TeachAssistant.AI Prototype 架構設計

🎯 Prototype 目標

建立一個可在瀏覽器實際運行的最小可行版本，驗證核心技術可行性：

1. 文件上傳與解析
2. Presenton API 整合
3. 基礎學科識別
4. 簡報生成與下載

🏗️ 精簡三層架構

前端層 (Next.js + TypeScript)

```
prototype-frontend/
├── pages/
│   ├── index.tsx          # 主頁面
│   ├── upload.tsx         # 文件上傳頁面
│   ├── preview.tsx        # 簡報預覽頁面
│   └── download.tsx       # 下載頁面
├── components/
│   ├── FileUploader.tsx     # 文件上傳組件
│   ├── SubjectSelector.tsx  # 學科選擇器
│   ├── ProgressBar.tsx      # 進度顯示
│   ├── PresentationViewer.tsx # 簡報預覽器
│   └── DownloadButton.tsx   # 下載按鈕
├── hooks/
│   ├── useFileUpload.ts     # 文件上傳邏輯
│   ├── useSubjectDetection.ts # 學科識別邏輯
│   └── usePresentationGen.ts  # 簡報生成邏輯
├── utils/
│   ├── fileParser.ts        # 前端文件解析
│   ├── apiClient.ts         # API 調用客戶端
│   └── errorHandler.ts      # 錯誤處理
└── styles/
    ├── globals.css          # 全局樣式
    └── components.css       # 組件樣式
```

中間層 (FastAPI + Python)

```
prototype-backend/
├── main.py              # FastAPI 主應用
├── api/
```

```
│   ├── __init__.py
│   ├── upload.py          # 文件上傳端點
│   ├── analyze.py         # 內容分析端點
│   ├── generate.py        # 簡報生成端點
│   └── download.py        # 文件下載端點
├── services/
│   ├── __init__.py
│   ├── file_processor.py    # 文件處理服務
│   ├── subject_detector.py  # 學科識別服務
│   ├── presenton_client.py  # Presenton API 客戶端
│   └── content_enhancer.py  # 內容增強服務
├── models/
│   ├── __init__.py
│   ├── file_models.py       # 文件數據模型
│   ├── subject_models.py    # 學科數據模型
│   └── presentation_models.py # 簡報數據模型
├── utils/
│   ├── __init__.py
│   ├── text_processor.py    # 文本處理工具
│   ├── subject_keywords.py  # 學科關鍵詞庫
│   └── logger.py            # 日誌工具
└── config/
    ├── __init__.py
    ├── settings.py          # 配置文件
    └── subjects.json        # 學科配置
```

數據層 (簡化版)

```
prototype-data/
├── storage/
│   ├── uploads/           # 用戶上傳文件
│   ├── processed/         # 處理後文件
│   ├── generated/         # 生成的簡報
│   └── templates/         # 學科模板
├── configs/
│   ├── subject_keywords.json  # 學科關鍵詞
│   ├── templates.json         # 模板配置
│   └── presenton_settings.json # Presenton 配置
└── cache/
    ├── file_analysis/       # 文件分析緩存
    └── api_responses/       # API 響應緩存
```

🔧 核心技術實現策略

1. 文件處理 (簡化但實用)

```python
# 支援的文件格式與處理方式
FILE_PROCESSORS = {
    '.pdf': 'PyPDF2 + pdfplumber',    # PDF 文本提取
    '.docx': 'python-docx',           # Word 文檔處理
    '.txt': 'built-in',               # 純文本讀取
    '.md': 'markdown parser'          # Markdown 解析
}
```

2. 學科識別 (關鍵詞 + 規則)

```python
# 簡化但有效的學科識別
SUBJECT_DETECTION = {
    'method': 'keyword_matching + content_analysis',
    'accuracy_target': '80%+',
    'processing_time': '<1 second',
    'fallback': 'general_template'
}
```

3. Presenton 整合 (直接 API 調用)

```python
# Presenton API 整合策略
PRESENTON_INTEGRATION = {
    'api_endpoint': 'localhost:5000/api/v1/ppt/presentation/generate',
    'enhancement_layer': 'post_processing',
    'template_mapping': 'subject_to_presenton_template',
    'error_handling': 'graceful_fallback'
}
```

📋 驗證計劃

Week 1: 基礎設施搭建

我負責撰寫:

1. Next.js 基礎框架 + TypeScript 配置
2. FastAPI 基礎框架 + CORS 配置

3. 文件上傳功能 (前後端)
4. 基礎的文件解析功能

你負責測試:

1. 在本地環境運行前後端
2. 測試文件上傳功能
3. 驗證不同格式文件的解析結果
4. 識別潛在問題

Week 2: 核心功能整合

我負責撰寫:

1. 學科識別邏輯 (關鍵詞匹配)
2. Presenton API 客戶端
3. 簡報生成流程
4. 前端預覽和下載功能

你負責測試:

1. 使用不同學科的文件測試識別準確度
2. 驗證 Presenton API 整合是否正常
3. 測試生成的簡報品質
4. 在不同瀏覽器測試兼容性

🚀 Prototype 部署計劃

本地開發環境

```bash
# 前端 (Port 3000)
cd prototype-frontend
npm install
npm run dev

# 後端 (Port 8000)
cd prototype-backend
pip install -r requirements.txt
uvicorn main:app --reload

# Presenton (Port 5000)
docker run -p 5000:5000 presenton/api
```

測試環境準備

1. **測試文件準備**: 收集不同學科的教學文件 (PDF, Word, TXT)
2. **瀏覽器測試**: Chrome, Firefox, Safari, Edge
3. **性能基準**: 文件處理時間, API 響應時間, 內存使用量
4. **錯誤場景**: 大文件, 格式錯誤, 網絡中斷等

📊 成功指標

技術指標

- [ ] 支援 PDF, DOCX, TXT 文件上傳和解析
- [ ] 學科識別準確率 > 80%
- [ ] 簡報生成成功率 > 95%
- [ ] 平均處理時間 < 30 秒
- [ ] 在主流瀏覽器正常運行

用戶體驗指標

- [ ] 操作流程直觀易懂
- [ ] 錯誤提示清晰有用
- [ ] 生成的簡報品質可接受
- [ ] 下載功能正常運作

🔄 迭代改進計劃

基於 Prototype 測試結果，我們可以：

1. 識別架構瓶頸和改進點
2. 調整技術選型和實現策略
3. 優化用戶體驗流程
4. 制定詳細的 MVP 開發計劃

下一步行動: 我準備開始撰寫 Prototype 的核心組件。你希望我從哪個部分開始？

1. 前端文件上傳組件？
2. 後端文件處理服務？
3. 學科識別邏輯？
4. 或者你有其他優先級建議？