Can ft\_transcendance Be Done Without a Backend?

Yes, ft\_transcendance can be implemented without a backend by using only frontend technologies, including JavaScript, WebSockets (or WebRTC), and browser storage. Below, I’ll explain how this is possible and why you might consider this approach.

---

1. How Can ft\_transcendance Work Without a Backend?

Without a backend, all logic, storage, and real-time communication must be handled client-side. The key areas that usually require a backend can be replaced with frontend-based solutions:

Key Technologies Used in a Backend-less Approach

WebSockets or WebRTC → For real-time multiplayer communication.

LocalStorage / IndexedDB → For storing tournament history and settings.

JavaScript Game Engine (Canvas API) → To run the Pong game fully in the browser.

---

2. Why Would You Want to Do ft\_transcendance Without a Backend?

Pros of a Backend-less Approach

1. Simplifies Development

No need to set up a Ruby/Django backend.

No database, so no need to manage storage and queries.

Fewer security concerns (e.g., SQL injections, authentication handling).

2. Easier Deployment

Can be hosted on GitHub Pages, Netlify, or Vercel without needing a server.

No need for a Dockerized backend—only frontend files are served.

3. Real-Time Performance

WebRTC allows direct peer-to-peer connections, reducing latency.

No server bottleneck, making the game potentially faster.

4. Less Resource-Intensive

No server costs (you avoid hosting databases or a backend API).

The entire game runs client-side, meaning it can be played offline (except multiplayer).

---

Cons of a Backend-less Approach

1. Limited User Authentication

Without a backend, you can’t have traditional user logins with passwords.

Solution: Use OAuth (e.g., 42 API, Google Login) or local user profiles stored in LocalStorage.

2. Difficult Multiplayer Handling

Matchmaking and keeping players connected require a workaround.

Solution: Use WebRTC for direct connections or WebSockets through a third-party relay.

3. Less Secure Data Storage

Storing tournament scores in LocalStorage or IndexedDB means users can manipulate them.

Solution: Accept this trade-off or use cloud storage (Firebase, Supabase, or Web3 options like IPFS).

---

3. How to Implement ft\_transcendance Without a Backend

A. Handling Multiplayer

Since there’s no backend to relay game data, you can:

Use WebRTC → Peer-to-peer communication between players.

Use WebSockets via a third-party service (like Pusher or Firebase).

Example: WebRTC Peer-to-Peer Connection

const peerConnection = new RTCPeerConnection();

// Signaling process (handled by a WebRTC STUN server)

peerConnection.onicecandidate = (event) => {

if (event.candidate) {

sendToOpponent(event.candidate);

}

};

peerConnection.ondatachannel = (event) => {

const channel = event.channel;

channel.onmessage = (e) => console.log("Received:", e.data);

};

const dataChannel = peerConnection.createDataChannel("game");

dataChannel.onopen = () => console.log("Connection established");

// Send game state

dataChannel.send(JSON.stringify({ ball: { x: 50, y: 50 } }));

Alternative: WebSocket with a Free Service

Instead of hosting a WebSocket server, you can use a free relay:

const socket = new WebSocket("wss://your-free-websocket-server.com");

socket.onmessage = (event) => {

const data = JSON.parse(event.data);

console.log("Received game update:", data);

};

socket.send(JSON.stringify({ player: "player1", action: "moveUp" }));

---

B. Storing User Data Without a Database

Since there’s no backend database, LocalStorage or IndexedDB can store user settings, match history, and leaderboard data.

Example: Store User Scores Locally

const scores = JSON.parse(localStorage.getItem("scores")) || [];

function saveScore(player, score) {

scores.push({ player, score, date: new Date() });

localStorage.setItem("scores", JSON.stringify(scores));

}

Alternative: Cloud Storage Without a Backend

Firebase (firebase.firestore()) → Store scores in a cloud database without a backend.

Supabase (supabase.from('scores').insert({...})) → An open-source Firebase alternative.

---

C. Running the Game in the Browser

Using JavaScript + Canvas API, you can make Pong fully functional without a server.

Example: Basic Pong Game Without Backend

const canvas = document.getElementById("pong");

const ctx = canvas.getContext("2d");

let ball = { x: 50, y: 50, dx: 2, dy: 2 };

let paddle = { x: 10, y: 40, width: 10, height: 50 };

function update() {

ball.x += ball.dx;

ball.y += ball.dy;

// Ball collision with walls

if (ball.y <= 0 || ball.y >= canvas.height) ball.dy \*= -1;

// Ball collision with paddle

if (ball.x <= paddle.x + paddle.width && ball.y > paddle.y && ball.y < paddle.y + paddle.height) {

ball.dx \*= -1;

}

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.fillRect(paddle.x, paddle.y, paddle.width, paddle.height);

ctx.fillRect(ball.x, ball.y, 10, 10);

requestAnimationFrame(update);

}

update();

---

4. Conclusion: Should You Go Backend-less?

Choose Backend-less If:

✅ You want a lighter, more deployable project that works in a browser.

✅ You prefer avoiding complex backend logic (e.g., databases, authentication).

✅ You like peer-to-peer WebRTC/WebSockets for multiplayer.

Choose a Backend If:

❌ You need user authentication beyond OAuth.

❌ You want secure data storage (e.g., protected leaderboards).

❌ You plan on scaling to many concurrent players (WebRTC has connection limits).

---

Next Steps

Would you like:

1. A step-by-step guide to implementing WebRTC for matchmaking?

2. Help on deploying the frontend to GitHub Pages/Vercel?

3. A fully frontend-based tournament system?

Let me know, and I’ll provide detailed instructions!