Realtime Game Portal — Project Guide

1️⃣ Project Overview

Realtime Game Portal is a full-stack web application that hosts multiple games in real time. Players can:

- Sign in and play multiple games (Callbreak, Hearts, Puzzle/Memory games)
- Track their scores on live leaderboards
- Compete with other players through real-time updates using WebSockets

The portal is divided into two main parts:

1. **Backend** — Node.js + Express + Socket.io
2. **Frontend** — React + Vite + Framer Motion

2️⃣ Folder Structure

```
Realtime_Game_Portal_Advanced/
|
├── backend/
|   ├── server.js        # Express server & Socket.io setup
|   ├── package.json     # Backend dependencies
|
├── frontend/
|   ├── src/
|   |   ├── App.js       # Main React component
|   |   ├── config.js    # Backend URL config
|   |   └── pages/       # Individual game pages
|   |       ├── Scores.jsx
|   |       └── History.jsx
|   ├── index.html
|   ├── vite.config.js
|   ├── package.json
|   └── dist/            # Production build
```

```
|
├── README.md
├── Game_Portal_Guide.pdf
└── Resume_Summary.txt
```

3️⃣ Backend Details

- **server.js**:

```
const express = require("express");
const cors = require("cors");
const http = require("http");
const { Server } = require("socket.io");

const app = express();
app.use(cors());
const server = http.createServer(app);

const io = new Server(server, {
  cors: { origin: "*" }
});

// Socket.io real-time events
io.on("connection", (socket) => {
  console.log("A user connected:", socket.id);

  socket.on("sendMove", (data) => {
    io.emit("receiveMove", data); // broadcast to all
  });

  socket.on("disconnect", () => {
    console.log("User disconnected:", socket.id);
  });
```

});

server.listen(5000, () => console.log("Backend ready on port 5000"));

- **Dependencies**:
  - express
  - cors
  - socket.io
- **Purpose**:
  - Handles API requests and WebSocket communication
  - Maintains real-time game state (moves, scores)

🔽 Frontend Details

- **Tech Stack**:
  - React 18
  - Vite (fast build tool)
  - React Router (routing)
  - Framer Motion (animations)
  - Socket.io-client (real-time events)
- **Key Files**:
  - App.js → Main app routing
  - config.js → Stores backend URL
  - pages/ → Individual game pages (Scores, History, Game UI)
- **Connecting Frontend to Backend**:
  - import { io } from "socket.io-client";
  - import { BACKEND_URL } from "./config";

  - const socket = io(BACKEND_URL);

  - socket.emit("sendMove", { player: "Alice", move: "card1" });
  - socket.on("receiveMove", (data) => console.log(data));

- **Build for Deployment**:
  - npm run build # creates production-ready dist folder

---

5 Deployment Instructions

Backend (Render / Any Node.js host)

1. Push backend folder to GitHub.
2. On Render: New → Web Service → Select repo → backend folder.
3. Build command: npm install
   Start command: node server.js
4. Your backend URL will look like: https://game-portal-backend.onrender.com

Frontend (Vercel / Static Host)

1. Push frontend folder to GitHub.
2. On Vercel: New Project → Select repo → frontend folder.
3. Build command: npm run build
   Output directory: dist
4. Frontend URL example: https://game-portal-frontend.vercel.app
5. Ensure frontend/src/config.js points to backend URL.

---

6 Adding New Games

1. Create a new page under frontend/src/pages/ (e.g., TicTacToe.jsx)
2. Add routing in App.js: <Route path="/tic-tac-toe" element={<TicTacToe />} />
3. Add backend logic for moves if needed in server.js.
4. Test locally → build → deploy.

---

7 Cross-Platform Best Practices

- **.gitignore** prevents node_modules and build artifacts from being committed.
- **.gitattributes** ensures consistent LF line endings on Windows, Linux, macOS.
- Anyone cloning the repo can run using:

npm install

npm run dev

- Production build:

npm run build

🛑 Notes

- Real-time games rely on WebSockets; ensure backend is live for frontend functionality.
- You can extend the portal by adding:
    - User authentication via JWT
    - Chat functionality
    - Game statistics and history storage (optional database)
- Frontend animations improve user experience but are optional.

✅ This guide gives a **full overview of architecture, setup, deployment, and extension** for your Realtime Game Portal.