

```

import React, { useState, useEffect, useReducer } from 'react';
import { RefreshCw, Zap, BookOpen, Shield, TrendingUp, Cpu } from
' lucide-react';

// --- M3 邏輯核心：模擬您的 GameState 和 Reducer ---
// 這是 PoC，用於演示生成的數據如何與您的 Reducer 系統互動

// 模擬 Action Type, 對應您的 GameState 更新
const MValueTypeTreeActionType = {
  SET_SYSTEM_TITLE: 'SET_SYSTEM_TITLE',
  UPDATE_ATTRIBUTES: 'UPDATE_ATTRIBUTES',
  UNLOCK_SKILLS: 'UNLOCK_SKILLS',
  ADD_DECISION_NODES: 'ADD_DECISION_NODES',
};

// 模擬 GameState 的 Reducer 結構
const initialState = {
  systemTitle: "N/A",
  attributes: [],
  skillGroups: [],
  decisionNodes: [],
  isReady: false,
};

const gameReducer = (state, action) => {
  switch (action.type) {
    case MValueTypeTreeActionType.SET_SYSTEM_TITLE:
      return { ...state, systemTitle: action.payload, isReady: true };
    case MValueTypeTreeActionType.UPDATE_ATTRIBUTES:
      // 假設這會更新 PlayerCharacter.stats
      return { ...state, attributes: action.payload };
    case MValueTypeTreeActionType.UNLOCK_SKILLS:
      // 假設這會將新的技能群組加載到 PlayerCharacter.skillTree
      return { ...state, skillGroups: action.payload };
    case MValueTypeTreeActionType.ADD_DECISION_NODES:
      // 假設這會加載到 GameContext.availableDecisions
      return { ...state, decisionNodes: action.payload };
    default:
      return state;
  }
};

// --- M5 知識核心：M-值框架數據 ---
const MValueInfo = {
  A: { id: 'M3/M5', label: 'A: 學院制式學徒', color: 'indigo', icon:
Cpu, desc: '邏輯確定性與知識圖譜優先。' },
  B: { id: 'M1/M6', label: 'B: 繼承傳承學徒', color: 'green', icon:
Shield, desc: '效率與生存韌性優先。' },

```

```

    C: { id: 'M2/M4', label: 'C: 學姊實驗學徒', color: 'red', icon: Zap,
desc: '自主演化與人身風險優先。' },
};

// 模擬 API 響應數據結構
const mockGeneratedData = {
  systemTitle: "精靈法師：禁忌知識與元素熵增體系 (M2/M4 傾向)",
  decisionTree: [
    { node: "遭遇 M2 混沌噪音", M_Value_Core: "M3 vs M2", options:
["執行 M3 校準程序 (高M1能耗, 穩定性+1)", "擁抱 M2 噪音 (穩定性-1, M2/M4 演化
因子+1)"] },
    { node: "資源分配決策", M_Value_Core: "M1 vs M4", options: ["極致
M1 效率煉金 (高產出, M4 爆炸風險)", "M6 安全煉金 (低產出, 零風險)"] }
  ],
  skillTree: [
    { skillGroup: "煉金系", XP_Type: "K-XP", skills: [{ name: "結構
檢測", effect: "分析目標 M3/M5 邏輯漏洞, 提供 +10% 暴擊機率。" }] },
    { skillGroup: "深淵系", XP_Type: "C-XP", skills: [{ name: "混沌
適應", effect: "在 M2/M4 高風險環境下, 每回合恢復 1 點 M6 護盾, 但永久失去 1
點 M3 穩定性。" }] }
  ],
  attributeTree: [
    { attribute: "M1 能耗效率", M_Value_Core: "M1", baseValue: 12,
dependencies: "直接影響技能的魔力消耗, 與 M6 護盾恢復速度互乘。" },
    { attribute: "M3 邏輯壓力", M_Value_Core: "M3/M4", baseValue: 5,
dependencies: "邏輯錯誤、M2 噪音會累積。滿值時導致 M4 人身風險爆發。" }
  ]
};

// --- API Service (M4 風險與 M1 效率模擬) ---

const MValueGeneratorService = {
  // 模擬您的 geminiService, 強制使用 JSON Schema 和 System Prompt
  generateSystemTree: async (concept, philosophy) => {
    // 模擬 API Rate Limiting 和 Task Queue 延遲
    await new Promise(resolve => setTimeout(resolve, 1500));

    // 模擬 API Call
    // 這裡應是實際的 fetch/retry/exponential backoff 邏輯

    // 由於無法調用實際 API, 我們返回預設數據, 但在實際應用中,
    // 這裡的邏輯必須嚴格遵循 systemPrompt 的 M-值指令來生成

    const philosophyData = MValueInfo[philosophy];
    const adjustedTitle =
`$${mockGeneratedData.systemTitle.split('(')[0].trim()}
($${philosophyData.id} 傾向)`;

```

```

        return { ...mockGeneratedData, systemTitle: adjustedTitle };
    }
};

// --- UI Component (React) ---

const App = () => {
    const [gameState, dispatch] = useReducer(gameReducer, initialState);
    const [concept, setConcept] = useState("設計一套圍繞「混沌侵蝕」和「維度外殼生成」的畢業設計挑戰系統。");
    const [philosophy, setPhilosophy] = useState('A');
    const [isLoading, setIsLoading] = useState(false);
    const [error, setError] = useState(null);

    // M3 邏輯: Dispatchers for GameState
    const loadTreeToGameState = (data) => {
        dispatch({ type: MValueTypeActionType.SET_SYSTEM_TITLE, payload: data.systemTitle });
        dispatch({ type: MValueTypeActionType.UPDATE_ATTRIBUTES, payload: data.attributeTree });
        dispatch({ type: MValueTypeActionType.UNLOCK_SKILLS, payload: data.skillTree });
        dispatch({ type: MValueTypeActionType.ADD_DECISION_NODES, payload: data.decisionTree });
        setError(null);
    };

    const handleGenerate = async () => {
        if (!concept.trim()) {
            setError("概念不能為空。");
            return;
        }

        setIsLoading(true);
        setError(null);

        try {
            // 模擬 TaskQueueService 將任務排隊
            const result = await
MValueGeneratorService.generateSystemTree(concept, philosophy);

            // M3 邏輯: 將結果轉換為 Reducer Action 進行狀態更新
            loadTreeToGameState(result);

        } catch (e) {
            // M4 風險處理: 如果 API 失敗, 更新 TaskQueue UI
            console.error("生成任務失敗:", e);
            setError("系統樹生成失敗。請檢查 API 連線或 Task Queue 狀態。");
        }
    };
};

```

```

    } finally {
      setIsLoading(false);
    }
  };

// M3 邏輯: Tree Renderer Component
const TreeRenderer = ({ title, data, type }) => {
  if (!data || data.length === 0) return null;

  let headerClass = '';
  let itemClasses = '';
  let coreKey = '';
  let IconComponent = BookOpen;

  if (type === 'decision') {
    headerClass = 'bg-indigo-700';
    itemClasses = 'bg-indigo-50 border-indigo-600 text-indigo-800';
    coreKey = 'M_Value_Core';
    IconComponent = Cpu;
  } else if (type === 'skill') {
    headerClass = 'bg-green-700';
    itemClasses = 'bg-green-50 border-green-600 text-green-800';
    coreKey = 'XP_Type';
    IconComponent = BookOpen;
  } else if (type === 'attribute') {
    headerClass = 'bg-red-700';
    itemClasses = 'bg-red-50 border-red-600 text-red-800';
    coreKey = 'M_Value_Core';
    IconComponent = Zap;
  }

  return (
    <div className="card shadow-lg flex flex-col h-full">
      <div className={`p-4 ${headerClass} text-white rounded-t-xl flex items-center`}>
        <IconComponent className="w-5 h-5 mr-2" />
        <h3 className="text-xl font-bold">{title}</h3>
      </div>
      <div className="p-4 space-y-4 flex-grow overflow-y-auto">
        {data.map((item, index) => (
          <div key={index} className={`p-3 rounded-lg border-l-4 ${itemClasses}`}>
            <p className="font-semibold">{item.node ||
item.skillGroup || item.attribute}</p>
            <p className="text-xs font-mono opacity-70 mb-2">核心:
{item[coreKey]}</p>

            {item.options && (

```

```

        <ul className="list-disc list-inside mt-1 text-sm ml-2
space-y-1">
            {item.options.map((opt, i) => <li
key={i}><opt></li>)}
        </ul>
    )}
    {item.skills && (
        <div className="text-sm space-y-2">
            {item.skills.map((skill, i) => (
                <p key={i}>
                    <span className="font-bold">{skill.name}</span>
                    <span className="italic ml-2
text-gray-600">{skill.effect}</span>
                </p>
            ))}
        </div>
    )}
    {item.baseValue !== undefined && (
        <div>
            <p className="text-2xl
font-extrabold">{item.baseValue}</p>
            <p className="text-xs mt-1 italic opacity-80">連動:
{item.dependencies}</p>
        </div>
    )}
    </div>
    )}
    </div>
    </div>
    );
};

return (
    <div className="p-4 sm:p-8 min-h-screen bg-gray-50 font-inter">
        <div className="w-full max-w-7xl mx-auto">
            <header className="text-center mb-8">
                <h1 className="text-4xl font-extrabold text-gray-900
mb-2">M-値 TTRPG 系統整合模型</h1>
                <p className="text-gray-600">M5 知識核心應用：生成並整合至
React GameState Reducer 結構。</p>
            </header>

            { /* 控制面板 */ }
            <div className="card p-6 mb-8">
                <h2 className="text-2xl font-semibold mb-4 text-gray-800">
系統配置</h2>

                { /* 哲學選擇 (M3/M5 哲學偏見) */ }

```

```

        <div className="mb-4 p-4 border border-gray-200 rounded-lg
bg-gray-50">
            <p className="font-medium text-gray-800 mb-2 flex
items-center"><BookOpen className="w-4 h-4 mr-2 text-gray-600"/>選擇生成
哲學傾向 (初始學徒卡):</p>
            <div className="grid grid-cols-1 sm:grid-cols-3 gap-3
text-sm">
                {Object.entries(MValueInfo).map(([key, info]) => (
                    <label key={key} className={`flex items-center
p-3 border border-gray-300 rounded-xl cursor-pointer transition
shadow-sm ${philosophy === key ? `ring-2 ring-${info.color}-500
bg-${info.color}-100` : 'bg-white hover:bg-gray-50'}>`>
                        <input type="radio" name="philosophy"
value={key} checked={philosophy === key} onChange={(e) =>
setPhilosophy(e.target.value)} className={`form-radio
text-${info.color}-600 focus:ring-${info.color}-500 h-4 w-4`}/>
                        <span className={`ml-3 font-bold
text-${info.color}-700`>{info.label} ({info.id})</span>
                    </label>
                )))}
            </div>
        </div>

        {/* 概念輸入 */}
        <textarea id="conceptInput" rows="3" className="w-full p-3
border border-gray-300 rounded-lg focus:ring-2 focus:ring-indigo-500
transition duration-150" placeholder="輸入遊戲概念或畢業設計主題..."
value={concept} onChange={(e) =>
setConcept(e.target.value)}></textarea>

        {/* 生成按鈕 (M1 效率核心) */}
        <button
            className={`w-full mt-4 font-bold py-3 rounded-xl
transition duration-300 flex items-center justify-center ${isLoading ?
'bg-gray-400 cursor-not-allowed' : 'bg-indigo-600 hover:bg-indigo-700
text-white'}>`
            <div className="loading-ring mr-3"></div>
            <span id="buttonText">任務已進入 Task
Queue...</span>
        </button>
        {isLoading ? (
            <div className="loading-ring mr-3"></div>
            <span id="buttonText">生成並整合 M-值 系統樹</span>
        ) : (
            <span id="buttonText">生成並整合 M-值 系統樹</span>
        )}
    </div>

```

```

        </button>

        { /* 錯誤/狀態訊息 (M4 風險與 M6 安全) */ }
        {error && <div className="mt-4 p-3 bg-red-100 border
border-red-400 text-red-700 rounded-lg text-sm">{error}</div>}
        </div>

        { /* 輸出區：顯示 GameState 內容 */ }
        {gameState.isReady && (
        <>
            <h2 className="text-3xl font-extrabold text-gray-900
mb-6 border-b pb-2">
                {gameState.systemTitle}
            </h2>

            <div className="grid grid-cols-1 md:grid-cols-3
gap-6">
                <TreeRenderer title="決策樹 (M3 邏輯衝突)"
data={gameState.decisionNodes} type="decision" />
                <TreeRenderer title="技能樹 (M5/M2 知識與演化)"
data={gameState.skillGroups} type="skill" />
                <TreeRenderer title="屬性樹 (M1/M4/M6 核心連動)"
data={gameState.attributes} type="attribute" />
            </div>
        </>
        )}
    </div>
</div>
);
};

export default App;

```