```
import React, { useState, useEffect, useRef
} from 'react';
import {
  doc,
  setDoc,
  onSnapshot,
  collection,
  deleteDoc,
  serverTimestamp,
  updateDoc
} from 'firebase/firestore';
import { signInAnonymously,
onAuthStateChanged,
signInWithCustomToken } from 'firebase/
auth';
import { auth, db, appId } from './config';

const COLORS = [
  '#6366f1', '#ef4444', '#10b981', '#f59e0b',
'#ffffff',
  '#e879f9', '#2dd4bf', '#a855f7', '#fb7185',
```

```
'#94a3b8'
];

import React, { useState, useEffect, useRef
} from 'react';
import { initializeApp } from 'firebase/app';
import {
  getFirestore,
  doc,
  setDoc,
  onSnapshot,
  collection,
  deleteDoc,
  serverTimestamp,
  updateDoc
} from 'firebase/firestore';
import { getAuth, signInAnonymously,
onAuthStateChanged,
signInWithCustomToken } from 'firebase/
auth';
```

```
const firebaseConfig =
JSON.parse(__firebase_config);
const app = initializeApp(firebaseConfig);
const auth = getAuth(app);
const db = getFirestore(app);
const appId = typeof __app_id !==
'undefined' ? __app_id : 'sky-dash-lobby-fix';

const COLORS = [
  '#6366f1', '#ef4444', '#10b981', '#f59e0b',
'#ffffff',
  '#e879f9', '#2dd4bf', '#a855f7', '#fb7185',
'#94a3b8'
];

const getShortId = () => {
  let id =
localStorage.getItem('skyDashShortId');
  if (!id) {
    id = Math.floor(1000 + Math.random() *
9000).toString();
```

```
    localStorage.setItem('skyDashShortId',
id);
  }
  return id;
};

export default function App() {
  const [screen, setScreen] =
useState('HOME');
  const [user, setUser] = useState(null);
  const [shortId] = useState(getShortId());
  const [shipColor, setShipColor] =
useState('#6366f1');
  const [multiplayerMode,
setMultiplayerMode] = useState(null);
  const [friendIdInput, setFriendIdInput] =
useState("");
  const [nickname, setNickname] =
useState(localStorage.getItem('skyDashNic
k') || "");
  const [gameScore, setGameScore] =
```

```
useState(0);
  const [others, setOthers] = useState({});
  const [highScore, setHighScore] =
useState(Number(localStorage.getItem('sk
yDashSpaceScore')) || 0);

  // States per Sfide e Matchmaking
  const [incomingChallenge,
setIncomingChallenge] = useState(null);
  const [challengeStatus,
setChallengeStatus] = useState(null);

  // States per Skin e PIN
  const [skinUnlocked, setSkinUnlocked] =
useState(false);
  const [showPinPad, setShowPinPad] =
useState(false);
  const [pendingColor, setPendingColor] =
useState(null);
  const [pinInput, setPinInput] =
useState("");
```

```
  const SECRET_PIN = "4012";

  const canvasRef = useRef(null);
  const engine = useRef({
    ship: { x: 100, y: 0, vy: 0 },
    pipes: [],
    stars: [],
    particles: [],
    score: 0,
    lastPipeTime: 0
  });

  // Init Auth
  useEffect(() => {
    const initAuth = async () => {
      if (typeof __initial_auth_token !==
'undefined' && __initial_auth_token) {
        await signInWithCustomToken(auth,
__initial_auth_token);
      } else {
        await signInAnonymously(auth);
```

```
      }
    };
    initAuth();
    const unsub =
onAuthStateChanged(auth, u =>
setUser(u));
    return () => unsub();
  }, []);

  // Monitoraggio Giocatori e Sfide
  useEffect(() => {
    if (!user) return;
    const pColl = collection(db, 'artifacts',
appId, 'public', 'data', 'players');
    const unsubPlayers = onSnapshot(pColl,
(snap) => {
      const pData = {};
      snap.forEach(d => { if (d.id !== user.uid)
pData[d.id] = d.data(); });
      setOthers(pData);
    });
```

```
    const cDoc = doc(db, 'artifacts', appId,
'public', 'data', 'challenges', shortId);
    const unsubChallenges =
onSnapshot(cDoc, (snap) => {
      if (snap.exists()) {
        const data = snap.data();
        if (data.fromId !== user.uid &&
data.status === 'pending')
setIncomingChallenge(data);
        else if (data.fromId === user.uid &&
data.status === 'accepted') {
          setMultiplayerMode('FRIEND');
setFriendIdInput(data.toSid); startGame();
deleteDoc(cDoc);
        } else if (data.fromId === user.uid &&
data.status === 'rejected') {
          setChallengeStatus('rejected');
setTimeout(() => setChallengeStatus(null),
2000); deleteDoc(cDoc);
        }
```

```
      }
    });
    return () => { unsubPlayers();
unsubChallenges(); };
  }, [user, shortId]);

  // Sincronizzazione Posizione
  useEffect(() => {
    if (!user) return;
    const myDoc = doc(db, 'artifacts', appId,
'public', 'data', 'players', user.uid);
    const sync = setInterval(() => {
      setDoc(myDoc, {
        id: user.uid, sid: shortId, y:
engine.current.ship.y, color: shipColor,
        nick: nickname || "Pilota", score:
engine.current.score, status: screen,
lastUpdate: serverTimestamp()
      }, { merge: true });
    }, 250);
    return () => { clearInterval(sync);
```

```
deleteDoc(myDoc).catch(() => {}); };
  }, [user, screen, shipColor, nickname,
shortId]);

  // Auto-Start dalla Lobby se arriva un
giocatore
  useEffect(() => {
    if (screen === 'LOBBY_WAIT') {
      const playersPlaying =
Object.values(others).filter(p => p.status
=== 'PLAYING');
      if (playersPlaying.length > 0) {
        setMultiplayerMode('RANDOM');
        startGame();
      }
    }
  }, [others, screen]);

  // Game Loop
  useEffect(() => {
    const canvas = canvasRef.current;
```

```
    if (!canvas) return;
    const ctx = canvas.getContext('2d', {
alpha: false });
    const cfg = { gravity: 0.25, pipeSpeed:
3.5, gap: 220, spawn: 1500 };

    if (engine.current.stars.length === 0) {
      engine.current.stars =
Array.from({length: 40}, () => ({
        x: Math.random() * canvas.width, y:
Math.random() * canvas.height, size: 1.5,
speed: Math.random() * 0.5 + 0.2
      }));
    }

    let animId;
    const update = (time) => {
      const e = engine.current;
      ctx.fillStyle = '#050505';
      ctx.fillRect(0, 0, canvas.width,
canvas.height);
```

```
      // Sfondo Stelle
      ctx.fillStyle = '#ffffff';
      e.stars.forEach(s => {
        s.x -= s.speed; if(s.x < -10) s.x =
canvas.width + 10;
        ctx.fillRect(s.x, s.y, s.size, s.size);
      });

      // Ostacoli
      if (['PLAYING',
'RESPAWN_MENU'].includes(screen)) {
        if (time - e.lastPipeTime > cfg.spawn) {
          const top = Math.random() *
(canvas.height - cfg.gap - 100) + 50;
          e.pipes.push({ x: canvas.width, top,
passed: false });
          e.lastPipeTime = time;
        }
        for (let i = e.pipes.length - 1; i >= 0; i--) {
          let p = e.pipes[i]; p.x -= cfg.pipeSpeed;
```

```
        ctx.fillStyle = '#1e1b4b';
ctx.strokeStyle = '#4f46e5'; ctx.lineWidth =
2;
        ctx.fillRect(p.x, 0, 80, p.top);
        ctx.strokeRect(p.x, 0, 80, p.top);
        ctx.fillRect(p.x, p.top + cfg.gap, 80,
canvas.height);
        ctx.strokeRect(p.x, p.top + cfg.gap,
80, canvas.height);

        if (screen === 'PLAYING') {
          if (e.ship.x + 12 > p.x && e.ship.x -
12 < p.x + 80) {
            if (e.ship.y - 8 < p.top || e.ship.y + 8
> p.top + cfg.gap) endGame();
          }
          if (!p.passed && p.x + 80 < e.ship.x)
{ p.passed = true; e.score++;
setGameScore(e.score); }
        }
        if (p.x < -100) e.pipes.splice(i, 1);
```

```
      }
    }

    // Motore e Particelle
    if (screen === 'PLAYING') {
      e.ship.vy += cfg.gravity;
      e.ship.y += e.ship.vy;

      // Fumo/Fuoco costante
      if (Math.random() > 0.4) {
        e.particles.push({
          x: e.ship.x - 14, y: e.ship.y,
          vx: -Math.random() * 2 - 1, vy:
(Math.random() - 0.5) * 1.5,
          life: 1.0, color: Math.random() > 0.6
? '#f59e0b' : '#ef4444',
          size: Math.random() * 3 + 1
        });
      }

      if (e.ship.y < 0 || e.ship.y >
```

```
canvas.height) endGame();
    } else {
      e.ship.y = canvas.height / 2 +
Math.sin(time * 0.003) * 15;
    }

    // Disegno Particelle
    for (let i = e.particles.length - 1; i >= 0;
i--) {
      let p = e.particles[i];
      p.x += p.vx; p.y += p.vy;
      p.life -= 0.035;
      if (p.life <= 0) e.particles.splice(i, 1);
      else {
        ctx.globalAlpha = p.life;
        ctx.fillStyle = p.color;
        ctx.fillRect(p.x, p.y, p.size, p.size);
      }
    }
    ctx.globalAlpha = 1.0;
```

```
      // Altri Giocatori
      Object.values(others).forEach(p => {
        if (multiplayerMode === 'FRIEND' &&
p.sid !== friendIdInput) return;
        if (multiplayerMode === 'RANDOM' &&
p.status !== 'PLAYING') return;
        ctx.globalAlpha = 0.25;
        ctx.save();
        ctx.translate(100, p.y);
        ctx.fillStyle = p.color;
        ctx.beginPath(); ctx.moveTo(18, 0);
ctx.lineTo(-12, -12); ctx.lineTo(-8, 0);
ctx.lineTo(-12, 12); ctx.fill();
        ctx.restore();
      });
      ctx.globalAlpha = 1.0;

      // Nave Giocatore
      ctx.save();
      ctx.translate(e.ship.x, e.ship.y);
      ctx.rotate(Math.min(0.4,
```

```
Math.max(-0.4, e.ship.vy * 0.08)));
      ctx.fillStyle = shipColor;
      ctx.beginPath(); ctx.moveTo(18, 0);
ctx.lineTo(-12, -12); ctx.lineTo(-8, 0);
ctx.lineTo(-12, 12); ctx.closePath(); ctx.fill();

      if (screen === 'PLAYING') {
        const grad =
ctx.createRadialGradient(-16, 0, 0, -16, 0,
18);
        grad.addColorStop(0, '#facc15');
        grad.addColorStop(1, 'transparent');
        ctx.fillStyle = grad;
        ctx.globalAlpha = 0.4;
        ctx.beginPath(); ctx.arc(-16, 0, 18, 0,
Math.PI * 2); ctx.fill();
        ctx.globalAlpha = 1.0;
      }
      ctx.restore();

      animId =
```

```
requestAnimationFrame(update);
    };

    const endGame = () => {
      if (engine.current.score > highScore) {
setHighScore(engine.current.score);
localStorage.setItem('skyDashSpaceScore',
 engine.current.score); }
      setScreen(multiplayerMode ?
'RESPAWN_MENU' : 'GAMEOVER');
    };

    animId =
requestAnimationFrame(update);
    return () =>
cancelAnimationFrame(animId);
  }, [screen, shipColor, multiplayerMode,
others, highScore, friendIdInput]);

  const handleInput = (e) => {
    if (e.target.closest('.menu-card') ||
```

```
e.target.closest('input') ||
e.target.closest('button')) return;
    if (screen === 'PLAYING') {
      engine.current.ship.vy = -5.5;
      // Scintille extra
      for (let i = 0; i < 6; i++) {
        engine.current.particles.push({
          x: engine.current.ship.x - 14, y:
engine.current.ship.y,
          vx: -Math.random() * 4 - 2, vy:
(Math.random() - 0.5) * 5,
          life: 0.8, color: '#facc15', size:
Math.random() * 4 + 2
        });
      }
    }
  };

  const startGame = () => {
    engine.current.score = 0;
engine.current.ship.y =
```

```
window.innerHeight / 2;
engine.current.ship.vy = 0;
    engine.current.pipes = [];
engine.current.lastPipeTime =
performance.now();
    setGameScore(0); setScreen('PLAYING');
  };

  const startRandomMatch = () => {
    if (nickname.length < 3) return;
    localStorage.setItem('skyDashNick',
nickname);
    // Controllo se c'è qualcuno
    const playersPlaying =
Object.values(others).filter(p => p.status
=== 'PLAYING');
    if (playersPlaying.length > 0) {
      setMultiplayerMode('RANDOM');
      startGame();
    } else {
      setScreen('LOBBY_WAIT');
```

```
    }
  };

  const sendChallenge = async () => {
    if (nickname.length < 3 ||
friendIdInput.length !== 4) return;
    localStorage.setItem('skyDashNick',
nickname);
    setChallengeStatus('waiting');
    await setDoc(doc(db, 'artifacts', appId,
'public', 'data', 'challenges', friendIdInput), {
      fromId: user.uid, fromNick: nickname,
fromSid: shortId, toSid: friendIdInput,
status: 'pending', timestamp:
serverTimestamp()
    });
  };

  const respondToChallenge = async
(accept) => {
    if (!incomingChallenge) return;
```

```
    const cRef = doc(db, 'artifacts', appId,
'public', 'data', 'challenges', shortId);
    if (accept) {
      await updateDoc(cRef, { status:
'accepted' });
      setMultiplayerMode('FRIEND');
setFriendIdInput(incomingChallenge.fromS
id); startGame();
    } else {
      await updateDoc(cRef, { status:
'rejected' });
    }
    setIncomingChallenge(null);
  };

  const handlePinPress = (num) => {
    if (pinInput.length < 4) {
      const p = pinInput + num;
setPinInput(p);
      if (p === SECRET_PIN) {
setSkinUnlocked(true);
```

```
setShowPinPad(false); if (pendingColor)
setShipColor(pendingColor);
setPinInput(""); }
      else if (p.length === 4) setTimeout(()
=> setPinInput(""), 300);
    }
  };

  return (
    <div className="fixed inset-0 bg-
[#050505] text-white font-sans overflow-
hidden select-none touch-none"
onMouseDown={handleInput}
onTouchStart={handleInput}>
      <canvas ref={canvasRef}
width={window.innerWidth}
height={window.innerHeight}
className="block w-full h-full" />

      {['PLAYING',
'RESPAWN_MENU'].includes(screen) &&
```

```
<div className="absolute top-10 w-full text-center text-6xl font-black pointer-events-none drop-shadow-lg">{gameScore}
</div>}

      <div className="absolute inset-0 flex items-center justify-center pointer-events-none p-4 text-center">

        {incomingChallenge && (
          <div className="menu-card bg-indigo-950 border-4 border-white p-8 rounded-3xl pointer-events-auto shadow-2xl animate-pulse">
            <h2 className="text-xl font-black mb-2 uppercase italic">Richiesta Sfida!</h2>
            <p className="mb-6 font-bold text-indigo-200 uppercase">{incomingChallenge.fromNick} ti sfida</p>
```

```
            <div className="flex gap-4">
              <button onClick={() =>
respondToChallenge(true)}
className="flex-1 py-4 bg-emerald-500
rounded-2xl font-black uppercase text-sm
border-b-4 border-emerald-700
active:border-0">Sì</button>
              <button onClick={() =>
respondToChallenge(false)}
className="flex-1 py-4 bg-red-500
rounded-2xl font-black uppercase text-sm
border-b-4 border-red-700
active:border-0">No</button>
            </div>
          </div>
        )}

        {screen === 'HOME' && (
          <div className="menu-card bg-
black/95 p-10 rounded-3xl border-2 border-
indigo-600 pointer-events-auto w-full max-
```

```
w-sm">
            <h1 className="text-4xl font-black
mb-1 italic uppercase tracking-tighter">Sky
Dash</h1>
            <p className="text-indigo-400
mb-8 text-[10px] uppercase font-bold
tracking-widest italic text-
center">Multiplayer Lobby Fix</p>
            <button onClick={() => {
setMultiplayerMode(null); startGame(); }}
className="w-full py-4 bg-indigo-600
rounded-2xl font-bold mb-3 active:scale-95
uppercase border-b-4 border-
indigo-800">Solo</button>
            <div className="grid grid-cols-2
gap-3">
              <button onClick={() =>
setScreen('SKIN')} className="py-3 bg-
zinc-900 border border-zinc-800
rounded-2xl font-bold text-sm
uppercase">Garage</button>
```

```
            <button onClick={() =>
setScreen('MULTIPLAYER')}
className="py-3 bg-emerald-900 border
border-emerald-500 rounded-2xl font-bold
text-sm uppercase">Online</button>
          </div>
        </div>
      )}

      {screen === 'SKIN' && (
        <div className="menu-card bg-
black/95 p-8 rounded-3xl border-2 border-
indigo-600 pointer-events-auto w-full max-
w-sm text-center">
          {!showPinPad ? (
            <div>
              <h2 className="text-xl font-bold
mb-6 uppercase italic">Verniciatura</h2>
              <div className="grid grid-cols-5
gap-3 mb-8">
                {COLORS.map(c => (
```

```
                <div key={c} onClick={() => { if
(skinUnlocked) setShipColor(c); else {
setPendingColor(c); setShowPinPad(true);
} }}
                  className={`aspect-square
rounded-xl cursor-pointer border-2
transition-all relative ${shipColor === c ?
'border-white scale-110 shadow-
[0_0_15px_rgba(99,102,241,0.5)]' : 'border-
transparent opacity-50'}`}
                  style={{ backgroundColor: c }}
                >
                  {!skinUnlocked && <div
className="absolute inset-0 bg-black/60
flex items-center justify-center text-[10px]">
🔒</div>}
                </div>
              ))}
            </div>
            <button onClick={() =>
setScreen('HOME')} className="w-full
```

```
py-3 bg-zinc-900 rounded-2xl font-bold text-xs uppercase">Chiudi</button>
            </div>
          ) : (
            <div className="flex flex-col items-center">
              <h2 className="text-lg font-bold mb-4 uppercase italic text-indigo-400">Inserire PIN</h2>
              <div className="flex gap-4 mb-8">
                {[0,1,2,3].map(i => <div key={i} className={`w-3 h-3 rounded-full border-2 border-indigo-600 ${pinInput.length > i ? 'bg-indigo-600 scale-125' : 'bg-transparent'} transition-all`}></div>)}
              </div>
              <div className="grid grid-cols-3 gap-3 w-full max-w-[220px]">

{[1,2,3,4,5,6,7,8,9,"C",0,"×"].map((btn, i) => (
```

```
                    <button key={i} onClick={() => {
if(btn === "C") setPinInput(""); else if(btn
=== "×") setShowPinPad(false); else
handlePinPress(btn); }}
className="aspect-square rounded-2xl
font-black text-xl bg-zinc-900 border
border-zinc-800 active:scale-90 transition-
all">
                      {btn}
                    </button>
                  ))}
                </div>
              </div>
            )}
          </div>
        )}

        {screen === 'MULTIPLAYER' && (
          <div className="menu-card bg-
black/95 p-8 rounded-3xl border-2 border-
indigo-600 pointer-events-auto w-full max-
```

```
w-sm flex flex-col gap-5">
            <h2 className="text-2xl font-black
italic uppercase text-center">Multiplayer</
h2>
            <input type="text" maxLength={6}
placeholder="NICKNAME..."
value={nickname}
onChange={e=>setNickname(e.target.value
.toUpperCase())}
              className="w-full bg-zinc-900
border border-zinc-800 p-4 rounded-2xl
text-sm text-center font-bold outline-none
focus:border-indigo-500" />

            <button
onClick={startRandomMatch}
className="w-full py-4 bg-indigo-600
rounded-2xl font-black active:scale-95
uppercase shadow-lg border-b-4 border-
indigo-800">Partita Rapida</button>
```

```
            <div className="bg-indigo-600/10
p-4 rounded-2xl border border-
indigo-500/30 text-center">
                <span className="text-[10px] text-
indigo-400 font-bold block mb-1
uppercase">Tuo Codice Amico</span>
                <div className="text-3xl font-
black tracking-widest text-white">{shortId}
</div>
            </div>

            <div className="flex flex-col gap-3
pt-5 border-t border-zinc-800">
                <input type="number"
placeholder="CODICE AMICO..."
value={friendIdInput}
onChange={e=>setFriendIdInput(e.target.va
lue.slice(0,4))}
                    className="w-full bg-zinc-950
border border-zinc-800 p-4 rounded-2xl
text-xl text-center font-bold outline-none" />
```

```
            <button onClick={sendChallenge}
disabled={challengeStatus === 'waiting'}
className="w-full py-4 bg-emerald-600
rounded-2xl font-black active:scale-95
uppercase shadow-lg border-b-4 border-
emerald-800">
              {challengeStatus === 'waiting' ?
'Inviando...' : 'Sfida Diretta'}
            </button>
          </div>
          <button onClick={() =>
setScreen('HOME')} className="text-
[10px] text-zinc-500 font-bold uppercase
underline mt-2 text-center">Indietro</
button>
        </div>
      )}

      {screen === 'LOBBY_WAIT' && (
        <div className="menu-card bg-
black/95 p-10 rounded-3xl border-2 border-
```

```
amber-600 pointer-events-auto w-full max-w-sm text-center">
            <div className="w-12 h-12 border-4 border-amber-600 border-t-transparent rounded-full animate-spin mx-auto mb-6"></div>
            <h2 className="text-xl font-bold mb-1 uppercase italic text-amber-500">Ricerca Piloti...</h2>
            <p className="text-zinc-600 text-[10px] mb-8 font-bold uppercase italic">Sarai lanciato appena un altro pilota entra</p>
            <button onClick={() => setScreen('MULTIPLAYER')} className="text-xs text-indigo-500 font-bold uppercase underline">Annulla</button>
        </div>
      )}
```

```
        {screen === 'GAMEOVER' && (
          <div className="menu-card bg-
black/95 p-10 rounded-3xl border-2 border-
red-600 pointer-events-auto w-full max-w-
sm text-center">
            <h2 className="text-3xl font-black
text-red-600 mb-2 italic
uppercase">Esploso!</h2>
            <p className="text-xl mb-8 font-
bold text-white uppercase">Score:
{gameScore}</p>
            <button onClick={startGame}
className="w-full py-4 bg-indigo-600
rounded-2xl font-bold mb-3 active:scale-95
uppercase border-b-4 border-
indigo-800">Riprova</button>
            <button onClick={() =>
setScreen('HOME')} className="w-full
py-2 text-zinc-600 font-bold text-xs
uppercase underline">Menu</button>
          </div>
```

```
      )}

      {screen === 'RESPAWN_MENU' && (
        <div className="menu-card bg-black/95 p-10 rounded-3xl border-2 border-amber-600 pointer-events-auto w-full max-w-sm text-center">
          <h2 className="text-3xl font-black text-amber-500 mb-2 italic uppercase">Collisione!</h2>
          <button onClick={startGame} className="w-full py-5 bg-amber-600 rounded-2xl font-black mb-4 active:scale-95 uppercase border-b-4 border-amber-800 text-lg">Rientra</button>
          <button onClick={() => { setMultiplayerMode(null); setScreen('HOME'); }} className="w-full py-2 text-zinc-600 font-bold text-xs uppercase underline">Abbandona</button>
        </div>
```

```
        )}
      </div>
    </div>
  );
}
```