**Music Streaming Queue:**

**Develop a dynamic playlist system where songs are represented as nodes in a circular doubly linked list. The user can queue songs, skip, replay, and move to the next song seamlessly. Ensure the playlist can handle thousands of songs with minimal memory overhead.**

<html lang="en">

<head>

<title>Music Streaming Queue</title>

<style>

/\* Basic styling \*/

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background: #f3f3f3;

margin: 0;

}

#app {

text-align: center;

width: 300px;

background: #fff;

padding: 20px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

border-radius: 10px;

}

h2 {

margin-bottom: 10px;

}

.song-info {

font-weight: bold;

margin: 10px 0;

}

.buttons button {

padding: 10px;

margin: 5px;

border: none;

border-radius: 5px;

cursor: pointer;

background-color: #007bff;

color: white;

}

.buttons button:disabled {

background-color: #d3d3d3;

cursor: not-allowed;

}

input[type="text"] {

padding: 8px;

width: 80%;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

</style>

</head>

<body>

<div id="app">

<h2>Music Streaming Queue</h2>

<input type="text" id="songInput" placeholder="Enter song name">

<button onclick="queueSong()">Add to Queue</button>

<div class="song-info" id="currentSong">No song playing</div>

<div class="buttons">

<button onclick="prevSong()">Previous</button>

<button onclick="replaySong()">Replay</button>

<button onclick="nextSong()">Next</button>

</div>

</div>

<script>

class SongNode {

constructor(name) {

this.name = name;

this.next = null;

this.prev = null;

}

}

class Playlist {

constructor() {

this.current = null;

this.size = 0;

}

// Add a song to the playlist

queueSong(name) {

const newNode = new SongNode(name);

if (!this.current) {

this.current = newNode;

this.current.next = newNode;

this.current.prev = newNode;

} else {

const lastNode = this.current.prev;

lastNode.next = newNode;

newNode.prev = lastNode;

newNode.next = this.current;

this.current.prev = newNode;

}

this.size++;

this.displayCurrentSong();

}

// Go to the next song

nextSong() {

if (this.current) {

this.current = this.current.next;

this.displayCurrentSong();

}

}

// Replay the current song

replaySong() {

this.displayCurrentSong();

}

// Go to the previous song

prevSong() {

if (this.current) {

this.current = this.current.prev;

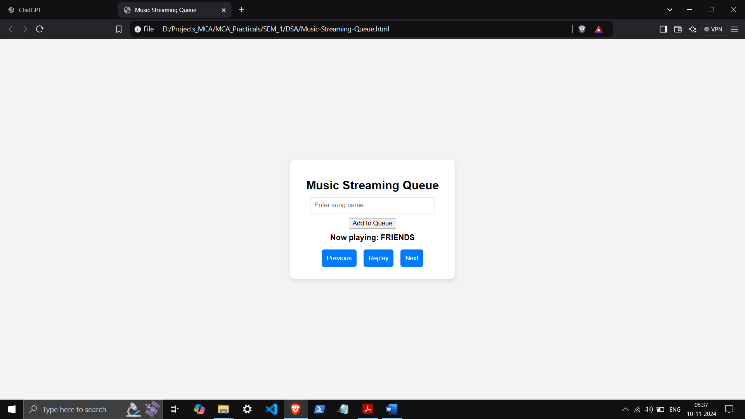
this.displayCurrentSong();

}

}

// Display the current song

displayCurrentSong() {

 const currentSongElement = document.getElementById("currentSong");

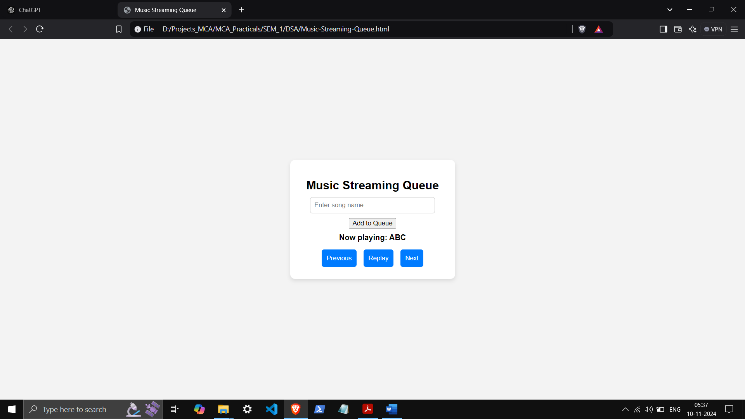
currentSongElement.innerText = this.current ? `Now playing: ${this.current.name}` : "No song playing";

}

}

const playlist = new Playlist();

// Add song to queue

 function queueSong() {

const songInput = document.getElementById("songInput");

const songName = songInput.value.trim();

if (songName) {

playlist.queueSong(songName);

songInput.value = "";

} else {

alert("Please enter a song name.");

}

}

// Button functions

function nextSong() {

playlist.nextSong();

}

function replaySong() {

playlist.replaySong();

}

function prevSong() {

playlist.prevSong();

}

</script>

</body>

</html>

