**Real-Time Collaborative Editor:**

**Build a version control system for a real-time collaborative text editor using a doubly linked list. Each edit is a node, allowing users to navigate forwards and backwards through changes. Include an undo/redo feature, tracking the position of multiple collaborators.**

<html lang="en">

<head>

<title>Collaborative Editor with Version Control</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f4f4f9;

}

.editor-container {

width: 80%;

max-width: 600px;

}

#editor {

width: 100%;

height: 200px;

padding: 10px;

font-size: 16px;

border: 1px solid #ccc;

resize: none;

}

.controls {

margin-top: 10px;

display: flex;

justify-content: space-between;

}

button {

padding: 8px 12px;

font-size: 14px;

cursor: pointer;

border: 1px solid #ccc;

background-color: #ffffff;

}

button:disabled {

color: #aaa;

cursor: not-allowed;

}

</style>

</head>

<body>

<div class="editor-container">

<textarea id="editor" placeholder="Start typing..."></textarea>

<div class="controls">

<button id="undo" disabled>Undo</button>

<button id="redo" disabled>Redo</button>

</div>

</div>

<script>

class Node {

constructor(content, cursorPos) {

this.content = content;

this.cursorPos = cursorPos;

this.prev = null;

this.next = null;

}

}

class VersionControl {

constructor() {

this.current = null;

}

addEdit(content, cursorPos) {

const newNode = new Node(content, cursorPos);

if (this.current) {

this.current.next = newNode;

newNode.prev = this.current;

}

this.current = newNode;

}

undo() {

if (this.current && this.current.prev) {

this.current = this.current.prev;

return this.current;

}

return null;

}

redo() {

if (this.current && this.current.next) {

this.current = this.current.next;

return this.current;

}

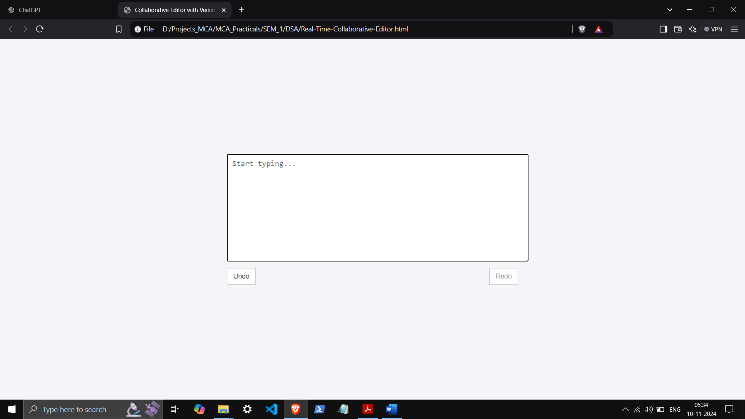
return null;

}

}

const editor = document.getElementById('editor');

const undoButton = document.getElementById('undo');

 const redoButton = document.getElementById('redo');

const versionControl = new VersionControl();

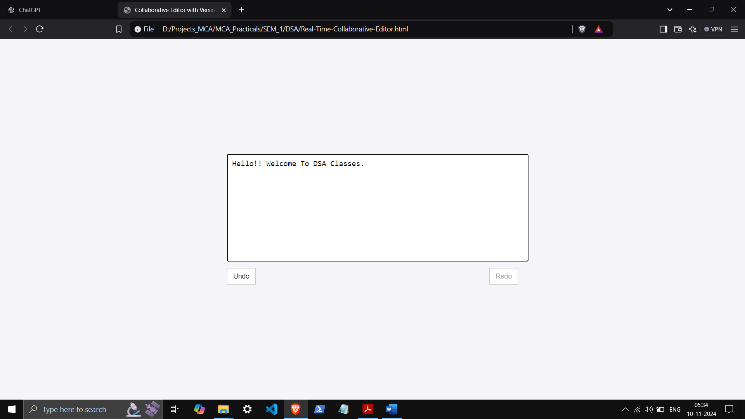
// Save initial state

versionControl.addEdit(editor.value, editor.selectionStart);

editor.addEventListener('input', () => {

versionControl.addEdit(editor.value, editor.selectionStart);

undoButton.disabled = false;

 redoButton.disabled = true;

});

undoButton.addEventListener('click', () => {

const node = versionControl.undo();

if (node) {

editor.value = node.content;

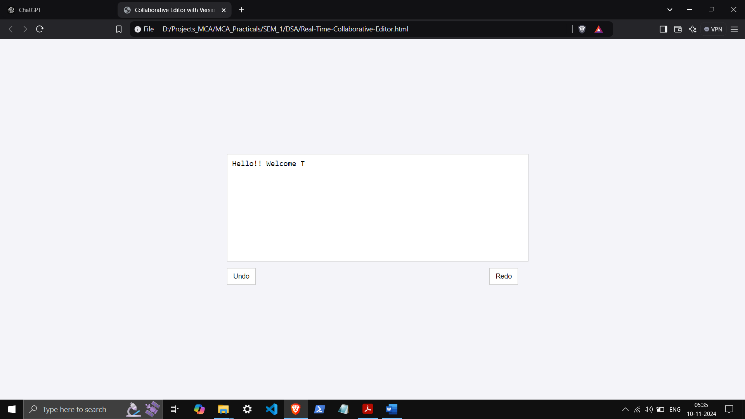
editor.selectionStart = editor.selectionEnd = node.cursorPos;

redoButton.disabled = false;

}

if (!versionControl.current.prev) {

undoButton.disabled = true;

 }

});

redoButton.addEventListener('click', () => {

const node = versionControl.redo();

if (node) {

editor.value = node.content;

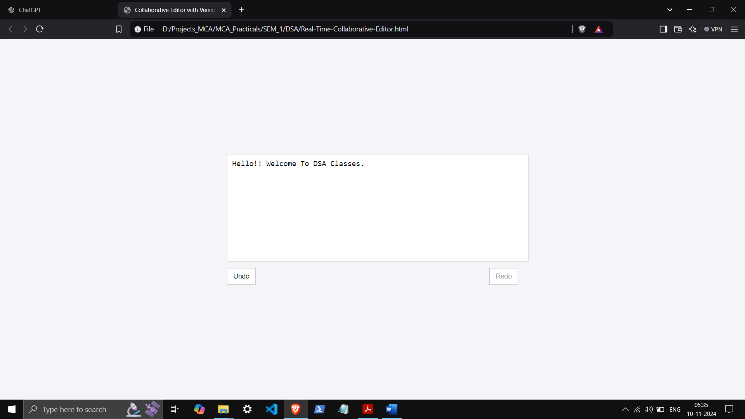
editor.selectionStart = editor.selectionEnd = node.cursorPos;

undoButton.disabled = false;

}

if (!versionControl.current.next) {

redoButton.disabled = true;

 }

});

</script>

</body>

</html>