**A picture containing text, metalware, gear

Description automatically generated**

**Project Report:**

**Multiplayer Snowball Game using Socket Programming**

**Group Members:**

**Abdul Rehman(21K-4529)**

**Bilal Ali(21K-3153)**

**Ehaab Tariq(21K-3372)**

**Instructed by:**

**Sir Kashan Hussain**

**1. Introduction:**

The Multiplayer Snowball Game is a real-time multiplayer game developed for the Computer Networks Laboratory semester project. This game enables two players to engage in a lively snowball fight on a compact map. Players control their characters using the WASD keys for movement and can hurl snowballs at each other by clicking the mouse, with the direction of the throw determined by cursor movement. The project employs Socket Programming, specifically utilizing Socket.io and Node.js technologies for real-time communication between players.

**2. Objective:**

The project aims to implement an entertaining multiplayer game utilizing Socket Programming, demonstrating seamless communication between clients and the server in real-time. The game intends to deliver an engaging and interactive experience, enabling players to compete against each other in an exciting snowball fight scenario.

**3. Technologies Used:**

* Socket.io: A JavaScript library facilitating real-time, bidirectional, and event-based communication between web clients and servers.
* Node.js: An open-source, cross-platform JavaScript runtime environment executing JavaScript code server-side.

**4. Features:**

* Real-time Multiplayer Gameplay: The game offers real-time multiplayer functionality, allowing two players to connect and play against each other simultaneously.
* WASD Control: Players can control their characters using the WASD keys for movement, providing intuitive and responsive control.
* Snowball Throwing Mechanism: Players can throw snowballs at each other by clicking the mouse. The direction and velocity of the throw are determined by cursor movement.
* Map with Environmental Elements: The game map includes various environmental elements such as rocks and trees, adding to the complexity and dynamics of the gameplay.
* Respawn Mechanism: When a player dies, they respawn at the starting location, located at coordinates (0, 0) on the map.
* Background Music: The game features background music of running on snow, enhancing the immersive experience for players.

**5. Implementation:**

* Server-Side: The server is implemented using Node.js, utilizing Socket.io for handling communication between clients. It manages player connections, game sessions, and facilitates real-time updates between players.
* Client-Side: The client-side of the game is developed using HTML, CSS, and JavaScript. Socket.io client library is used to establish a connection with the server and handle real-time events such as player movements and snowball throws.

**6. Challenges Faced:**

* Synchronization: Ensuring synchronization between client actions and server responses posed a challenge, especially in a real-time multiplayer environment.
* Latency: Managing latency and minimizing delays in communication between clients and the server was crucial to maintaining smooth gameplay.
* Collision Detection: Implementing accurate collision detection for snowballs and player characters required careful consideration and testing.

**7. Future Enhancements:**

* Enhanced Graphics and Animation: Improving the visual elements of the game, including graphics and animations, to create a more immersive experience.
* Additional Environmental Elements: Adding more environmental elements such as obstacles and interactive objects to diversify gameplay and increase strategic depth.
* Expanded Sound Design: Incorporating a wider range of sound effects and ambient audio to further enhance the immersive experience for players.

**8. Conclusion:**

The Multiplayer Snowball Game project demonstrates the successful implementation of real-time multiplayer gameplay using Socket Programming. Through effective utilization of Socket.io and Node.js, the game provides an engaging and interactive experience for players, fostering competition and collaboration in a snowball fight scenario. With further enhancements and refinements, the project has the potential to evolve into a fully-fledged multiplayer gaming platform.

**Coded Representation:**

**Server Side:**

**const express = require("express");**

**const { createServer } = require("http");**

**const { Server } = require("socket.io");**

**const app = express();**

**const httpServer = createServer(app);**

**const io = new Server(httpServer);**

**const PORT = process.env.PORT || 5000;**

**const loadMap = require("./mapLoader");**

**const SPEED = 5;**

**const TICK\_RATE = 30;**

**const SNOWBALL\_SPEED = 11;**

**const PLAYER\_SIZE = 32;**

**const TILE\_SIZE = 32;**

**let players = [];**

**let snowballs = [];**

**const inputsMap = {};**

**let ground2D, decal2D;**

**function isColliding(rect1, rect2) {**

**return (**

**rect1.x < rect2.x + rect2.w &&**

**rect1.x + rect1.w > rect2.x &&**

**rect1.y < rect2.y + rect2.h &&**

**rect1.h + rect1.y > rect2.y**

**);**

**}**

**function isCollidingWithMap(player) {**

**for (let row = 0; row < decal2D.length; row++) {**

**for (let col = 0; col < decal2D[0].length; col++) {**

**const tile = decal2D[row][col];**

**if (**

**tile &&**

**isColliding(**

**{**

**x: player.x,**

**y: player.y,**

**w: 32,**

**h: 32,**

**},**

**{**

**x: col \* TILE\_SIZE,**

**y: row \* TILE\_SIZE,**

**w: TILE\_SIZE,**

**h: TILE\_SIZE,**

**}**

**)**

**) {**

**return true;**

**}**

**}**

**}**

**return false;**

**}**

**function tick(delta) {**

**for (const player of players) {**

**const inputs = inputsMap[player.id];**

**const previousY = player.y;**

**const previousX = player.x;**

**if (inputs.up) {**

**player.y -= SPEED;**

**} else if (inputs.down) {**

**player.y += SPEED;**

**}**

**if (isCollidingWithMap(player)) {**

**player.y = previousY;**

**}**

**if (inputs.left) {**

**player.x -= SPEED;**

**} else if (inputs.right) {**

**player.x += SPEED;**

**}**

**if (isCollidingWithMap(player)) {**

**player.x = previousX;**

**}**

**}**

**for (const snowball of snowballs) {**

**snowball.x += Math.cos(snowball.angle) \* SNOWBALL\_SPEED;**

**snowball.y += Math.sin(snowball.angle) \* SNOWBALL\_SPEED;**

**snowball.timeLeft -= delta;**

**for (const player of players) {**

**if (player.id === snowball.playerId) continue;**

**const distance = Math.sqrt(**

**(player.x + PLAYER\_SIZE / 2 - snowball.x) \*\* 2 +**

**(player.y + PLAYER\_SIZE / 2 - snowball.y) \*\* 2**

**);**

**if (distance <= PLAYER\_SIZE / 2) {**

**player.x = 0;**

**player.y = 0;**

**snowball.timeLeft = -1;**

**break;**

**}**

**}**

**}**

**snowballs = snowballs.filter((snowball) => snowball.timeLeft > 0);**

**io.emit("players", players);**

**io.emit("snowballs", snowballs);**

**}**

**async function main() {**

**({ ground2D, decal2D } = await loadMap());**

**io.on("connect", (socket) => {**

**console.log("user connected", socket.id);**

**inputsMap[socket.id] = {**

**up: false,**

**down: false,**

**left: false,**

**right: false,**

**};**

**players.push({**

**id: socket.id,**

**x: 800,**

**y: 800,**

**});**

**socket.emit("map", {**

**ground: ground2D,**

**decal: decal2D,**

**});**

**socket.on("inputs", (inputs) => {**

**inputsMap[socket.id] = inputs;**

**});**

**// socket.on("mute", (isMuted) => {**

**// const player = players.find((player) => player.id === socket.id);**

**// player.isMuted = isMuted;**

**// });**

**socket.on("voiceId", (voiceId) => {**

**const player = players.find((player) => player.id === socket.id);**

**player.voiceId = voiceId;**

**});**

**socket.on("snowball", (angle) => {**

**const player = players.find((player) => player.id === socket.id);**

**snowballs.push({**

**angle,**

**x: player.x,**

**y: player.y,**

**timeLeft: 1000,**

**playerId: socket.id,**

**});**

**});**

**socket.on("disconnect", () => {**

**players = players.filter((player) => player.id !== socket.id);**

**});**

**});**

**app.use(express.static("public"));**

**httpServer.listen(PORT);**

**let lastUpdate = Date.now();**

**setInterval(() => {**

**const now = Date.now();**

**const delta = now - lastUpdate;**

**tick(delta);**

**lastUpdate = now;**

**}, 1000 / TICK\_RATE);**

**}**

**main();**

**Client Side(Front End):**

**const mapImage = new Image();**

**mapImage.src = "/snowy-sheet.png";**

**const santaImage = new Image();**

**santaImage.src = "/fastgol.png";**

**const walkSnow = new Audio("walk-snow.mp3");**

**const canvasEl = document.getElementById("canvas");**

**canvasEl.width = window.innerWidth;**

**canvasEl.height = window.innerHeight;**

**const canvas = canvasEl.getContext("2d");**

**const socket = io();**

**//const client = AgoraRTC.createClient({ mode: "rtc", codec: "vp8" });**

**const localTracks = {**

**audioTrack: null,**

**};**

**let isPlaying = true;**

**const remoteUsers = {};**

**window.remoteUsers = remoteUsers;**

**let groundMap = [[]];**

**let decalMap = [[]];**

**let players = [];**

**let snowballs = [];**

**const TILE\_SIZE = 32;**

**const SNOWBALL\_RADIUS = 5;**

**socket.on("connect", () => {**

**console.log("connected");**

**});**

**socket.on("map", (loadedMap) => {**

**groundMap = loadedMap.ground;**

**decalMap = loadedMap.decal;**

**});**

**socket.on("players", (serverPlayers) => {**

**players = serverPlayers;**

**});**

**socket.on("snowballs", (serverSnowballs) => {**

**snowballs = serverSnowballs;**

**});**

**const inputs = {**

**up: false,**

**down: false,**

**left: false,**

**right: false,**

**};**

**window.addEventListener("keydown", (e) => {**

**if (e.key === "w") {**

**inputs["up"] = true;**

**} else if (e.key === "s") {**

**inputs["down"] = true;**

**} else if (e.key === "d") {**

**inputs["right"] = true;**

**} else if (e.key === "a") {**

**inputs["left"] = true;**

**}**

**if (["a", "s", "w", "d"].includes(e.key) && walkSnow.paused) {**

**walkSnow.play();**

**}**

**socket.emit("inputs", inputs);**

**});**

**window.addEventListener("keyup", (e) => {**

**if (e.key === "w") {**

**inputs["up"] = false;**

**} else if (e.key === "s") {**

**inputs["down"] = false;**

**} else if (e.key === "d") {**

**inputs["right"] = false;**

**} else if (e.key === "a") {**

**inputs["left"] = false;**

**}**

**if (["a", "s", "w", "d"].includes(e.key)) {**

**walkSnow.pause();**

**walkSnow.currentTime = 0;**

**}**

**socket.emit("inputs", inputs);**

**});**

**window.addEventListener("click", (e) => {**

**const angle = Math.atan2(**

**e.clientY - canvasEl.height / 2,**

**e.clientX - canvasEl.width / 2**

**);**

**socket.emit("snowball", angle);**

**});**

**function loop() {**

**canvas.clearRect(0, 0, canvasEl.width, canvasEl.height);**

**const myPlayer = players.find((player) => player.id === socket.id);**

**let cameraX = 0;**

**let cameraY = 0;**

**if (myPlayer) {**

**cameraX = parseInt(myPlayer.x - canvasEl.width / 2);**

**cameraY = parseInt(myPlayer.y - canvasEl.height / 2);**

**}**

**const TILES\_IN\_ROW = 8;**

**// ground**

**for (let row = 0; row < groundMap.length; row++) {**

**for (let col = 0; col < groundMap[0].length; col++) {**

**let { id } = groundMap[row][col];**

**const imageRow = parseInt(id / TILES\_IN\_ROW);**

**const imageCol = id % TILES\_IN\_ROW;**

**canvas.drawImage(**

**mapImage,**

**imageCol \* TILE\_SIZE,**

**imageRow \* TILE\_SIZE,**

**TILE\_SIZE,**

**TILE\_SIZE,**

**col \* TILE\_SIZE - cameraX,**

**row \* TILE\_SIZE - cameraY,**

**TILE\_SIZE,**

**TILE\_SIZE**

**);**

**}**

**}**

**// decals**

**for (let row = 0; row < decalMap.length; row++) {**

**for (let col = 0; col < decalMap[0].length; col++) {**

**let { id } = decalMap[row][col] ?? { id: undefined };**

**const imageRow = parseInt(id / TILES\_IN\_ROW);**

**const imageCol = id % TILES\_IN\_ROW;**

**canvas.drawImage(**

**mapImage,**

**imageCol \* TILE\_SIZE,**

**imageRow \* TILE\_SIZE,**

**TILE\_SIZE,**

**TILE\_SIZE,**

**col \* TILE\_SIZE - cameraX,**

**row \* TILE\_SIZE - cameraY,**

**TILE\_SIZE,**

**TILE\_SIZE**

**);**

**}**

**}**

**for (const player of players) {**

**canvas.drawImage(santaImage, player.x - cameraX, player.y - cameraY);**

**if (player !== myPlayer) {**

**if (**

**remoteUsers[player.voiceId] &&**

**remoteUsers[player.voiceId].audioTrack**

**) {**

**const distance = Math.sqrt(**

**(player.x - myPlayer.x) \*\* 2 + (player.y - myPlayer.y) \*\* 2**

**);**

**const ratio = 1.0 - Math.min(distance / 700, 1);**

**remoteUsers[player.voiceId].audioTrack.setVolume(**

**Math.floor(ratio \* 100)**

**);**

**}**

**}**

**}**

**for (const snowball of snowballs) {**

**canvas.fillStyle = "#FFFFFF";**

**canvas.beginPath();**

**canvas.arc(**

**snowball.x - cameraX,**

**snowball.y - cameraY,**

**SNOWBALL\_RADIUS,**

**0,**

**2 \* Math.PI**

**);**

**canvas.fill();**

**}**

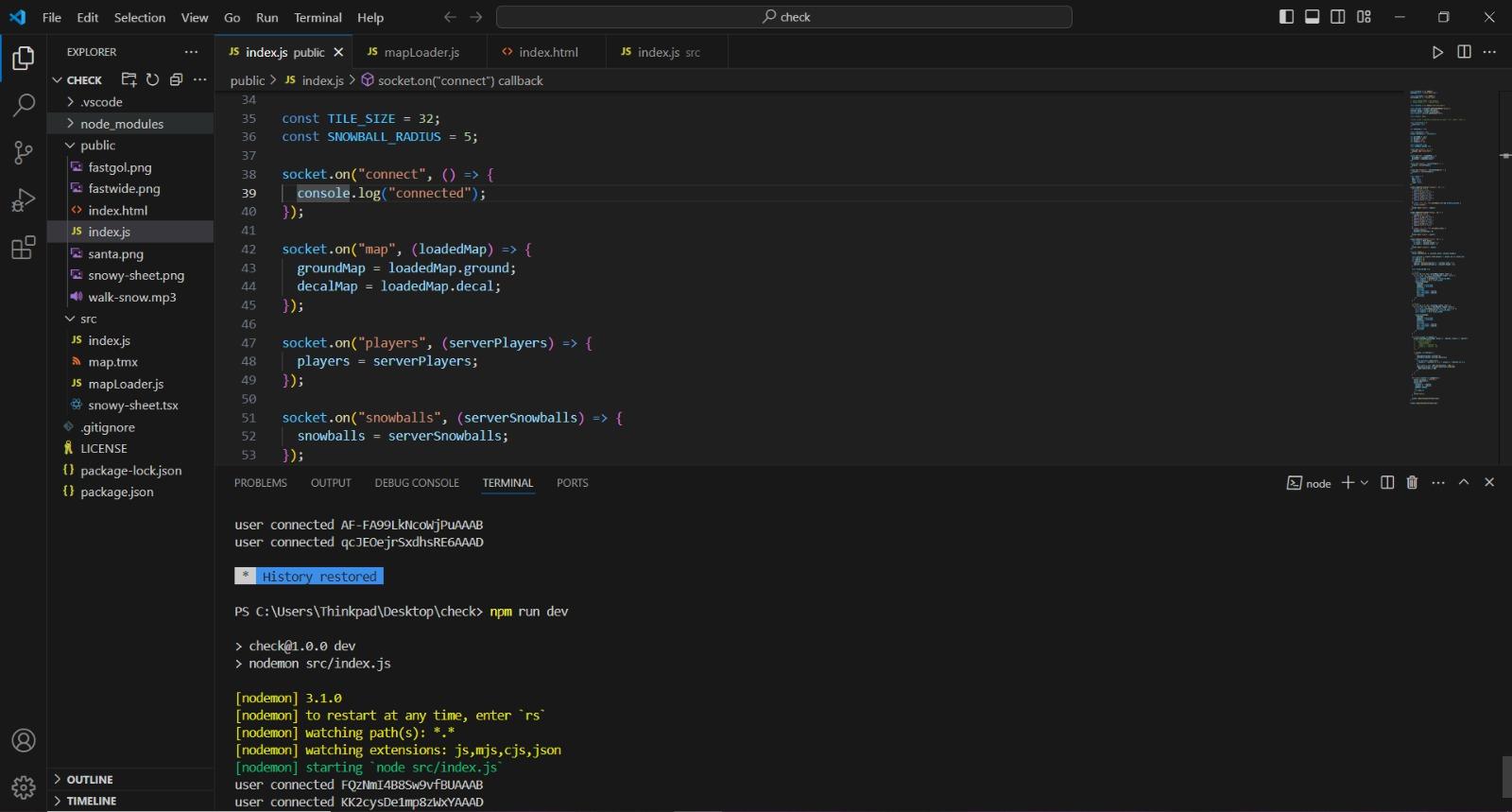
**window.requestAnimationFrame(loop);**

**}**

**window.requestAnimationFrame(loop);**

**Graphical Representation of the Project(Screenshots):**

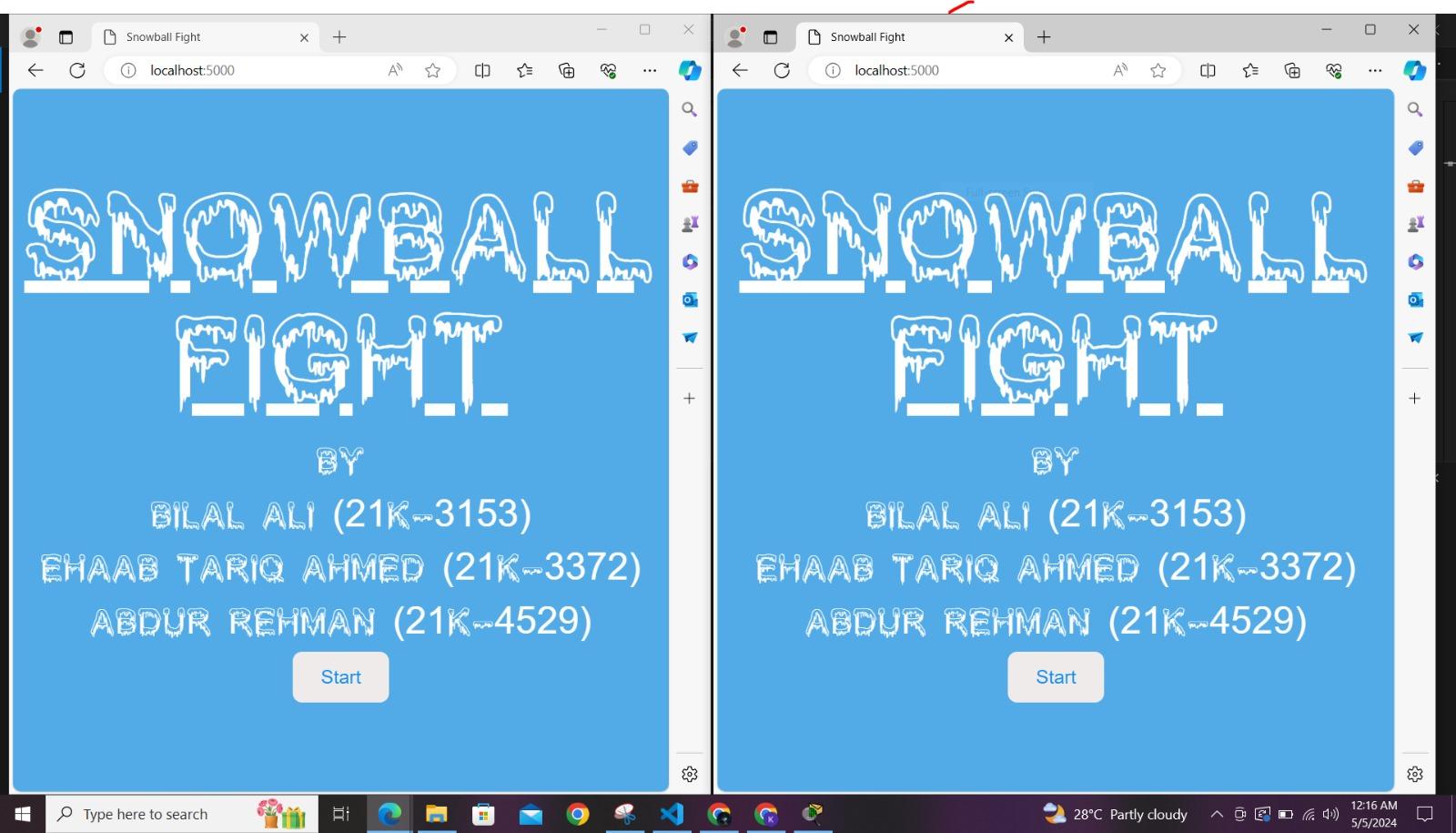
**Showing two users connecting:**

****

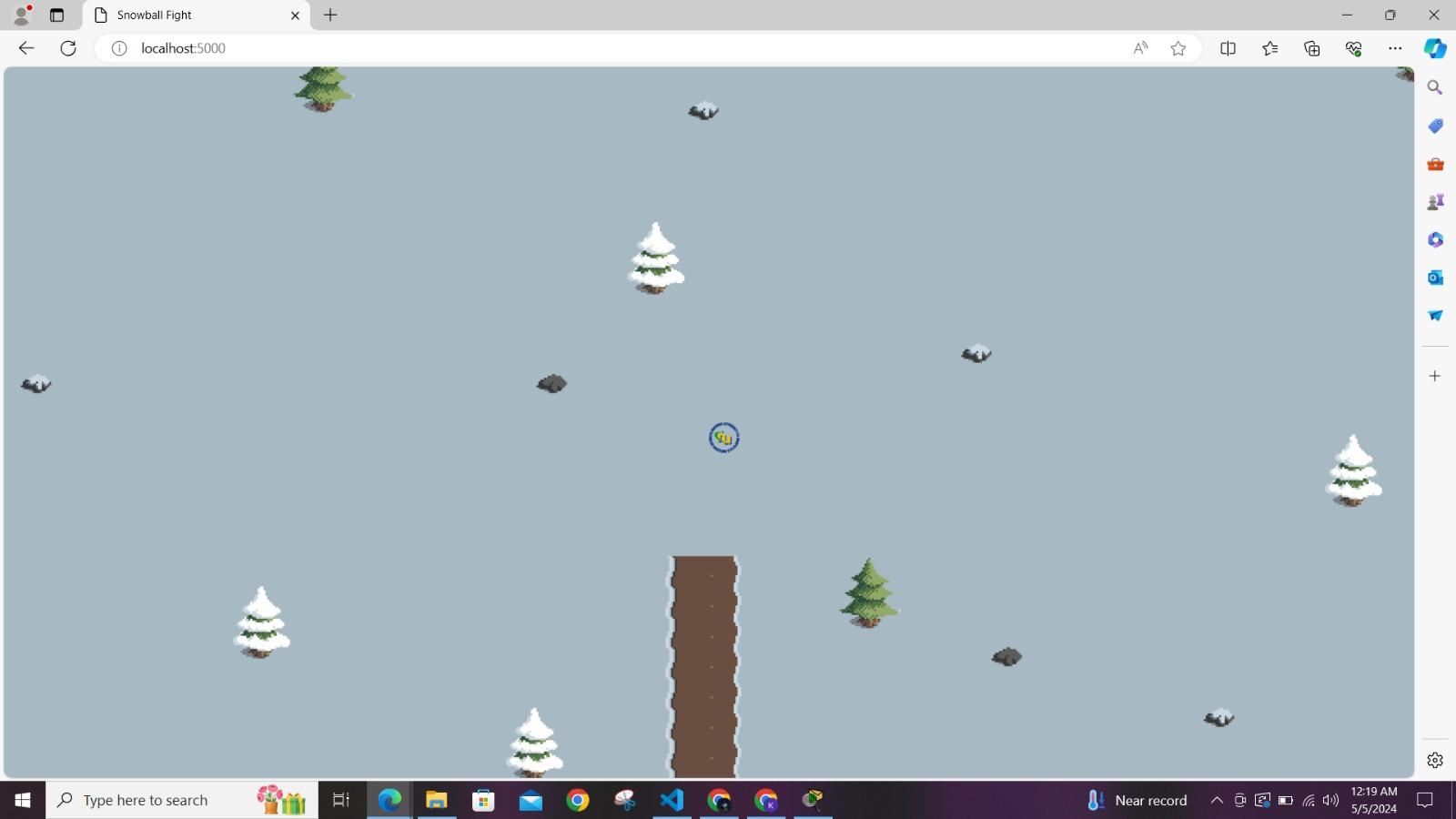
**Main Page:**

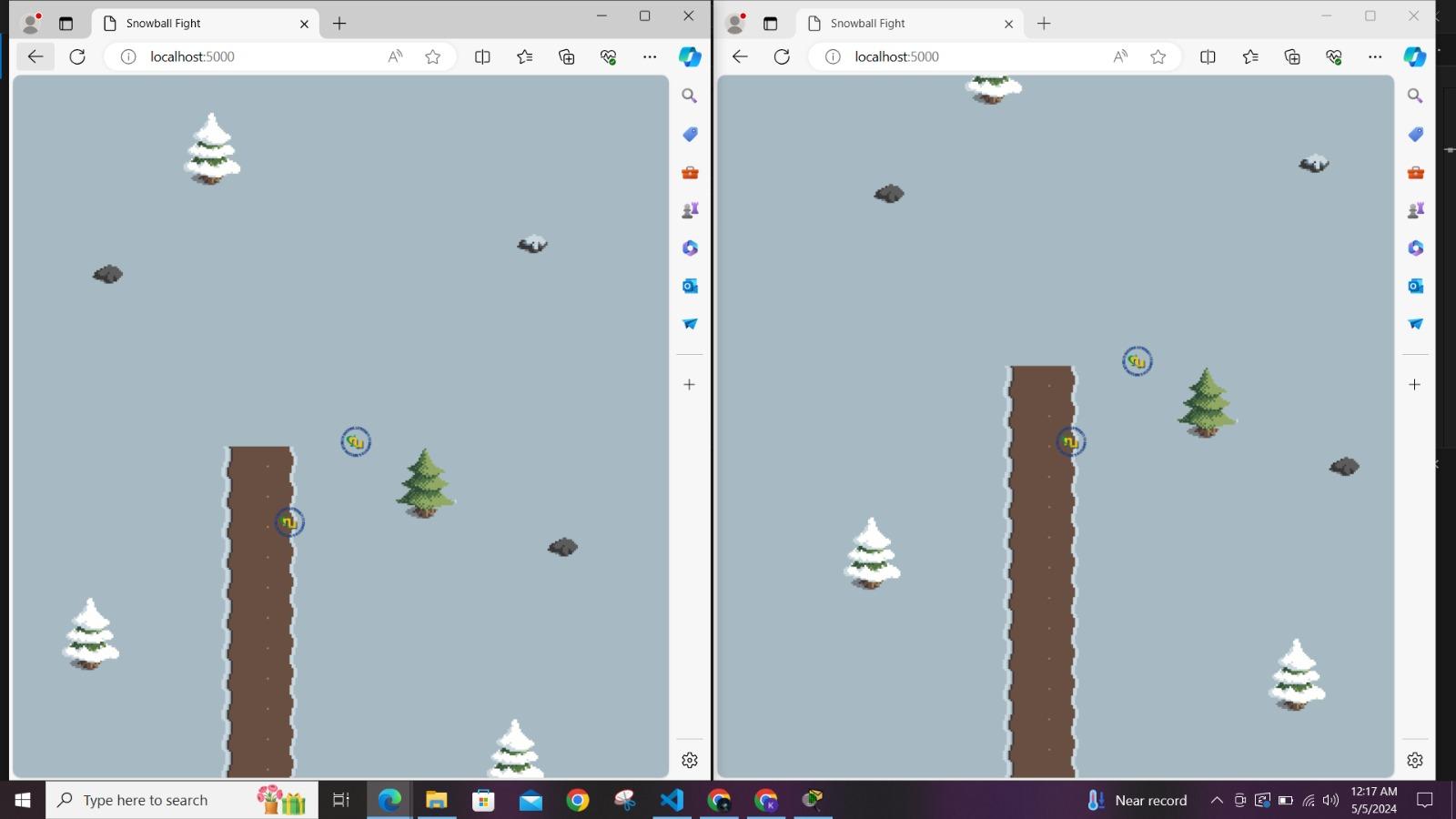
****

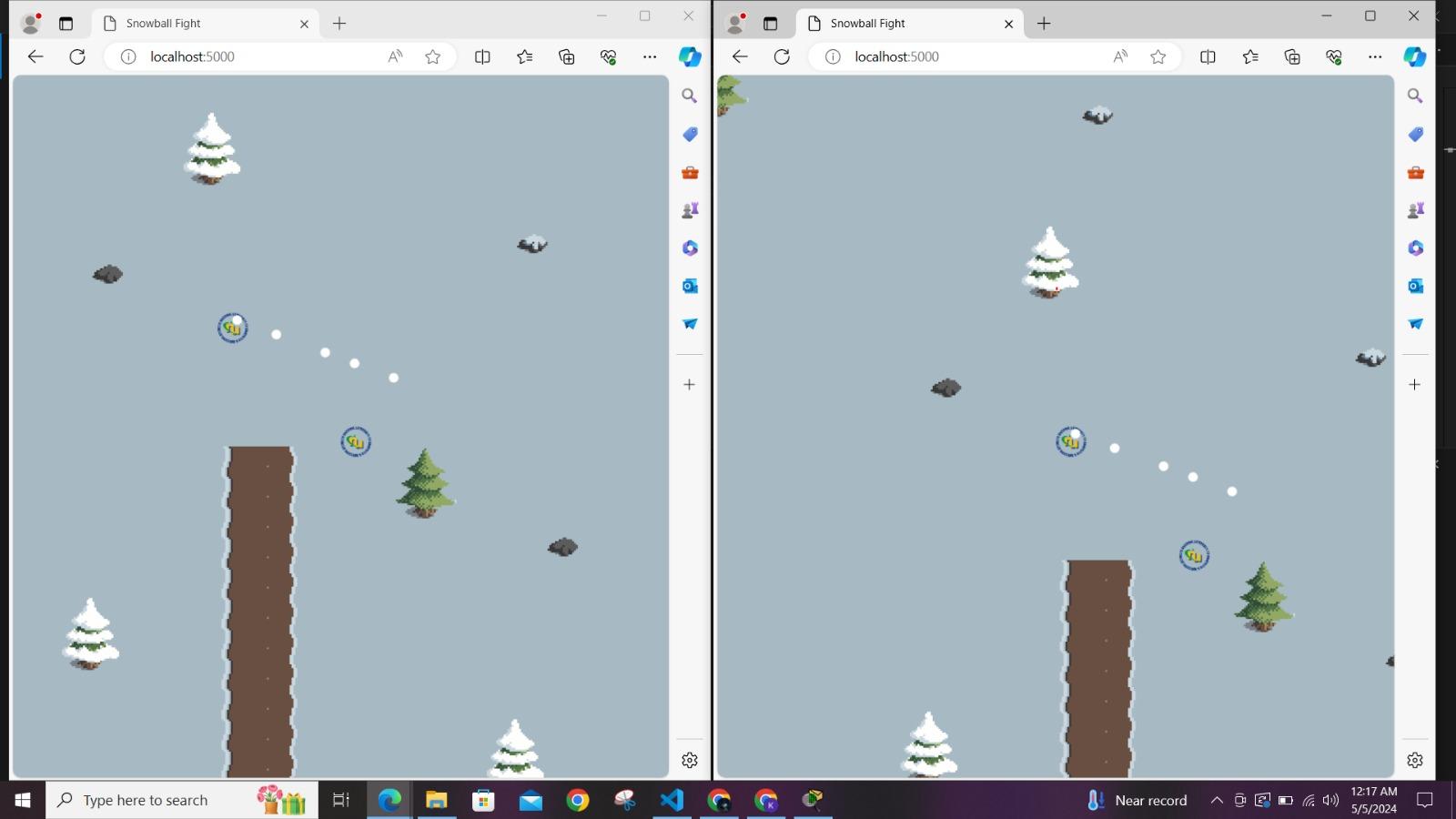
**Multiple Clients:**

****

**Game Started:**

****

****

****

**9. References:**

* Socket.io Documentation: https://socket.io/docs/
* Node.js Documentation: https://nodejs.org/en/docs/