1. Initial Situation

Problem

Currently, NO existing game engine that is fully browser-based meeting the following criteria:

1. **No Local Installation Required**: Many engines rely on installable software or require special hardware.
2. **Flexible Use of Controllers and Displays**:
    - Standard PC games require keyboard/mouse combo or gamepads
    - Consoles rely on dedicated controllers, which are included when buying a new console, but very expensive when replacing them.
    - Browser-based games rarely use smartphones as controllers or support multiple screens simultaneously.
3. **Multiplayer Optimization**:
    - Existing browser-based games are often limited to very simple concepts, mainly found in the gambling industry.
    - Latency issues prevent real-time interaction in more complex game genres.

Solution & Challenges

- Web-based game services like **Google Stadia/Xbox Cloud Gaming** do exist, but are **mainly streaming services** that require powerful servers.
- Technologies like **WebSockets and WebRTC** enable real-time communication, but no unified platform has been developed specifically for browser-based multiplayer games.
- Most established game engines like Unity, Unreal or Godot are designed for native applications/games and require additional plugins/addons for web compatibility.

---

2. Concept

> Connect is a platform that revolutionizes gaming by allowing any type of browser-capable device to be used as a display or controller.

- **Registration and Role Assignment:**
    - Display:
      Each session can be shown on one or more displays.

- Controller:
  Smartphones or tablets can serve as game controllers.
- Role Selection:
  Users choose on the landing page whether their device will act as a display or controller.

- **Session Management:**
  - Session Creation:
    A user creates a session and receives a QR code or link.
  - Session Joining:
    Others scan the QR code or use the link to connect.
  - Flexibility:
    Support for local, online, and mixed gaming modes.
- **User-Friendliness:**
  - Easy operation through QR codes and an intuitive, minimalist interface.
- **Technical Implementation:**
  - Web-App:
    Blazor/AvaloniaUI for the frontend, ASP.NET for the backend, and a light/slim database/manager.
  - Mobile-App:
    Can be implemented with the same tech stack.

---

3. Objectives/Goals

The goal of this project is to develop a fully browser-based game engine with the following features:

- **Simplicity & Accessibility**: The engine should be intuitive and easy to use for users of all ages.
- **Platform Independence**: Games should run on any modern web browser without requiring installation or specialized hardware.
- **Mobile Devices as Controllers/Game Pads**: Smartphone and tablets can be registered as input devices since they are widely available and accessible to most users.
- **Flexible Display Options**: Any internet-capable screen with a browser (e.g.smartTvs, laptops, tablets, phones) can be used as a display for the game.
- **Optimization for Multiplayer Scenarios**: The engine will support different game modes:

- **Couch Co-op**: Multiplayer, people in the same room play together.
- **Distant Multiplayer**: Players connect over the internet.
- **Multi-Screen**: Players can view different content on separate displays for innovative gameplay mechanics. e.g. active player, common area, opponent player.

- **Technical Aspects and Expandability**:
  - Initial focus on **Android + Chromium** for a quick development of a working prototype.
  - Maybe later expansion to **iOS + Safari** to reach broader user-base.
  - Use of modern web technologies to provide real-time interaction and synchronization.
- **Latency Minimization through Turn-Based Gameplay**: Since browser-based technologies inherently introduce some delays, the focus will be on turn-based games to ensure a fluid gaming experience.

---

4. General Conditions and Constraints

Technical Frameworks

- **Technology Stack**:
  > - Frontend: HTML5, CSS and JavaScript including Libraries and Frameworks. **>>TO BE REPLACED<<**
  - Real-time communication: WebSockets / WebRTC for synchronization between controllers and screens
  - Backend:

  > Node.js with Express / WebSocket server for session and turn management. **>>TO BE REPLACED<<**

  - Storage: Minimal data requirements (e.g. for high scores and game states) => Optional integration of NOSQL databases like MongoDB or Firebase.
  - Compatibility: Initial version Android & Chromium, later expansion to iOS.

Gameplay Mechanics and Use Cases

- **Game Types**: Limited to turn-based or non-time-sensitive games because of the fact of latency issues, e.g. educational games, turn-based strategy a.s.o.
- **Connection Methods**:
  - Devices identify themselves via QR codes or short links for easy user-friendly registration in the game.
  - No app required - everything runs in te browser!

Limitations and Challenges

- **Latency Issues**: Real-time games require fast reaction times. This is not our target group.

- **Platform Differences**: iOS and Safari have different restrictions for WebSockets and WebRTC!
- **Privacy & Security**:
  - Minimization of personal data collection, the least possible data overhead.
  - Secure session management
  - No third-party cloud services needed unless necessary for data protection.

---

5. Opportunities and Risks

- **5.1 Risks:**
  - Technical Challenges:
    Synchronization across different devices could lead to latency and compatibility issues, hence in the initial development phase, we focus on integrating turn-based (card) games and similar.
  - User Acceptance:
    Risk that users might not be willing to deviate from conventional systems.
  - Data Privacy:
    Danger of privacy breaches or cyber-attacks.
  - Market Competition:
    Competition from established or emerging technologies.
- **Comprehensive Market-analysis**

- **5.2 Opportunities:**
  - Market Leader:
    Potential to establish a new market leader in gaming connectivity.
  - Expandability:
    Possibility to integrate additional features like cloud gaming, more complex games, and educational games.
  - Community Building:
    Strengthening social interaction and community activities. Digitizing family game night.

Our Solution Imagines It Different:

- **Device Flexibility:**

  Our solution allows the use of any type of device as either a display or a controller, local co-op, or online, which is a true innovation. This could lower the barrier to entry for cloud gaming by not requiring specialized hardware.

- **Simplicity and Accessibility:**

  By using QR codes and a simple role assignment (Display vs. Controller), **we greatly simplify the gaming experience for end-users**. This can help even those less tech-savvy to use the system.

- **Session Flexibility:**

  The ability to **play locally, online, or in mixed modes offers versatility that many current solutions do not provide without limitations**. This could make gaming more attractive for both solo players and groups.

- **Cost and Pricing Structure:**

  By offering a flat structure or a flexible payment system, we could lower the cost barrier that exists with some current providers, thus attracting more users. **Most importantly, the need to purchase additional hardware is completely eliminated with our solution.**

- **Privacy and Security:**

  With a focus on privacy, we can build trust, which is essential in today's world. The ability to secure sessions could attract additional users who are concerned about their privacy.

- **Market Focus:**

  Our solution target niche markets like families or friend groups looking for uncomplicated, social gaming.

---

6. Planning

Timeline:

- **Phase 1**:
  Concept and Design (1 month) - Creation of the detailed concept and design of the user interface.
- **Phase 2**:
  Web App Development (3 months) - Implementation of the core functionalities of the web app.

- **Phase 3**:
  Mobile App Development (2 months) - Development of the mobile app for Android.
- **Phase 4**:
  Integration and Testing (1 month) - Ensuring compatibility and bug fixing.
- **Phase 5**:
  Deployment and Beta Phase (1 month) - Going live and collecting feedback from beta testers.

Budget:

- Developer Costs:
  Based on hourly rates and project scope.
- Server/Infrastructure:
  Costs for hosting and scalability of the platform.
- Marketing:
  Budget for the launch and advertising of the platform.

Human Resources:

- A team of developers (Frontend, Backend, Mobile, QA).
- UX/UI Designer for the interface.
- Project Manager for coordinating all phases.

> The planning should be flexible enough to respond to feedback from the beta phase, and should aim to identify and resolve bottlenecks in development early on.