1.4 backend-less roadmap to check

Roadmap for ft\_transcendance Without a Backend

This roadmap logically distributes tasks while ensuring the project is functional, scalable, and meets the required modules. Since you have two main team members and a third member handling Docker/infrastructure, this plan focuses on an efficient workflow.

---

📌 Project Plan Overview

✔ Core Technologies: Vanilla JavaScript, WebRTC/WebSockets, IndexedDB, Canvas API

✔ Deployment: Static hosting (GitHub Pages, Netlify, or Vercel) + Dockerized front-end

✔ Modules Targeted: Focus on essential features with logical upgrades

✔ Goal: A fully front-end SPA with real-time multiplayer Pong

---

🛠 Project Phases & Module Choices

---

📍 Phase 1: Setup & Basic Game (Week 1)

🏗 Structure the Project

Goal: Build a Single Page Application (SPA) using only JavaScript (no backend).

File Structure:

ft\_transcendance/

├── index.html # Main page

├── styles.css # CSS

├── app.js # Core logic

├── game.js # Pong mechanics

├── matchmaking.js # WebRTC handling

├── storage.js # IndexedDB handling

├── ui.js # Menus and animations

🕹 Implement Basic Pong Game

Use Canvas API for rendering the game.

Implement ball movement, paddle control, collisions.

Add score tracking (local display).

Example (game.js)

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;

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

if (ball.x <= 0 || ball.x >= canvas.width) 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();

---

📍 Phase 2: Multiplayer with WebRTC (Week 2)

🎮 Implement Remote Players

Why WebRTC? → Peer-to-peer (no backend needed).

How to connect players?

STUN/TURN server for WebRTC connection.

Signaling server (third-party WebSocket service like Pusher or Firebase).

🛠 Steps to Implement

1. Player 1 starts a WebRTC connection.

2. Player 2 receives an invite via a signaling service.

3. Game state is synced between players.

Example WebRTC Setup (matchmaking.js)

const peerConnection = new RTCPeerConnection();

peerConnection.onicecandidate = (event) => {

if (event.candidate) {

sendToOpponent(event.candidate); // Use signaling to send ICE candidates

}

};

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");

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

---

📍 Phase 3: Tournament System (Week 3)

🏆 Manage Matchmaking & Brackets

Store player scores and tournament history using IndexedDB.

Pair players based on past wins/losses.

🛠 Steps to Implement

1. LocalStorage or IndexedDB for persistent data.

2. Matchmaking assigns players in a queue.

3. Winners move forward in brackets.

Example IndexedDB Storage (storage.js)

let db;

const request = indexedDB.open("PongDB", 1);

request.onsuccess = () => {

db = request.result;

};

function saveMatchResult(player1, player2, winner) {

const transaction = db.transaction(["matches"], "readwrite");

const store = transaction.objectStore("matches");

store.add({ player1, player2, winner, date: new Date() });

}

---

📍 Phase 4: User Profiles & Stats (Week 4)

📊 Store Local Player Profiles

Allow users to set a nickname and avatar.

Track their personal stats.

🛠 Steps to Implement

1. Use LocalStorage to save user preferences.

2. Store and retrieve player stats.

3. Display leaderboards on UI.

Example (ui.js)

const username = localStorage.getItem("username") || "Guest";

document.getElementById("player-name").innerText = username;

document.getElementById("set-username").addEventListener("click", () => {

const newName = prompt("Enter your username:");

localStorage.setItem("username", newName);

});

---

📍 Phase 5: UI Enhancements (Week 5)

🎨 Improve UX/UI

Add animations (CSS).

Implement settings and sound effects.

---

📍 Phase 6: Infrastructure & Deployment (Week 6)

🚀 Dockerize the Frontend

Goal: Serve the frontend with NGINX inside Docker.

🛠 Steps

1. Create a Dockerfile:

FROM nginx:alpine

COPY . /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

2. Create a docker-compose.yml:

version: '3'

services:

frontend:

build: .

ports:

- "8080:80"

3. Run:

docker-compose up --build

---

📌 Final Notes

This plan ensures: ✅ Minimal dependencies (no backend)

✅ Logical module distribution

✅ Fast deployment with Docker

✅ P2P Multiplayer using WebRTC

Would you like more details on:

1. WebRTC signaling via Firebase/Pusher?

2. Refining tournament logic?

3. Optimizing WebRTC for better performance?