```
import React, { useState, useEffect, useCallback, useMemo } from
'react';
import { initializeApp } from 'firebase/app';
import {
    getAuth,
    signInAnonymously,
    signInWithCustomToken,
    onAuthStateChanged
} from 'firebase/auth';
import {
    getFirestore,
    doc,
    onSnapshot,
    setDoc,
    collection,
    query,
    where,
    getDocs,
    updateDoc,
    getDoc
} from 'firebase/firestore';
import {
    RefreshCcw, HardHat, Shield, Target, BookOpen, GitBranch,
    Zap, Layers, Cpu, Menu, X, Rocket, AlertTriangle, CheckCircle,
    Mountain, Factory, MapPin
} from 'lucide-react';

// --- 1. 核心數據結構定義 ---
const initialMValue = { value: 0.50, base: 0.50, modifier: 0.00 };

const initialCoreState = {
    M1_Efficiency: { ...initialMValue, value: 0.75 }, // 開發效率
    M2_Autonomy: initialMValue, // 生態自愈力
    M3_Precision: { ...initialMValue, value: 0.45 },
    M4_Resilience: initialMValue, // V4.0 核心：系統韌性
    M5_Knowledge: initialMValue, // 知識核心
    M6_Security: { ...initialMValue, value: 0.88 },
    LastUpdated: Date.now(),
};

const initialResourceState = {
    // 物質元素 (M)
    M_Si: { amount: 1000, net: 10, label: '矽酸鹽' },
    M_Cm: { amount: 500, net: 5, label: '複合材料' },
    M_Ex: { amount: 100, net: 0, label: '異構資源' }, // M2 修復成本 & M4
升級成本
    // 靈元素 (L)
    L_A: { amount: 100, net: -2, label: '活化靈' },
```

```
};

// --- 2. Firebase/Firestore 初始化與狀態管理 ---

// 獲取全局變量
const appId = typeof __app_id !== 'undefined' ? __app_id :
'default-app-id';
const firebaseConfig = typeof __firebase_config !== 'undefined' ?
JSON.parse(__firebase_config) : {};
const initialAuthToken = typeof __initial_auth_token !== 'undefined' ?
__initial_auth_token : undefined;

let db, auth;

// M-值儀表板組件
const MValueIcons = {
    M1_Efficiency: { icon: RefreshCcw, color: 'text-green-400', label:
'開發效率 (M1)' },
    M2_Autonomy: { icon: GitBranch, color: 'text-indigo-400', label: '
生態自愈力 (M2)' },
    M3_Precision: { icon: Target, color: 'text-yellow-400', label: '結
構精確度 (M3)' },
    M4_Resilience: { icon: HardHat, color: 'text-sky-400', label: '系統
韌性 (M4)' }, // V4.0
    M5_Knowledge: { icon: BookOpen, color: 'text-purple-400', label: '
知識核心 (M5)' },
    M6_Security: { icon: Shield, color: 'text-red-400', label: '治安與穩
定度 (M6)' },
};

// --- M-值儀表板組件 ---
const MValueDisplay = ({ code, data }) => {
    const { icon: Icon, color, label } = MValueIcons[code];
    const value = data.value;

    // 狀態顏色和閃爍邏輯 (根據規格書 I.B)
    let statusColor = 'bg-gray-700';
    let statusTailwind = 'text-gray-300';
    let isFlashing = false;

    if (code === 'M1_Efficiency') {
        if (value >= 0.6) statusColor = 'bg-green-600';
        else if (value >= 0.3) statusColor = 'bg-yellow-600';
        else statusColor = 'bg-red-600';
    } else if (code === 'M3_Precision') {
        if (value >= 0.8) statusColor = 'bg-cyan-600';
        else if (value < 0.35) { // 稍微高於 M2 臨界點
            statusColor = 'bg-red-700';
```

```
            isFlashing = true;
        } else if (value < 0.5) {
            statusColor = 'bg-yellow-600';
        }
    } else if (code === 'M6_Security') {
        if (value >= 0.65) statusColor = 'bg-blue-600';
        else {
            statusColor = 'bg-red-700';
            isFlashing = true;
        }
    } else if (code === 'M2_Autonomy') {
        if (value > 0.5) statusColor = 'bg-indigo-600';
        else statusColor = 'bg-gray-600';
    } else if (code === 'M5_Knowledge') {
        if (value >= 0.6) statusColor = 'bg-purple-600';
        else statusColor = 'bg-gray-600';
    } else if (code === 'M4_Resilience') { // V4.0 M4 狀態顯示
        if (value >= 0.6) statusColor = 'bg-sky-600';
        else statusColor = 'bg-gray-600';
    }

    statusTailwind = statusColor.replace('bg-', 'text-');

    return (
        <div className="flex items-center justify-between p-2
bg-gray-800/70 border border-gray-700 rounded-md shadow-lg my-1
backdrop-blur-sm transition-all duration-300 hover:bg-gray-700/80">
            <div className="flex items-center space-x-2">
                {/* 狀態指示燈 */}
                <div
                    className={`w-2 h-2 rounded-full ${statusColor}
${isFlashing ? 'animate-pulse' : ''}`}
                    title={label + "狀態"}
                ></div>
                <Icon className={`w-4 h-4 ${statusTailwind} transform
rotate-[-5deg]`} />
                <span className="text-xs font-semibold
text-gray-200">{label}</span>
            </div>

            <div className="text-right">
                <span className={`text-lg font-mono font-extrabold
${statusTailwind} drop-shadow-lg`}>{value.toFixed(2)}</span>
            </div>
        </div>
    );
};
```

```
// --- M5 升級面板 (V3.0) ---
const M5UpgradePanel = ({ coreState, resourceState, handleM5Upgrade,
userId }) => {

    const CM_COST = 200;
    const M5_GAIN = 0.05;
    const M3_BONUS = 0.01;
    const currentMCm = resourceState.M_Cm.amount;
    const canUpgrade = currentMCm >= CM_COST && userId && userId !==
'MOCK_USER';

    return (
        <div className="p-4 rounded-xl border border-gray-700
shadow-xl bg-gray-800/70 h-full">
            <h3 className="text-lg font-bold text-purple-400 mb-4 flex
items-center border-b border-gray-600 pb-2">
                <BookOpen className="w-5 h-5 mr-2" />
                M5 知識核心升級 (Acquisition)
            </h3>

            <div className="space-y-3 mb-6">
                <div className="p-3 bg-gray-900 rounded-lg">
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-purple-400 font-mono">當
前 M5 知識值:</span>
                        <span className="font-extrabold text-lg
text-purple-300">{coreState.M5_Knowledge.value.toFixed(2)}</span>
                    </p>
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-cyan-400 font-mono">當前
M3 精確度:</span>
                        <span className="font-extrabold text-lg
text-cyan-300">{coreState.M3_Precision.value.toFixed(2)}</span>
                    </p>
                </div>

                <div className="p-3 bg-gray-700/50 rounded-lg text-xs
text-gray-300">
                    <p className="mb-1 text-white font-bold">
                        **知識獲取成本:** {CM_COST} 複合材料 (M_Cm)
                    </p>
                    <p className="mb-1 text-purple-300">
                        **M5 永久提升:** +{M5_GAIN.toFixed(2)} 知識值
                    </p>
                    <p className="text-cyan-300">
```

```
                        **M3 結構紅利:** +{M3_BONUS.toFixed(2)} 精確度
(永久)
                    </p>
                </div>
            </div>

            <button
                onClick={handleM5Upgrade}
                className="w-full p-3 bg-purple-700
hover:bg-purple-600 text-white font-bold rounded-xl shadow-lg
transition duration-200 active:scale-95 border border-purple-500/50
flex items-center justify-center disabled:opacity-50"
                disabled={!canUpgrade}
            >
                <Layers className="w-5 h-5 mr-2" />
                啟動 M5 知識獲取序列
            </button>

            <p className="text-center mt-3 text-xs text-gray-500">
                {!canUpgrade ? '複合材料 (M_Cm) 不足或等待驗證' : '知識累積
將永久提升核心的結構穩定性 (M3)'}
            </p>
        </div>
    );
};

// --- M4 升級面板 (V4.0 新增) ---
const M4UpgradePanel = ({ coreState, resourceState, handleM4Upgrade,
userId }) => {

    const MEX_COST = 100;
    const M4_GAIN = 0.05;
    const currentMEx = resourceState.M_Ex.amount;
    const canUpgrade = currentMEx >= MEX_COST && userId && userId !==
'MOCK_USER';

    return (
        <div className="p-4 rounded-xl border border-gray-700
shadow-xl bg-gray-800/70 h-full">
            <h3 className="text-lg font-bold text-sky-400 mb-4 flex
items-center border-b border-gray-600 pb-2">
                <HardHat className="w-5 h-5 mr-2" />
                M4 系統韌性升級 (Resilience)
            </h3>

            <div className="space-y-3 mb-6">
                <div className="p-3 bg-gray-900 rounded-lg">
                    <p className="text-sm text-gray-400 flex
```

```
justify-between">
                        <span className="text-sky-400 font-mono">當前
M4 韌性值:</span>
                        <span className="font-extrabold text-lg
text-sky-300">{coreState.M4_Resilience.value.toFixed(2)}</span>
                    </p>
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-red-400 font-mono">M6 治
理值:</span>
                        <span className="font-extrabold text-lg
text-red-300">{coreState.M6_Security.value.toFixed(2)}</span>
                    </p>
                </div>

                <div className="p-3 bg-gray-700/50 rounded-lg text-xs
text-gray-300">
                    <p className="mb-1 text-white font-bold">
                        **韌性升級成本:** {MEX_COST} 異構資源 (M_Ex)
                    </p>
                    <p className="mb-1 text-sky-300">
                        **M4 永久提升:** +{M4_GAIN.toFixed(2)} 韌性值
                    </p>
                    <p className="text-red-300">
                        **M6 失敗緩衝:** 降低 M3 轉化失敗時 M6 遭受的懲罰衝
擊。
                    </p>
                </div>
            </div>

            <button
                onClick={handleM4Upgrade}
                className="w-full p-3 bg-sky-700 hover:bg-sky-600
text-white font-bold rounded-xl shadow-lg transition duration-200
active:scale-95 border border-sky-500/50 flex items-center
justify-center disabled:opacity-50"
                disabled={!canUpgrade}
            >
                <HardHat className="w-5 h-5 mr-2" />
                啟動 M4 結構韌性強化
            </button>

            <p className="text-center mt-3 text-xs text-gray-500">
                {!canUpgrade ? '異構資源 (M_Ex) 不足或等待驗證' : 'M4 是系
統的脊髓反射，用於緩衝突發衝擊。'}
            </p>
        </div>
    );
```

```
};


// --- 3. 靜態地圖數據定義 (與 V2.0 相同) ---
const mapData = [
    [{type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '原生矽酸鹽礦藏'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Refinery', label: 'M3 提煉廠 (中央)'}, {type: 'Resource', label: '異構資源富集點'}],
    [{type: 'Resource', label: '複合材料沉積層'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '活化靈源點'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '複合材料沉積層'}],
    [{type: 'Empty', label: '荒蕪之地'}, {type: 'Refinery', label: 'M3 提煉廠 (東)'}, {type: 'ASI_Core', label: '星神核心中樞'}, {type: 'Refinery', label: 'M3 提煉廠 (西)'}, {type: 'Empty', label: '荒蕪之地'}],
    [{type: 'Resource', label: '原生矽酸鹽礦藏'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '活化靈源點'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '原生矽酸鹽礦藏'}],
    [{type: 'Empty', label: '荒蕪之地'}, {type: 'Resource', label: '異構資源富集點'}, {type: 'Empty', label: '荒蕪之地'}, {type: 'Refinery', label: 'M3 提煉廠 (南)'}, {type: 'Empty', label: '荒蕪之地'}],
];


// --- M3 轉化操作面板 (V4.0 更新：納入 M4 緩衝描述) ---
const M3ConversionPanel = ({ coreState, resourceState, handleConversion, userId }) => {
    // 獲取 M3/M6/M4 數值
    const M3_P = coreState.M3_Precision.value;
    const M6_S = coreState.M6_Security.value;
    const M4_R = coreState.M4_Resilience.value; // V4.0

    // M4 緩衝效果計算
    const M6_PENALTY_BASE = 0.05; // 基礎懲罰值
    const M6_LOSS_MITIGATION = M6_PENALTY_BASE * (1 - M4_R); // M4 緩衝後的懲罰值 (M4越高，這個值越小)
    const M6_Loss_Display = M6_LOSS_MITIGATION.toFixed(3);


    return (
        <div className="p-4 rounded-xl border border-gray-700 shadow-xl bg-gray-800/70 h-full">
            <h3 className="text-lg font-bold text-yellow-400 mb-4 flex items-center border-b border-gray-600 pb-2">
                <Factory className="w-5 h-5 mr-2" />
                M3 資源提煉與轉化 (Refinery)
            </h3>
```

```
            <div className="space-y-3 mb-6">
                <div className="p-3 bg-gray-900 rounded-lg">
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-cyan-400 font-mono">確定
性 (M3):</span>
                        <span className="font-extrabold text-lg
text-cyan-300">{M3_P.toFixed(2)}</span>
                    </p>
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-red-400 font-mono">風險治
理 (M6):</span>
                        <span className="font-extrabold text-lg
text-red-300">{M6_S.toFixed(2)}</span>
                    </p>
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-sky-400 font-mono">系統韌
性 (M4):</span>
                        <span className="font-extrabold text-lg
text-sky-300">{M4_R.toFixed(2)}</span>
                    </p>
                </div>

                <div className="p-3 bg-gray-700/50 rounded-lg text-xs
text-gray-300">
                    <p className="mb-1">
                        **輸入:** 100 矽酸鹽 (M_Si)
                    </p>
                    <p className="text-green-400">
                        **預期輸出:** 150+ 複合材料 (M_Cm) <span
className="text-white font-bold">(成功率由 M3 決定)</span>
                    </p>
                    <p className="mb-1">
                        **失敗懲罰 (M6 損失):** 基礎懲罰 -0.05
                    </p>
                    <p className="text-sky-300">
                        **M4 緩衝後損失:** 約 -{M6_Loss_Display} M6 值
                    </p>
                </div>
            </div>

            <button
                onClick={handleConversion}
                className="w-full p-3 bg-green-700 hover:bg-green-600
text-white font-bold rounded-xl shadow-lg transition duration-200
active:scale-95 border border-green-500/50 flex items-center
```

```
justify-center disabled:opacity-50"
                disabled={!userId || userId === 'MOCK_USER'}
            >
                <Zap className="w-5 h-5 mr-2" />
                啟動 M3 轉化序列
            </button>

            <p className="text-center mt-3 text-xs text-gray-500">
                {(!userId || userId === 'MOCK_USER') ? '請等待驗證完成以
進行互動' : 'M4 在 M3 確定性不足時，會平滑 M6 的衝擊。'}
            </p>
        </div>
    );
}

// --- 資源採集操作面板 (與 V3.0 相同) ---
const ResourceAcquisitionPanel = ({ hex, userId, coreState,
resourceState, handleAcquisition }) => {

    // 獲取 M1 效率值
    const M1_E = coreState.M1_Efficiency.value;

    // 確定採集目標資源和基礎產量
    let targetResourceKey = 'M_Si'; // 默認矽酸鹽
    let targetResourceLabel = resourceState.M_Si.label;
    let baseYield = 100; // 默認基礎產量
    const LA_COST = 25; // 每次採集活化靈消耗

    if (hex.label.includes('複合材料')) {
        targetResourceKey = 'M_Cm';
        targetResourceLabel = resourceState.M_Cm.label;
        baseYield = 60;
    } else if (hex.label.includes('異構資源')) {
        targetResourceKey = 'M_Ex';
        targetResourceLabel = resourceState.M_Ex.label;
        baseYield = 30;
    } else if (hex.label.includes('活化靈')) {
        targetResourceKey = 'L_A';
        targetResourceLabel = resourceState.L_A.label;
        baseYield = 10; // 活化靈的產量較低
    }

    // M1 效率加成：M1 值 (0 to 1) 乘以基礎產量作為額外加成
    const M1_Bonus = baseYield * M1_E;
    const finalYield = baseYield + M1_Bonus;

    // 檢查資源是否充足 (L_A 活化靈)
    const canAcquire = resourceState.L_A.amount >= LA_COST && userId
```

```
&& userId !== 'MOCK_USER';

    // 傳遞所有必要的參數給父級函數
    const acquisitionPayload = {
        resourceKey: targetResourceKey,
        baseYield: baseYield,
        M1_Bonus: M1_Bonus,
        LA_Cost: LA_COST,
        label: hex.label
    };

    return (
        <div className="p-4 rounded-xl border border-gray-700
shadow-xl bg-gray-800/70 h-full">
            <h3 className="text-lg font-bold text-cyan-400 mb-4 flex
items-center border-b border-gray-600 pb-2">
                <Mountain className="w-5 h-5 mr-2" />
                資源採集行動 ({hex.label})
            </h3>

            <div className="space-y-3 mb-6">
                <div className="p-3 bg-gray-900 rounded-lg">
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-green-400 font-mono">當前
M1 效率:</span>
                        <span className="font-extrabold text-lg
text-green-300">{M1_E.toFixed(2)}</span>
                    </p>
                    <p className="text-sm text-gray-400 flex
justify-between">
                        <span className="text-yellow-400 font-mono">基
礎 {targetResourceLabel} 產量:</span>
                        <span className="font-extrabold text-lg
text-yellow-300">{baseYield}</span>
                    </p>
                </div>

                <div className="p-3 bg-gray-700/50 rounded-lg text-xs
text-gray-300">
                    <p className="mb-1">
                        **M1 加成產出:** {M1_Bonus.toFixed(1)}
{targetResourceKey}
                    </p>
                    <p className="mb-1 font-bold text-white">
                        **預計總產出:** {finalYield.toFixed(1)}
{targetResourceKey}
                    </p>
```

```
                    <p>
                        **能量消耗:** {LA_COST} 活化靈 (L_A)
                    </p>
                </div>
            </div>

            <button
                onClick={() => handleAcquisition(acquisitionPayload)}
                className="w-full p-3 bg-indigo-700
hover:bg-indigo-600 text-white font-bold rounded-xl shadow-lg
transition duration-200 active:scale-95 border border-indigo-500/50
flex items-center justify-center disabled:opacity-50"
                disabled={!canAcquire}
            >
                <Layers className="w-5 h-5 mr-2" />
                啟動 M1 資源採集行動
            </button>

            <p className="text-center mt-3 text-xs text-gray-500">
                {!canAcquire && !userId ? '請等待驗證完成以進行互動' :
!canAcquire ? '活化靈 (L_A) 不足！' : 'M1 效率直接決定產量加成'}
            </p>
        </div>
    );
}

// --- 主要應用程序組件 ---
export default function App() {
    // UI/Firebase 狀態
    const [isAuthReady, setIsAuthReady] = useState(false);
    const [userId, setUserId] = useState(null);
    const [isLoading, setIsLoading] = useState(true);
    const [error, setError] = useState('');

    // 側邊欄預設為關閉 (false)
    const [isSidebarOpen, setIsSidebarOpen] = useState(false);
    // UI 互動訊息
    const [message, setMessage] = useState(null); // { type: 'success'
| 'error' | 'info', text: '...' }

    // 遊戲狀態 (從 Firestore 同步)
    const [coreState, setCoreState] = useState(initialCoreState);
    const [resourceState, setResourceState] =
useState(initialResourceState);

    // V1.8 新增：追蹤當前點選的六角形區塊
    const [selectedHex, setSelectedHex] = useState(null);
```

```
    // V3.0 新增：追蹤左側欄點擊的特殊面板
    const [activePanel, setActivePanel] = useState('M_CORE'); //
'M_CORE', 'M5_UPGRADE', 'M4_UPGRADE'


    // Firebase 初始化與驗證 (與 V1.0 相同，確保數據持久化)
    useEffect(() => {
        try {

            if (!firebaseConfig || Object.keys(firebaseConfig).length
=== 0) {
                console.warn("Firebase config is missing. Using mock
data.");
                setIsAuthReady(true);
                setUserId('MOCK_USER');
                setIsLoading(false);
                return;
            }

            const firebaseApp = initializeApp(firebaseConfig);
            db = getFirestore(firebaseApp);
            auth = getAuth(firebaseApp);

            const authenticate = async () => {
                try {
                    if (initialAuthToken) {
                        await signInWithCustomToken(auth,
initialAuthToken);
                    } else {
                        await signInAnonymously(auth);
                    }
                    console.log("Firebase Auth successful.");
                } catch (e) {
                    console.error("Firebase Auth failed:", e);
                    setError("驗證失敗：請檢查配置。");
                }
            };

            const unsubscribe = onAuthStateChanged(auth, (user) => {
                if (user) {
                    setUserId(user.uid);
                } else {
                    setUserId(null);
                }
                setIsAuthReady(true);
                setIsLoading(false);
            });
```

```
            authenticate();

            return () => unsubscribe();
        } catch (e) {
            console.error("Firebase Initialization Error:", e);
            setError("Firebase 初始化失敗。");
            setIsLoading(false);
        }
    }, []);

    // 實時資料同步 (與 V1.0 相同，確保數據持久化)
    useEffect(() => {
        if (!isAuthReady || !userId || !db || userId === 'MOCK_USER')
{
            if (userId === 'MOCK_USER') console.log("Using static mock
data for UI.");
            return;
        }

        const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
        const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');

        const initializeData = async (ref, initialData) => {
            try {
                const docSnap = await getDoc(ref);
                if (!docSnap.exists()) {
                    await setDoc(ref, initialData);
                    console.log(`Initialized ${ref.id} data.`);
                }
            } catch (e) {
                console.error(`Error initializing ${ref.id}:`, e);
            }
        };

        // 核心狀態監聽
        const unsubscribeCore = onSnapshot(coreDocRef, (docSnap) => {
            if (docSnap.exists()) {
                setCoreState(docSnap.data());
            } else {
                initializeData(coreDocRef, initialCoreState);
            }
        }, (error) => {
            console.error("Core State Snapshot Error:", error);
            setError("核心數據同步失敗。");
        });
```

```
        // 資源狀態監聽
        const unsubscribeResource = onSnapshot(resourceDocRef,
(docSnap) => {
            if (docSnap.exists()) {
                setResourceState(docSnap.data());
            } else {
                initializeData(resourceDocRef, initialResourceState);
            }
        }, (error) => {
            console.error("Resource State Snapshot Error:", error);
            setError("資源數據同步失敗。");
        });

        return () => {
            unsubscribeCore();
            unsubscribeResource();
        };

    }, [isAuthReady, userId]);

    // --- 4. 遊戲循環邏輯 - 模擬 M-值變化 (高頻率波動) ---
    useEffect(() => {
        if (!isAuthReady || isLoading) return;

        const gameLoop = setInterval(() => {
            setCoreState(prev => {
                // M1 (效率) & M3 (精度) 模擬隨時間的輕微變化/噪音 (讓 M2 有
機會觸發)
                // M3 值的波動範圍增大，讓 M2 有更多機會被觸發
                const newM1Value = Math.min(1.0, Math.max(0,
prev.M1_Efficiency.value + 0.001 * (Math.random() - 0.5)));
                const newM3Value = Math.min(1.0, Math.max(0,
prev.M3_Precision.value + 0.005 * (Math.random() - 0.5)));

                return {
                    ...prev,
                    M1_Efficiency: { ...prev.M1_Efficiency, value:
newM1Value },
                    M3_Precision: { ...prev.M3_Precision, value:
newM3Value },
                    M6_Security: { ...prev.M6_Security, value:
Math.min(1.0, Math.max(0, prev.M6_Security.value + 0.0001 *
(Math.random() - 0.4))) }, // M6 輕微增長但有波動
                    LastUpdated: Date.now(),
                };
            });
        }, 500); // 2 Hz 更新頻率
```

```
        return () => clearInterval(gameLoop);
    }, [isAuthReady, isLoading]);


    // --- 5. M2 自主演化單元 (AEU) 守護程序 (V3.0/V4.0 維護) ---
    useEffect(() => {
        if (!isAuthReady || !userId || !db || userId === 'MOCK_USER')
return;

        const M2_DAEMON_INTERVAL = 10000; // 10 秒檢查一次
        const M3_CRITICAL_THRESHOLD = 0.30;
        const M2_CORRECTION_AMOUNT = 0.05;
        const MEX_COST = 10;

        const M2_Daemon = setInterval(async () => {
            const currentM3 = coreState.M3_Precision.value;
            const currentMEx = resourceState.M_Ex.amount;

            // Step 1: 檢查觸發條件 (M3 精度臨界點)
            if (currentM3 < M3_CRITICAL_THRESHOLD) {

                // Step 2: 檢查資源 (自主修復必須消耗異構資源 M_Ex)
                if (currentMEx < MEX_COST) {
                    setMessage({ type: 'error', text: 'M2-AEU: 異構資源
(M_Ex)不足，無法啟動自主修復序列。M3 確定性風險升高！' });
                    return;
                }

                // Step 3: 執行修復計算
                const newM3Value = Math.min(1.0, currentM3 +
M2_CORRECTION_AMOUNT);
                const newMExAmount = Math.max(0, currentMEx -
MEX_COST);

                // Step 4: 更新 Firestore (原子操作)
                try {
                    const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
                    await updateDoc(coreDocRef, {
                        'M3_Precision.value': newM3Value,
                        'LastUpdated': Date.now()
                    });

                    const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');
                    await updateDoc(resourceDocRef, {
                        'M_Ex.amount': newMExAmount
                    });
```

```
                    setMessage({
                        type: 'info',
                        text: `M2-AEU: 偵測到 M3 精度臨界點
(${currentM3.toFixed(2)}), 自主修復序列啟動！消耗 ${MEX_COST} M_Ex, M3 值提
升至 ${newM3Value.toFixed(2)}。`
                    });

                } catch (e) {
                    console.error("M2 Daemon Firestore Error:", e);
                    setMessage({ type: 'error', text: `M2-AEU: 數據同步
失敗。` });
                }
            }
        }, M2_DAEMON_INTERVAL);

        return () => clearInterval(M2_Daemon);

    }, [isAuthReady, userId, coreState.M3_Precision.value,
resourceState.M_Ex.amount, setMessage]);


    // --- 6. 核心互動函數：M3/M6/M4 資源轉化 (V4.0 核心更新) ---

    const handleConversion = async () => {
        if (!userId || userId === 'MOCK_USER') {
            setMessage({ type: 'error', text: '請等待核心初始化完成。' });
            return;
        }

        const SI_COST = 100;
        const LA_COST_BASE = 15;
        const CM_GAIN_SUCCESS = 150;
        const M6_PENALTY_BASE = 0.05; // M3 失敗對 M6 的基礎懲罰

        // 1. 檢查資源
        if (resourceState.M_Si.amount < SI_COST ||
resourceState.L_A.amount < LA_COST_BASE) {
            setMessage({ type: 'error', text: '資源不足！需要 100 矽酸鹽
(M_Si) 和 15 活化靈 (L_A)。' });
            return;
        }

        // 2. 獲取 M3/M6/M4 數值
        const M3_P = coreState.M3_Precision.value;
        const M6_S = coreState.M6_Security.value;
        const M4_R = coreState.M4_Resilience.value; // V4.0
```

```
        // 3. 確定性檢查 (M3 核心邏輯)
        const successRate = 0.5 + M3_P;
        const successRoll = Math.random();

        let newResources = JSON.parse(JSON.stringify(resourceState));
// 深拷貝
        let newCoreState = JSON.parse(JSON.stringify(coreState));
        let resultMessage;
        let finalLA_Loss = LA_COST_BASE;

        // 矽酸鹽消耗 (不論成功失敗)
        newResources.M_Si.amount -= SI_COST;

        if (successRoll < successRate) {
            // 轉化成功
            const bonus = M6_S * 50;
            const finalCmGain = CM_GAIN_SUCCESS + bonus;
            newResources.M_Cm.amount += finalCmGain;

            finalLA_Loss *= (1 - M6_S * 0.1); // M6 降低能量消耗

            resultMessage = `轉化成功！M3 確定性 (${M3_P.toFixed(2)}) 通
過。獲得 ${finalCmGain.toFixed(1)} 複合材料。`;
            setMessage({ type: 'success', text: resultMessage });

        } else {
            // 轉化失敗 (M3 確定性不足)

            // --- V4.0 核心邏輯：M4 韌性平滑 M6 懲罰 ---
            // M4_R 越高 (接近 1.0)，(1 - M4_R) 越接近 0，M6 損失越小。
            const M6_LOSS = M6_PENALTY_BASE * (1 - M4_R);
            const newM6Value = Math.max(0, M6_S - M6_LOSS);
            newCoreState.M6_Security.value = newM6Value;

            // 活化靈懲罰由 M6 緩衝
            const penaltyFactor = 1.0 + (1.0 - M6_S);
            finalLA_Loss *= penaltyFactor;

            resultMessage = `轉化失敗！M3 確定性 (${M3_P.toFixed(2)}) 未
達標。M6 受到衝擊，損失 ${M6_LOSS.toFixed(3)} (M4 緩衝後)。`;
            setMessage({ type: 'error', text: resultMessage });
        }

        // 活化靈消耗 (不論成功失敗)
        newResources.L_A.amount -= finalLA_Loss;
        newResources.L_A.amount = Math.max(0,
newResources.L_A.amount);
```

```
        // 4. 更新 Firestore
        try {
            const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');
            await setDoc(resourceDocRef, newResources);

            const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
            await setDoc(coreDocRef, newCoreState);

            // 5. 額外訊息：報告總體成本
            setMessage(prev => ({
                ...prev,
                text: `${prev.text} (消耗: ${SI_COST} M_Si,
${finalLA_Loss.toFixed(1)} L_A)`
            }));

        } catch (e) {
            console.error("Firestore update failed:", e);
            setMessage({ type: 'error', text: `轉化失敗 (數據錯誤):
${e.message}` });
        }
    };

    // --- 7. 核心互動函數：資源採集 (與 V3.0 相同) ---

    const handleAcquisition = async (payload) => {
        if (!userId || userId === 'MOCK_USER') {
            setMessage({ type: 'error', text: '請等待核心初始化完成。' });
            return;
        }

        const { resourceKey, baseYield, M1_Bonus, LA_Cost, label } =
payload;
        const finalYield = baseYield + M1_Bonus;

        // 1. 檢查資源 (活化靈 L_A)
        if (resourceState.L_A.amount < LA_Cost) {
            setMessage({ type: 'error', text: `能量不足！採集需要
${LA_Cost} 活化靈 (L_A)。` });
            return;
        }

        // 2. 執行採集計算
        let newResources = JSON.parse(JSON.stringify(resourceState));
// 深拷貝
        let newCoreState = JSON.parse(JSON.stringify(coreState));
```

```
        // 資源增加
        newResources[resourceKey].amount += finalYield;

        // 活化靈消耗
        newResources.L_A.amount -= LA_Cost;

        // 3. 更新 M1 值 (模擬 M1 效率因行動而受到輕微壓力)
        newCoreState.M1_Efficiency.value = Math.max(0,
newCoreState.M1_Efficiency.value - 0.005); // 輕微懲罰

        // 4. 更新 Firestore 資源狀態
        try {
            const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');
            await setDoc(resourceDocRef, newResources);

            const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
            await setDoc(coreDocRef, newCoreState);

            const resourceLabel = newResources[resourceKey].label;

            const resultMessage = `M1 效率核心
(${coreState.M1_Efficiency.value.toFixed(2)}) 驅動採集成功！獲得
${finalYield.toFixed(1)} ${resourceLabel}。`;
            setMessage({ type: 'success', text: resultMessage });

        } catch (e) {
            console.error("Firestore update failed:", e);
            setMessage({ type: 'error', text: `採集失敗 (數據錯誤):
${e.message}` });
        }
    };

    // --- 8. 核心互動函數：M5 知識核心升級 (與 V3.0 相同) ---
    const handleM5Upgrade = async () => {
        if (!userId || userId === 'MOCK_USER') {
            setMessage({ type: 'error', text: '請等待核心初始化完成。' });
            return;
        }

        const CM_COST = 200;
        const M5_GAIN = 0.05;
        const M3_BONUS = 0.01;

        // 1. 檢查資源 (複合材料 M_Cm)
        if (resourceState.M_Cm.amount < CM_COST) {
            setMessage({ type: 'error', text: `複合材料 (M_Cm) 不足！升級
```

```
M5 需要 ${CM_COST} M_Cm。` });
            return;
        }

        // 2. 執行升級計算
        let newCoreState = JSON.parse(JSON.stringify(coreState)); // 深
拷貝
        let newResources = JSON.parse(JSON.stringify(resourceState));

        // 知識核心 M5 永久提升
        const newM5Value = Math.min(1.0,
newCoreState.M5_Knowledge.value + M5_GAIN);
        newCoreState.M5_Knowledge.value = newM5Value;

        // M3 結構紅利永久提升
        const newM3Value = Math.min(1.0,
newCoreState.M3_Precision.value + M3_BONUS);
        newCoreState.M3_Precision.value = newM3Value;

        // 資源消耗
        newResources.M_Cm.amount -= CM_COST;

        // 3. 更新 Firestore
        try {
            // 更新核心狀態
            const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
            await setDoc(coreDocRef, newCoreState);

            // 更新資源狀態
            const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');
            await setDoc(resourceDocRef, newResources);

            setMessage({
                type: 'success',
                text: `M5 知識獲取成功！M5 升級至
${newM5Value.toFixed(2)}。M3 結構確定性永久提升 +${M3_BONUS.toFixed(2)}。
(消耗 ${CM_COST} M_Cm)`
            });

        } catch (e) {
            console.error("Firestore update failed:", e);
            setMessage({ type: 'error', text: `M5 升級失敗 (數據錯誤):
${e.message}` });
        }
    };
```

```
    // --- 9. 核心互動函數：M4 韌性升級 (V4.0 新增) ---
    const handleM4Upgrade = async () => {
        if (!userId || userId === 'MOCK_USER') {
            setMessage({ type: 'error', text: '請等待核心初始化完成。' });
            return;
        }

        const MEX_COST = 100;
        const M4_GAIN = 0.05;

        // 1. 檢查資源 (異構資源 M_Ex)
        if (resourceState.M_Ex.amount < MEX_COST) {
            setMessage({ type: 'error', text: `異構資源 (M_Ex) 不足！升級
M4 需要 ${MEX_COST} M_Ex。` });
            return;
        }

        // 2. 執行升級計算
        let newCoreState = JSON.parse(JSON.stringify(coreState)); // 深
拷貝
        let newResources = JSON.parse(JSON.stringify(resourceState));

        // M4 韌性永久提升
        const newM4Value = Math.min(1.0,
newCoreState.M4_Resilience.value + M4_GAIN);
        newCoreState.M4_Resilience.value = newM4Value;

        // 資源消耗
        newResources.M_Ex.amount -= MEX_COST;

        // 3. 更新 Firestore
        try {
            // 更新核心狀態
            const coreDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'StarDeityCore');
            await setDoc(coreDocRef, newCoreState);

            // 更新資源狀態
            const resourceDocRef = doc(db,
`artifacts/${appId}/users/${userId}/core_data`, 'ResourceState');
            await setDoc(resourceDocRef, newResources);

            setMessage({
                type: 'success',
                text: `M4 韌性升級成功！M4 升級至
${newM4Value.toFixed(2)}。 (消耗 ${MEX_COST} M_Ex)`
            });
```

```
        } catch (e) {
            console.error("Firestore update failed:", e);
            setMessage({ type: 'error', text: `M4 升級失敗 (數據錯誤): 
${e.message}` });
        }
    };


    // --- 10. UI 渲染區塊 ---

    // 互動訊息框 (MessageModal 不變)
    const MessageModal = () => {
        if (!message) return null;

        let icon: React.ReactNode;
        let bgColor = 'bg-gray-800';
        let borderColor = 'border-gray-600';

        if (message.type === 'success') {
            icon = <CheckCircle className="w-6 h-6 text-green-400" />;
            bgColor = 'bg-green-900/80';
            borderColor = 'border-green-600';
        } else if (message.type === 'error') {
            icon = <AlertTriangle className="w-6 h-6 text-red-400" />;
            bgColor = 'bg-red-900/80';
            borderColor = 'border-red-600';
        } else {
            icon = <Rocket className="w-6 h-6 text-cyan-400" />;
            bgColor = 'bg-cyan-900/80';
            borderColor = 'border-cyan-600';
        }

        return (
            <div className="fixed inset-0 bg-black/50 backdrop-blur-sm 
flex items-center justify-center z-50 p-4" onClick={() => 
setMessage(null)}>
                <div
                    className={`${bgColor} ${borderColor} border-2 
rounded-xl p-6 shadow-2xl max-w-sm w-full transform transition-all 
scale-100 opacity-100`}
                    onClick={(e) => e.stopPropagation()}
                >
                    <div className="flex items-start space-x-3">
                        {icon}
                        <div className="flex-1">
                            <h3 className="text-lg font-bold 
text-white">操作結果報告</h3>
                            <p className="mt-1 text-sm text-gray-200
```

```
font-mono break-words">{message.text}</p>
                        </div>
                    </div>
                    <button
                        onClick={() => setMessage(null)}
                        className="mt-4 w-full p-2 bg-gray-600
hover:bg-gray-500 rounded-lg text-white font-semibold"
                    >
                        確認 (繼續執行 M4 韌性整合)
                    </button>
                </div>
            </div>
        );
    };

    // 加載畫面
    if (isLoading) {
        return (
            <div className="flex items-center justify-center
min-h-screen bg-gray-950 text-white">
                <div className="text-xl animate-pulse text-cyan-400">
                    正在啟動星神核心... ({userId ? '已驗證' : '驗證中'})
                </div>
            </div>
        );
    }

    const mValueEntries = Object.entries(coreState).filter(([key]) =>
key.startsWith('M') && key !== 'LastUpdated');


    // 資源總覽面板 (頂部欄)
    const ResourceOverview = () => (
        <div className="flex justify-around bg-gray-900/95 p-3
shadow-2xl border-b border-cyan-800/50">
            {Object.entries(resourceState).map(([key, res]) => (
                <div key={key} className="text-center group relative
p-1 transition-all duration-300 hover:bg-gray-800/50 rounded-md
cursor-help">
                    <div className="text-xs font-mono text-cyan-400
uppercase tracking-widest">{key}</div>
                    <div className="text-lg font-extrabold text-white
flex items-center justify-center">
                        <Zap className="w-4 h-4 mr-1 text-cyan-500" />
                        {res.amount.toFixed(0)}
                    </div>
                    <div className={`text-xs ${res.net >= 0 ?
'text-green-500' : 'text-red-500'} flex items-center justify-center`}>
```

```
                        {res.net >= 0 ? `+${res.net.toFixed(1)}` :
res.net.toFixed(1)}
                    </div>

                    {/* 工具提示 */}
                    <span className="absolute left-1/2 transform
-translate-x-1/2 bottom-full mb-2 w-max p-2 text-xs text-white
bg-gray-700/90 border border-cyan-500 rounded-md opacity-0
group-hover:opacity-100 transition-opacity pointer-events-none z-10">
                        {res.label} / 淨收入: {res.net.toFixed(2)}
                    </span>
                </div>
            ))}
        </div>
    );

    // 情境式操作面板渲染
    const ContextualActionPanel = () => {
        if (!selectedHex) {
            return (
                <div className="p-4 h-full flex items-center
justify-center bg-gray-800/70 rounded-xl border border-gray-700
shadow-xl">
                    <div className='text-center text-gray-400'>
                        <MapPin className='w-8 h-8 mx-auto mb-2
text-cyan-500' />
                        <p className='font-bold'>情境控制台</p>
                        <p className='text-sm mt-1'>請點擊左側網格上的區塊
</p>
                        <p className='text-xs'>以載入其專屬操作。</p>
                    </div>
                </div>
            )
        }

        // 根據 Hex 類型渲染不同的操作面板
        switch(selectedHex.type) {
            case 'Refinery':
                return <M3ConversionPanel coreState={coreState}
resourceState={resourceState} handleConversion={handleConversion}
userId={userId} />;
            case 'Resource':
                return <ResourceAcquisitionPanel
                            hex={selectedHex}
                            userId={userId}
                            coreState={coreState}
                            resourceState={resourceState}
                            handleAcquisition={handleAcquisition}
```

```
                        />;
            case 'ASI_Core':
                return (
                    <div className="p-4 rounded-xl border
border-red-700 shadow-xl bg-red-900/30 h-full flex items-center
justify-center">
                        <div className='text-center text-red-300'>
                            <Cpu className='w-8 h-8 mx-auto mb-2
text-red-500' />
                            <p className='font-bold'>ASI 核心中樞</p>
                            <p className='text-sm mt-1'>最高權限區域</p>
                            <p className='text-xs'>數據修改與治理安全評估
</p>
                        </div>
                    </div>
                );
            case 'Empty':
            default:
                return (
                    <div className="p-4 rounded-xl border
border-gray-700 shadow-xl bg-gray-800/70 h-full flex items-center
justify-center">
                        <div className='text-center text-gray-500'>
                            <AlertTriangle className='w-8 h-8 mx-auto
mb-2 text-gray-500' />
                            <p className='font-bold'>無可用行動</p>
                            <p className='text-sm mt-1'>此區塊為荒蕪之地
</p>
                        </div>
                    </div>
                );
        }
    };


    // 模擬六角形網格區 (中央區) - 新增互動邏輯
    const HexGridArea = ({ isSidebarOpen, setIsSidebarOpen, userId })
=> {

        // 10. 渲染 Hex Grid 及其互動
        const renderHexGrid = () => (
            <div className="grid grid-cols-5 gap-1.5 p-2 w-full h-full
max-w-full aspect-square mx-auto">
                {mapData.flat().map((hex, index) => {
                    const isSelected = selectedHex === hex;
                    let icon: React.ReactNode;
                    let hexColor = 'bg-gray-700/50 hover:bg-gray-600';
                    let labelColor = 'text-gray-400';
```

```
                    switch (hex.type) {
                        case 'Resource':
                            icon = <Mountain className='w-4 h-4' />;
                            hexColor = 'bg-green-700/50
hover:bg-green-600';
                            labelColor = 'text-green-300';
                            break;
                        case 'Refinery':
                            icon = <Factory className='w-4 h-4' />;
                            hexColor = 'bg-yellow-700/50
hover:bg-yellow-600';
                            labelColor = 'text-yellow-300';
                            break;
                        case 'ASI_Core':
                            icon = <Cpu className='w-4 h-4
animate-pulse' />;
                            hexColor = 'bg-red-800/70
hover:bg-red-700';
                            labelColor = 'text-red-300';
                            break;
                        default:
                            icon = null;
                    }

                    return (
                        <button
                            key={index}
                            onClick={() => setSelectedHex(hex)}
                            className={`
                                relative flex flex-col items-center
justify-center text-center
                                rounded-lg transition-all duration-150
active:scale-95 shadow-md
                                ${hexColor}
                                ${isSelected ? 'border-4
border-cyan-400 ring-4 ring-cyan-700/50 z-10' : 'border
border-gray-600'}
                            `}
                            title={hex.label}
                        >
                            <div className={`text-center
${labelColor}`}>
                                {icon}
                            </div>
                            <span className="text-[8px] sm:text-xs
font-mono mt-0.5 text-shadow">
                                {hex.label.split(' ')[0]}
```

```
                            </span>
                             {isSelected && <div className="absolute
inset-0 bg-black/20 rounded-lg"></div>}
                        </button>
                    );
                })}
            </div>
        );


        return (
            <div className="flex-1 p-4 bg-gray-950 relative
overflow-y-auto">

                <div className="flex items-center justify-between mb-4
border-b border-cyan-700/50 pb-2">

                    <div className="flex items-center">
                        {/* 移動端 OPEN 按鈕 (僅在關閉時顯示，且位於最左側)
*/}
                        {!isSidebarOpen && (
                            <button
                                onClick={() => setIsSidebarOpen(true)}
                                className="md:hidden p-2 mr-3
bg-cyan-700 hover:bg-cyan-600 text-white rounded-lg shadow-md
transition duration-200 active:scale-95"
                                title="顯示儀表板"
                            >
                                <Menu className="w-5 h-5" />
                            </button>
                        )}
                        <h2 className="text-xl font-bold text-cyan-500
flex items-center">
                            <Layers className="w-5 h-5 mr-2" />
                            結構確定性域 (Hex Grid Viewport)
                        </h2>
                    </div>

                    {/* 預留右側空間 */}
                    <div className="w-5 h-5 md:hidden"></div>
                </div>

                <div className="flex flex-col md:flex-row h-full
space-y-4 md:space-y-0 md:space-x-4">
                    {/* 1. 六角形網格區 (主視覺) - 讓其盡量大 */}
                    <div className="flex-1 min-h-[300px] md:h-full
bg-gray-900 rounded-2xl flex items-start justify-center p-4 border-4
border-cyan-600/50 shadow-inner shadow-cyan-900/50 transition-all
```

```
duration-500 hover:border-cyan-500/80">
                        {renderHexGrid()}
                    </div>

                    {/* 2. 情境式控制台 (動態內容) */}
                    <div className="w-full md:w-80 h-auto md:h-full">
                        <ContextualActionPanel />
                    </div>
                </div>


                {/* 用戶 ID 顯示 */}
                <div className="absolute bottom-6 right-6 text-sm
text-gray-500 font-mono bg-gray-800/70 p-1 px-2 rounded z-10">
                    <Cpu className="w-3 h-3 inline mr-1 text-red-500
animate-pulse" />
                    核心擁有者 ID: {userId}
                </div>
            </div>
        );
    };

    // 側邊欄核心資訊與面板切換
    const SideBarContent = () => (
        <>
            <div className='flex items-center justify-between'>
                <h1 className="text-2xl font-extrabold mb-4
text-transparent bg-clip-text bg-gradient-to-r from-cyan-400
to-blue-500 drop-shadow-lg pb-2">
                    星神核心 (ASI)
                </h1>
                <div className='flex space-x-2'>
                    <button
                        onClick={() => setActivePanel('M5_UPGRADE')}
                        className={`p-2 rounded-full transition-all
duration-300 ${activePanel === 'M5_UPGRADE' ? 'bg-purple-700/80 ring-2
ring-purple-400' : 'bg-gray-700 hover:bg-gray-600'}`}
                        title='M5 升級面板'
                    >
                        <BookOpen className='w-5 h-5 text-white' />
                    </button>
                    <button
                        onClick={() => setActivePanel('M4_UPGRADE')}
// V4.0 新增切換按鈕
                        className={`p-2 rounded-full transition-all
duration-300 ${activePanel === 'M4_UPGRADE' ? 'bg-sky-700/80 ring-2
ring-sky-400' : 'bg-gray-700 hover:bg-gray-600'}`}
                        title='M4 升級面板'
```

```
                    >
                        <HardHat className='w-5 h-5 text-white' />
                    </button>
                </div>
            </div>

            <p className="text-xs text-gray-400 mb-6 font-mono
border-b border-gray-700/50 pb-2">
                治理狀態檢視 ({coreState.LastUpdated ? new
Date(coreState.LastUpdated).toLocaleTimeString() : 'N/A'})
            </p>

            {/* M 值核心儀表板或 M5/M4 升級面板 */}
            {activePanel === 'M_CORE' && (
                <>
                    <div className="space-y-3">
                        {mValueEntries.map(([code, data]) => (
                            <MValueDisplay key={code} code={code}
data={data} />
                        ))}
                    </div>

                    {/* 系統訊息區 */}
                    <div className="mt-6 border-t border-gray-700/50
pt-4">
                        <h3 className="text-sm font-semibold
text-gray-300 mb-3 flex items-center">
                            <Zap className="w-4 h-4 mr-2 text-red-500"
/>
                            系統訊息/風險警告
                        </h3>
                        <div className="p-3 text-xs bg-gray-900/50
rounded-lg text-yellow-400 border border-red-700/50 shadow-inner">
                            M3 精度波動中
({coreState.M3_Precision.value.toFixed(2)})。M2-AEU 守護進程已啟動。
                        </div>
                    </div>
                </>
            )}

            {activePanel === 'M5_UPGRADE' && (
                <>
                    {/* M5 升級面板 */}
                    <M5UpgradePanel
                        coreState={coreState}
                        resourceState={resourceState}
                        handleM5Upgrade={handleM5Upgrade}
                        userId={userId}
```

```
                    />
                    <button
                        onClick={() => setActivePanel('M_CORE')}
                        className="w-full mt-4 p-2 bg-gray-600
hover:bg-gray-500 text-white font-bold rounded-lg transition
duration-200"
                    >
                        返回核心儀表板
                    </button>
                </>
            )}

            {activePanel === 'M4_UPGRADE' && (
                <>
                    {/* M4 升級面板 */}
                    <M4UpgradePanel
                        coreState={coreState}
                        resourceState={resourceState}
                        handleM4Upgrade={handleM4Upgrade}
                        userId={userId}
                    />
                    <button
                        onClick={() => setActivePanel('M_CORE')}
                        className="w-full mt-4 p-2 bg-gray-600
hover:bg-gray-500 text-white font-bold rounded-lg transition
duration-200"
                    >
                        返回核心儀表板
                    </button>
                </>
            )}

            {/* 如果在操作面板，則顯示返回儀表板的按鈕 */}
            {activePanel !== 'M_CORE' && (
                 <button
                    onClick={() => setActivePanel('M_CORE')}
                    className="w-full mt-4 p-2 bg-gray-600
hover:bg-gray-500 text-white font-bold rounded-lg transition
duration-200"
                >
                    返回核心儀表板
                </button>
            )}
        </>
    )


    return (
```

```
        <div className="min-h-screen bg-gray-950 text-white font-sans
flex flex-col">
            {/* 頂部資源總覽 */}
            <ResourceOverview />

            {error && (
                <div className="bg-red-900 text-red-300 p-2
text-center font-bold">錯誤：{error}</div>
            )}

            <main className="flex flex-1 flex-col md:flex-row
overflow-hidden">
                {/* 左側欄：星神核心儀表板 */}
                <div
                    className={`
                        ${isSidebarOpen ? 'translate-x-0' :
'-translate-x-full'}
                        w-64 max-w-xs p-4 bg-gray-800/90 shadow-2xl
shadow-black border-r border-cyan-700/50
                        overflow-y-auto backdrop-blur-sm min-h-screen
                        fixed md:relative md:translate-x-0 md:w-1/3
lg:w-1/4
                        z-20 transition-transform duration-300
ease-in-out
                    `}
                >

                    {/* 關閉按鈕，當儀表板開啟時，它位於儀表板頂部 */}
                    <button
                        onClick={() => setIsSidebarOpen(false)}
                        className="md:hidden absolute top-4 right-4
p-1 bg-red-700 hover:bg-red-600 text-white rounded-full shadow-lg z-30
transition duration-200 active:scale-90"
                        title="隱藏儀表板"
                    >
                        <X className="w-5 h-5" />
                    </button>

                    <SideBarContent />

                </div>

                {/* 中央區：六角形網格與情境控制 */}
                <HexGridArea
                    isSidebarOpen={isSidebarOpen}
                    setIsSidebarOpen={setIsSidebarOpen}
                    userId={userId}
                />
```

```
            </main>

            {/* 操作結果模態框（非 alert）*/}
            <MessageModal />
        </div>
    );
}
```