TeacherAssist.AI 漸進式開發順序表

核心原則：邊移動邊開槍

- 每個里程碑都能在瀏覽器展示功能
- 70% 功能完成即可測試，留待重構時段優化
- 模組間並行開發，不必等前一模組完工

第一輪：基礎可運行版本 (1-2週)

目標: 用戶能上傳文件，看到基本的簡報生成結果

1.1 前端基礎框架 (優先度: 最高)

```
frontend/
├── pages/index.js          # 主頁面 - 先實作
├── components/FileUpload.js   # 文件上傳 - 先實作
├── utils/apiClient.js        # API調用 - 先實作
└── styles/globals.css        # 基礎樣式 - 先實作
```

實作順序:

1. `apiClient.js`
   - `uploadFile()` 函式
   - `checkStatus()` 函式
   - 基礎錯誤處理
2. `FileUpload.js`
   - `FileUploadComponent` 類別
   - `handleFileSelect()` 方法
   - `uploadProgress()` 方法
3. `pages/index.js`
   - 基礎頁面結構
   - 整合 FileUpload 組件

1.2 後端文件處理核心 (優先度: 最高)

```
backend/
├── main.py               # FastAPI主程式 - 先實作
├── api/upload.py         # 文件上傳端點 - 先實作
```

```
├── services/file_processor.py # 文件解析 - 先實作
└── utils/text_extractor.py   # 文本提取 - 先實作
```

實作順序:

1. `text_extractor.py`
   - `extract_from_pdf()` 函式 (使用 PyPDF2)
   - `extract_from_docx()` 函式 (使用 python-docx)
   - `extract_from_txt()` 函式
2. `file_processor.py`
   - `FileProcessor` 類別
   - `process_file()` 方法
   - `get_file_content()` 方法
3. `api/upload.py`
   - `/upload` POST 端點
   - 文件驗證邏輯
   - 返回處理結果
4. `main.py`
   - FastAPI 應用初始化
   - 路由註冊
   - CORS 配置

第一輪完成標準: 用戶能上傳文件，看到提取的文本內容

第二輪：簡報生成功能 (接續1-2週)

目標: 用戶能看到實際生成的PPT檔案

2.1 Presenton API 整合 (優先度: 最高)

```
backend/services/presenton_client.py  # 先實作這個檔案
```

實作順序:

1. `PresentonClient` 類別
   - `__init__()` - 基礎配置
   - `generate_presentation()` - 核心生成方法

- `download_result()` - 文件下載方法

2. `api/generate.py`
   - `/generate` POST 端點
   - 調用 Presenton 服務
   - 返回生成結果

2.2 前端結果展示

```
frontend/components/ResultViewer.js   # 新增組件
frontend/pages/result.js              # 新增頁面
```

實作順序:

1. `ResultViewer.js`
   - `PresentationResult` 組件
   - `downloadPPT()` 方法
   - 進度顯示邏輯
2. `pages/result.js`
   - 結果展示頁面
   - 下載按鈕整合

第二輪完成標準: 用戶能生成並下載PPT檔案

第三輪：學科識別功能 (接續1-2週)

目標: 根據內容自動識別學科並應用不同風格

3.1 學科識別邏輯 (優先度: 高)

```
backend/services/subject_detector.py  # 先實作
backend/config/subject_keywords.py    # 同時實作
```

實作順序:

1. `subject_keywords.py`
   - `SCIENCE_KEYWORDS` 字典
   - `SOCIAL_KEYWORDS` 字典
   - `ARTS_KEYWORDS` 字典

2. `subject_detector.py`
   - `SubjectDetector` 類別
   - `detect_subject()` 方法 (關鍵詞匹配)
   - `get_confidence_score()` 方法

3.2 風格適配邏輯

```
backend/services/style_adapter.py    # 後實作
backend/config/templates.json        # 同時實作
```

實作順序:

1. `templates.json`
   - 基礎的學科模板配置
   - Presenton 參數映射
2. `style_adapter.py`
   - `StyleAdapter` 類別
   - `adapt_for_subject()` 方法
   - `get_template_config()` 方法

3.3 前端學科顯示

```
frontend/components/SubjectBadge.js   # 新增組件
```

第三輪完成標準: 系統能自動識別學科並顯示不同風格的PPT

第四輪：用戶體驗優化 (接續1週)

目標: 提升操作流暢度和錯誤處理

4.1 進度追蹤系統

```
backend/services/progress_tracker.py  # 實作進度追蹤
frontend/components/ProgressBar.js    # 實作進度顯示
```

4.2 錯誤處理優化

```
backend/utils/error_handler.py      # 統一錯誤處理
frontend/components/ErrorMessage.js  # 友好錯誤顯示
```

第四輪完成標準: 用戶體驗流暢，錯誤提示清晰

驗證小程式建議

在每個輪次開始前，可以用簡單的console程式驗證核心邏輯：

Console驗證程式1: 文件解析測試

```python
# test_file_extraction.py - 驗證文件解析功能
import sys
from utils.text_extractor import extract_from_pdf, extract_from_docx

def test_extraction(file_path):
    if file_path.endswith('.pdf'):
        content = extract_from_pdf(file_path)
    elif file_path.endswith('.docx'):
        content = extract_from_docx(file_path)

    print(f"提取內容長度: {len(content)}")
    print(f"前200字: {content[:200]}")
    return content

if __name__ == "__main__":
    content = test_extraction(sys.argv[1])
```

Console驗證程式2: 學科識別測試

```python
```

```python
# test_subject_detection.py - 驗證學科識別準確度
from services.subject_detector import SubjectDetector

def test_detection():
    detector = SubjectDetector()

    test_cases = [
        "牛頓第二定律 F=ma 在物理學中的應用",
        "中國古代朝代的政治制度演變",
        "色彩在繪畫中的運用技巧"
    ]

    for text in test_cases:
        result = detector.detect_subject(text)
        print(f"文本: {text[:30]}...")
        print(f"識別結果: {result}")
        print("---")

if __name__ == "__main__":
    test_detection()
```

Console驗證程式3: Presenton API測試

```python
# test_presenton_api.py - 驗證API整合
from services.presenton_client import PresentonClient

def test_api():
    client = PresentonClient()

    test_content = "人工智慧在教育中的應用"

    result = client.generate_presentation(
        content=test_content,
        template="science",
        n_slides=5
    )

    print(f"生成結果: {result}")
    print(f"文件路徑: {result.get('path', 'N/A')}")

if __name__ == "__main__":
    test_api()
```

重構時段安排建議

每完成一輪功能後，安排半天到一天的重構時段：

1. **第一次重構** (第二輪完成後)
   - 整理文件上傳和API調用邏輯
   - 統一錯誤處理方式
   - 優化前端組件結構
2. **第二次重構** (第三輪完成後)
   - 優化學科識別算法
   - 重構服務層架構
   - 改善代碼可讀性和註解
3. **第三次重構** (第四輪完成後)
   - 性能優化
   - 安全性檢查
   - 準備部署配置

開發環境快速設置

```bash
# 後端環境
cd backend
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
pip install fastapi uvicorn python-multipart PyPDF2 python-docx requests

# 前端環境
cd frontend
npm init next-app . --typescript
npm install axios

# 快速啟動
# Terminal 1: uvicorn main:app --reload
# Terminal 2: npm run dev
```

這個順序確保每1-2週就能在瀏覽器看到實際功能進展，同時保持架構完整性。你希望從哪個console驗證程式開始？