**Levelboard.jsx**

import React from "react";

import { useNavigate } from "react-router-dom";

import "./LevelBoard.scss";

function LevelBoard() {

  const navigate = useNavigate();

  // Function to handle level selection and navigate to Game

  const startGame = (chances) => {

    navigate("/game", { state: { chances } }); // Pass the chosen chances as state

  };

  return (

    <div className="level-board-wrapper">

     <div className="title2"><h2>Select your level</h2></div>

      <div className="button-container">

        <button onClick={() => startGame(5)}>Easy</button>

        <button onClick={() => startGame(3)}>Medium</button>

        <button onClick={() => startGame(1)}>Hard </button>

      </div>

    </div>

  );

}

export default LevelBoard;

**leaderboard.jsx**

// src/leaderboard.jsx

import React, { useEffect, useState } from "react";

import io from "socket.io-client";

import "./leaderboard.scss";

const SOCKET\_SERVER\_URL = "http://localhost:8800";

const Leaderboard = () => {

  const [scoreboard, setScoreboard] = useState([]);

  useEffect(() => {

    // Connect to the Socket.io server

    const socket = io(SOCKET\_SERVER\_URL);

    // Emit event to request the scoreboard data initially

    socket.emit("fetchScoreboard");

    // Listen for 'scoreboardData' event and update state when new data arrives

    socket.on("scoreboardData", (data) => {

      setScoreboard(data);

    });

    // Refresh the scoreboard every 2 seconds by emitting the fetch event again

    const intervalId = setInterval(() => {

      socket.emit("fetchScoreboard");

    }, 2000); // 2-second interval

    // Cleanup when the component unmounts

    return () => {

      socket.disconnect();

      clearInterval(intervalId); // Clear the interval to prevent memory leaks

    };

  }, []);

  // Sort data by score in descending order for display

  const sortedData = [...scoreboard].sort((a, b) => b.score - a.score);

  return (

    <div className="wrapper">

      <div className="title">LeaderBoard</div>

      <div className="leaderboard-container">

        <table className="leaderboard-table">

          <thead>

            <tr>

              <th>Rank</th>

              <th>User Name</th>

              <th>Score</th>

            </tr>

          </thead>

          <tbody>

            {sortedData.map((player, index) => (

              <tr key={player.userId || index}>

                <td>{index + 1}</td>

                <td>{player.username}</td>

                <td>{player.score}</td>

              </tr>

            ))}

          </tbody>

        </table>

      </div>

    </div>

  );

};

export default Leaderboard;

**game.jsx**

import React, { useState, useEffect } from "react";

import { useLocation, useNavigate } from "react-router-dom";

import axios from "axios";

import "./game.scss";

function Game() {

  const location = useLocation();

  const navigate = useNavigate();

  const [chances, setChances] = useState(location.state?.chances || 5);

  const [questionData, setQuestionData] = useState(null);

  const [userAnswer, setUserAnswer] = useState("");

  const [solvedCount, setSolvedCount] = useState(0);

  const [questionNumber, setQuestionNumber] = useState(1);

  const [timeLeft, setTimeLeft] = useState(60);

  const [gameOver, setGameOver] = useState(false);

  const [message, setMessage] = useState("");

  useEffect(() => {

    if (!location.state) {

      navigate("/levelboard");

    }

  }, [location.state, navigate]);

  useEffect(() => {

    fetchQuestion();

  }, []);

  const fetchQuestion = async () => {

    try {

      const response = await axios.get("https://marcconrad.com/uob/banana/api.php");

      setQuestionData(response.data);

      setUserAnswer("");

      setMessage("");

    } catch (error) {

      console.error("Error fetching question:", error);

    }

  };

  const checkAnswer = () => {

    if (parseInt(userAnswer) === questionData.solution) {

      setSolvedCount(solvedCount + 1);

      setMessage("Correct!");

      setQuestionNumber(questionNumber + 1);

      fetchQuestion();

    } else {

      setChances((prevChances) => {

        if (prevChances - 1 <= 0) {

          setGameOver(true);

          setMessage("Game Over! All chances are used.");

        } else {

          setMessage("Answer is wrong. Try again!");

        }

        return prevChances - 1;

      });

    }

  };

  useEffect(() => {

    const handleKeyPress = (e) => {

      if (e.key === "Enter" && userAnswer !== "") {

        checkAnswer();

      }

    };

    window.addEventListener("keydown", handleKeyPress);

    return () => window.removeEventListener("keydown", handleKeyPress);

  }, [userAnswer]);

  useEffect(() => {

    const timer = setInterval(() => {

      setTimeLeft((prevTime) => {

        if (prevTime === 1) {

          setGameOver(true);

          setMessage("Game Over! Time is up.");

          return 0;

        }

        return prevTime - 1;

      });

    }, 1000);

    return () => clearInterval(timer);

  }, []);

  useEffect(() => {

    if (gameOver) {

      setQuestionData(null);

      setUserAnswer("");

    }

  }, [gameOver]);

  const handleNumberClick = (number) => {

    setUserAnswer(number.toString());

  };

  return (

    <div className="game-wrapper">

      {gameOver ? (

        <div className="game-over">

          <h3>{message}</h3>

          <p>Your Score: {solvedCount}</p>

          <button onClick={() => navigate("/levelboard")}>Play Again</button>

        </div>

      ) : (

        <>

          <div className="left-box">

            <p>Time Left: {timeLeft}s</p>

            <span className="countdown-timer">🎵</span> {/\* Music Icon Placeholder \*/}

          </div>

          <div className="middle-box">

            <div className="upper-part">

              <h2>Find the Banana Value</h2>

              <p>Question: {questionNumber}</p>

            </div>

            {questionData && (

              <div>

                <img src={questionData.question} alt="Banana question" className="question-image" />

              </div>

            )}

            {/\* Feedback message \*/}

            <p className={`message ${message === "Correct!" ? "correct" : "wrong"}`}>

              {message}

            </p>

            <div className="lower-part">

              <div className="number-buttons">

                {[...Array(10).keys()].map((num) => (

                  <button

                    key={num}

                    onClick={() => handleNumberClick(num)}

                    className="number-button"

                  >

                    {num}

                  </button>

                ))}

              </div>

              <p className="solved-count">Solved Count: {solvedCount}</p>

            </div>

          </div>

          <div className="right-box">

            <p>Remaining Chances: {chances}</p>

          </div>

        </>

      )}

    </div>

  );

}

export default Game;