {

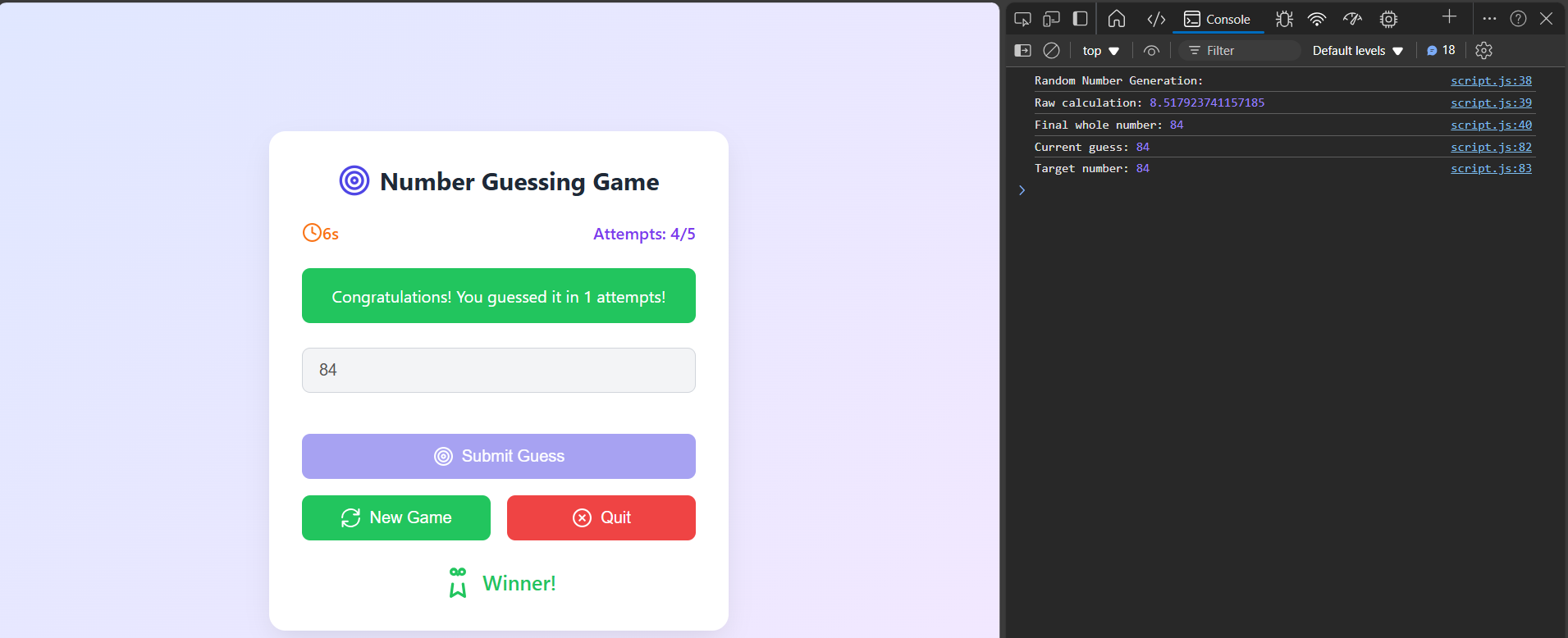
    new NumberGuessingGame();

});

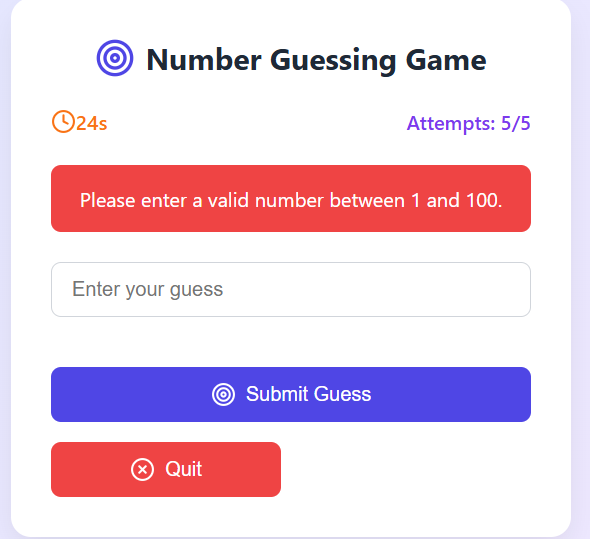
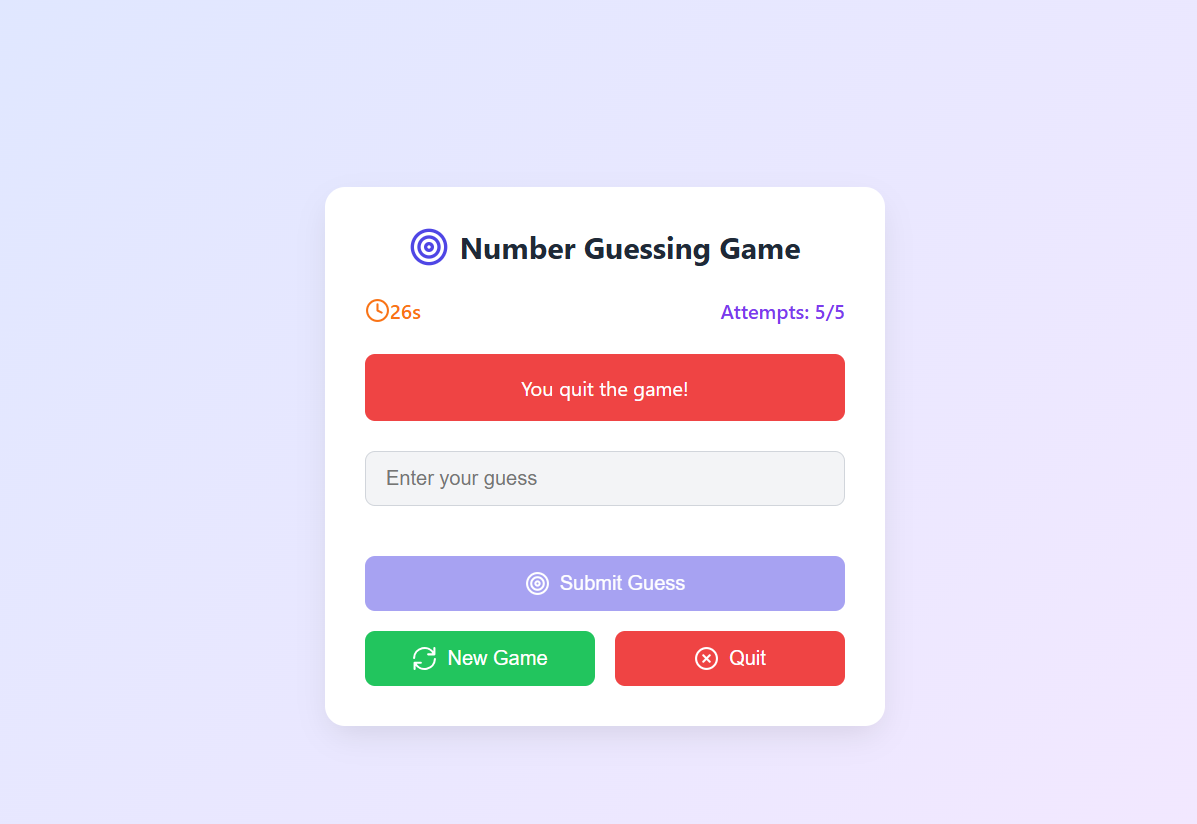
**Output:**



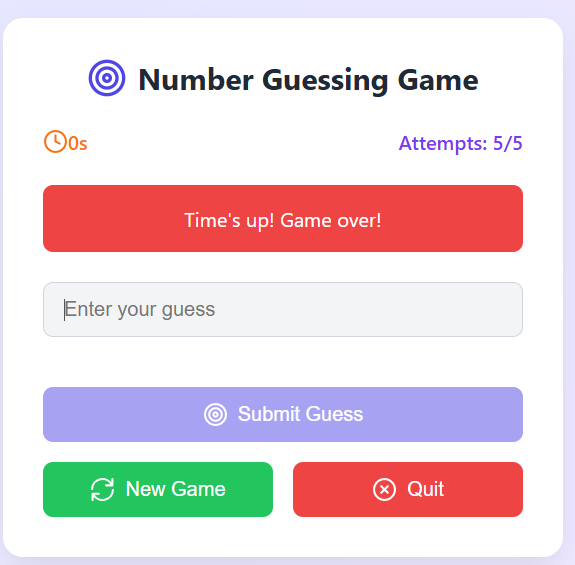
***Figure*** *I*



***Figure*** *II*

******

***Figure*** *III* ***Figure*** *IV*

******   
 ***Figure*** *V* ***Figure*** *VI*