

****

PRD｜AI 学习助手

（PDF Study Assistant）v1.0

目录

[PRD｜AI 学习助手 1](#_Toc207100568)

[（PDF Study Assistant）v1.0 1](#_Toc207100569)

[1. 背景与目标 3](#_Toc207100570)

[1.1 背景（Problem Statement） 3](#_Toc207100571)

[1.2 产品定位 3](#_Toc207100572)

[1.3 目标（v1.0，1 周可交付） 4](#_Toc207100573)

[1.4 范围（In / Out） 4](#_Toc207100574)

[1.5 关键假设与限制 4](#_Toc207100575)

[2. 用户需求（用户故事） 5](#_Toc207100576)

[2.1 用户画像 5](#_Toc207100577)

[2.2 用户故事 5](#_Toc207100578)

[2.3 用户需求优先级（MoSCoW） 5](#_Toc207100579)

[2.4 非功能性需求（v1.0） 6](#_Toc207100580)

[3. 产品功能设计 6](#_Toc207100581)

[3.1 功能概览 6](#_Toc207100582)

[3.2 核心功能模块 7](#_Toc207100583)

[3.2.1 文件上传模块 7](#_Toc207100584)

[3.2.2 内容解析模块 7](#_Toc207100585)

[3.2.3 总结生成模块 7](#_Toc207100586)

[3.2.4 结果展示模块 7](#_Toc207100587)

[3.3 非功能性需求 8](#_Toc207100588)

[3.3.1 性能要求 8](#_Toc207100589)

[3.3.2 易用性要求 8](#_Toc207100590)

[3.3.3 安全性要求 8](#_Toc207100591)

[3.3.4 兼容性要求 8](#_Toc207100592)

[4. 系统架构设计 8](#_Toc207100593)

[4.1 架构设计思路 8](#_Toc207100594)

[4.2 系统总体架构 9](#_Toc207100595)

[4.3 技术选型 9](#_Toc207100596)

[4.4 模块交互与流程 9](#_Toc207100597)

[4.5 稳定性与扩展性设计 10](#_Toc207100598)

[5. 迭代与规划 10](#_Toc207100599)

[5.1 短期迭代（v1.1 ~ v1.2） 10](#_Toc207100600)

[5.2 中期迭代（v2.0） 10](#_Toc207100601)

[5.3 长期发展愿景 11](#_Toc207100602)

**AI 学习助手**

**（PDF Study Assistant）**

v1.0

# 1. 背景与目标

## 1.1 背景（Problem Statement）

在我暑期的教研实习经历中，我主要参与了初高中学科讲义的编辑与整理工作。过程中发现，教师和教研人员在日常工作中经常需要从各类 PDF 学习资料（如试卷、讲义、习题集）中提取关键信息，用于二次加工和教学应用。

然而，传统的处理方式通常依赖人工复制和整理，效率低且容易出错。尤其是在需要快速定位重点内容、提炼知识结构或对资料进行再利用时，手工操作的局限性尤为突出。

基于这一痛点，我尝试结合 Python 与 AI 技术开发“AI 学习助手 v1.0”。

该工具目前能够对单份 PDF 文档进行处理，自动完成文本提取与重点信息整理，帮助用户更高效地理解和利用资料。虽然在功能上尚未支持批量处理或转为 Excel，但这一版本已初步验证了 AI 在教育资料处理中的可行性与价值。

## 1.2 产品定位

本项目旨在开发一款 **本地轻量级 AI 学习小工具**，以“上传 PDF → 一键生成考点总结”为核心功能，帮助用户快速从学习资料中提取重点内容。

* **定位**：聚焦单份文档的快速处理与重点提炼，避免复杂配置和后端依赖，提供简单、直接的使用体验。
* **目标用户**：以 **大学生与考研生** 为核心人群，满足其日常复习与资料整理的需求，同时兼顾 **教师与助教** 在备课、讲义归纳中的使用场景。
* **形态**：工具以 **本地运行 + 网页化 UI（Gradio）** 的形式呈现，无需额外服务器部署；核心功能调用 **通义千问（DashScope）** 实现生成式总结，确保轻量化与智能化兼顾

## 1.3 目标（v1.0，1 周可交付）

**功能目标**  
实现单份 PDF 文件的上传、文本抽取、考点总结与结果展示，形成完整的功能闭环。

**体验目标**  
确保用户从打开工具到获取总结的流程 **≤ 3 步**，操作直观，降低学习成本。

**性能目标**  
在 **10 页以内、纯文本型 PDF** 的场景下，点击生成后 **≤ 20 秒** 输出考点总结，满足流畅交互需求。

**稳定性目标**  
对常见失败场景（如：未选择文件、文件类型错误、文本抽取失败、模型调用异常）提供明确提示，避免出现“无响应”或系统卡死的情况。

**可演示目标**  
交付物需包含：

**演示截图：**至少两份（上传前界面、生成后总结界面），突出交互闭环。

**演示视频：**1 段约 30 秒，完整演示“上传 → 生成 → 总结”流程，保证 UI 简洁、操作顺畅。

**交付要求：**截图与视频需可直接作为面试或项目展示素材（无需二次加工）

## 1.4 范围（In / Out）

* **In（v1.0）**：
  + 单 PDF 文件上传；
  + 文本解析（基于 PyPDF2）；
  + 调用通义千问（DashScope，qwen-turbo）生成总结；
  + 网页化界面展示总结，并支持一键复制
* **Out（v1.0）**：
  + 批量文件处理；
  + OCR（图片型/扫描型 PDF 解析）；
  + 对话式问答交互；
  + 基于检索增强（RAG）的深度学习支持。

（以上功能留待后续迭代版本评估与实现）

## 1.5 关键假设与限制

* 输入 PDF **为文本型**（非扫描图片）；若为空或提取失败，则提示用户。
* 模型：使用 **qwen-turbo**（或同等推理类）进行摘要；网络需可访问阿里云。
* 隐私：**本地 UI + 云端推理**；不做长期存储，提示用户对涉密文件谨慎使用。

# 2. 用户需求（用户故事）

## 2.1 用户画像

* **大学生/考研生**
  + 学习任务重，需要快速整理课堂讲义、PDF资料，提炼重点用于复习。
  + 技术能力有限，期望工具“即开即用”，操作门槛低。
* **教研实习生/老师助教**
  + 需要整理多份资料，形成归纳总结，用于备课或辅导。
  + 注重输出的“条理性”和“可演示性”，希望能直接拿来作为教学辅助。

## 2.2 用户故事

|  |  |  |  |
| --- | --- | --- | --- |
| 用户身份 | 场景（When） | 动机（Why） | 目标（Then） |
| 大学生 | 在复习考试前，需要整理课堂 PDF 讲义 | 不想花大量时间手动划重点 | 上传讲义 → 一键生成考点总结 |
| 考研生 | 在短时间内浏览多章节 PDF 资料 | 希望快速抓住核心知识点 | 工具能在几秒内生成简要总结 |
| 教研实习生 | 在准备讨论课或教案时，需要整合课程资料 | 提高备课效率，避免重复劳动 | 上传资料 → 自动生成结构化总结 |
| 老师助教 | 在批改或辅导过程中，遇到无法解析的 PDF | 不希望因错误卡住进度 | 工具给出清晰报错提示，方便快速解决 |

## 2.3 用户需求优先级（MoSCoW）

* **Must Have（必须有）**
  + 支持单份 PDF 上传。
  + 成功解析文本并生成总结。
  + 明确提示常见错误（非 PDF / 空文件 / 解析失败 / 模型报错）。
* **Should Have（最好有）**
  + 结果可以一键复制，方便粘贴到笔记或教案。
  + 界面简洁，步骤不超过 3 步。
* **Could Have（可以有）**
  + 自定义总结字数（简要版 / 详细版）。
  + 对总结结果进行简单格式优化（分点展示）。
* **Won’t Have（暂不考虑）**
  + 批量文件处理。
  + OCR 图片解析。
  + 对话式问答。

**用户需求总结矩阵**

| **用户身份** | **高优先级需求 🔴** | **中优先级需求 🟡** | **低优先级需求 ⚪** |
| --- | --- | --- | --- |
| **大学生** | 一键生成考点总结 | 快速复制结果 | 美观排版导出 |
| **考研生** | 跨章节总结整合 | 深度分析与扩展 | 历年真题对照 |
| **实习生** | 结构化总结（方便讲义编辑） | 教案内容框架化 | 导出为 Word/PPT |

**🔑 图例**

* 🔴 **高优先级**：v1.0 必须覆盖的核心需求（上传 PDF → 一键总结 → 结果可读）
* 🟡 **中优先级**：v1.x 可迭代优化（如更灵活的总结方式）
* ⚪ **低优先级**：未来方向（如导出/批量处理）

## 2.4 非功能性需求（v1.0）

* **可用性**：3 步完成核心流程；中文界面与提示文案。
* **性能**：10 页以内文本型 PDF，20 秒内返回；超时给到提示并允许重试。
* **稳定性**：常见异常均有可恢复路径（返回首页/重新上传/重试）。
* **隐私**：不落盘存储原文与结果（除非用户自行复制/另存）；在 README 中明确声明。

# 3. 产品功能设计

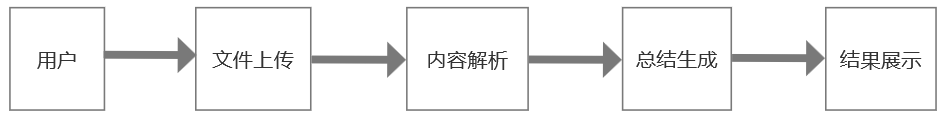
## 3.1 功能概览

本产品旨在提供一个 **本地轻量级学习小工具**，帮助大学生、考研生及教研人员快速完成文档要点提炼。整体功能遵循 **“上传 PDF → 自动生成考点总结 → 展示结果”** 的闭环，最大化降低用户的操作负担。

系统功能结构可分为四大模块：

1. **文件上传模块**：支持用户将单份 PDF 文档上传至系统，作为后续总结的输入。
2. **内容解析模块**：自动提取 PDF 文档中的纯文本内容，忽略图片、表格等非文本信息。
3. **总结生成模块**：调用通义千问（DashScope）API 对解析文本进行处理，生成结构化的考点总结。
4. **结果展示模块**：在网页化界面（Gradio）中实时显示总结结果，并提供复制、清空等轻量交互。

功能整体框架如下：



该工具定位为 **“一键式 PDF 总结助手”**，突出简洁、快速、无需部署的使用体验。

## 3.2 核心功能模块

### 3.2.1 文件上传模块

* **输入格式**：支持单份 PDF 文件（不支持 Excel、Word、图片格式）。
* **交互方式**：用户通过网页化 UI（Gradio 界面）进行上传。
* **约束条件**：文件大小建议不超过 20MB；暂不支持多文件批量上传。
* **目标体验**：用户只需完成一次点击上传操作，即可进入自动处理流程。

### 3.2.2 内容解析模块

* **解析方式**：对 PDF 文档进行逐页解析，提取可识别的文字信息；忽略表格、图片、公式等非文本元素。
* **数据清洗**：去除空白页、乱码字符、无效符号。
* **输出结果**：得到结构化的纯文本内容，为后续总结提供输入。
* **目标体验**：确保输出内容简洁、准确，避免用户手动复制粘贴。

### 3.2.3 总结生成模块

* **调用服务**：接入通义千问（DashScope）API，基于大模型进行自然语言总结。
* **总结逻辑**：将解析文本分段送入模型，生成符合学习场景的要点总结；总结内容以条列式或分层结构呈现。
* **容错处理**：当 API 调用失败时，提示用户“生成失败，请重试”。
* **目标体验**：让用户一键即可获得完整考点总结，避免冗余回答。

### 3.2.4 结果展示模块

* **展示方式**：在 Gradio 界面中实时呈现总结结果。
* **辅助功能**：提供复制总结、一键清空等功能，便于用户整理或二次编辑。
* **目标体验**：轻量、直观、可直接用于学习和备课。

## 3.3 非功能性需求

### 3.3.1 性能要求

* **响应速度**：在正常网络环境下，完成“上传 → 生成总结”全流程应控制在 **30 秒内**（不含大文件情况）。
* **处理能力**：支持处理 **20MB 以下的 PDF 文件**，可稳定完成文字解析与总结生成。
* **可扩展性**：预留支持多文件批量处理和多语言文档的能力。

### 3.3.2 易用性要求

* **操作简便**：用户只需完成 **三个操作步骤**（上传 → 点击生成 → 查看总结），无需额外配置。
* **界面友好**：采用 **Gradio 网页化界面**，直观展示上传按钮与总结结果。
* **结果可复制**：支持用户一键复制总结内容，方便粘贴到学习笔记或讲义中。

### 3.3.3 安全性要求

* **本地运行**：工具运行在用户本地环境中，不依赖后端服务，保障数据安全。
* **数据隐私**：上传的 PDF 不会被保存或二次传输，解析内容仅用于即时总结。
* **调用安全**：调用 DashScope API 时，需对 API Key 做本地化配置，避免泄露。

### 3.3.4 兼容性要求

* **运行环境**：支持在 Windows 10/11 与主流 Linux 环境下运行。
* **依赖管理**：通过 requirements.txt 管理 Python 依赖，支持一键安装。
* **浏览器支持**：结果展示兼容 Chrome、Edge 等现代浏览器。

# 4. 系统架构设计

## 4.1 架构设计思路

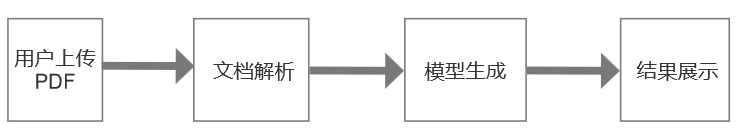
AI 学习助手 v1.0 定位为 **本地轻量级学习工具**，架构设计以 **简洁、易实现、可演示** 为原则。

* **轻量本地化**：不依赖后端部署，前端通过 Gradio 提供网页化交互，便于快速演示。
* **模块化分层**：拆解为「文件解析 → 模型调用 → 结果展示」三层，清晰独立，方便后续扩展。
* **可扩展性**：保留接口能力，未来可逐步加入 OCR、批量处理、结构化导出等功能。

## 4.2 系统总体架构

系统整体由四个核心模块构成：

1. **用户交互层（Gradio Web UI）**
   * 提供简洁界面，支持 PDF 文件上传、生成按钮点击、结果展示与复制。
   * 特点：本地网页运行，打开即用，操作 ≤ 3 步。
2. **文档解析层（pdfplumber/PyPDF2）**
   * 输入：用户上传的 PDF 文件。
   * 处理：提取文本内容，过滤非文本/空白页。
   * 输出：清洗后的原始文本。
3. **模型调用层（DashScope API / qwen-turbo）**
   * 输入：解析后的文本。
   * 处理：调用通义千问模型，生成考点总结。
   * 输出：结构化的要点总结文本。
4. **结果展示层（Gradio 输出区）**
   * 输入：模型返回的总结内容。
   * 展示：在网页端输出整洁文本，支持复制。
   * 错误处理：文件为空、格式错误或模型报错时，均有明确提示。

📌 **系统架构图**  


## 4.3 技术选型

* **前端交互**：Gradio → 快速搭建本地 Web 界面，降低上手门槛。
* **文件解析**：pdfplumber（优先，兼容性较好）/PyPDF2（备选）。
* **模型调用**：DashScope SDK，模型选用 qwen-turbo，满足轻量推理需求。
* **运行环境**：Python 3.10+，本地运行即可，无需复杂部署。

## 4.4 模块交互与流程

1. 用户上传 PDF 文件；
2. 系统调用文档解析模块提取文本；
3. 若解析失败（空文件/扫描件），返回错误提示；
4. 若解析成功，则调用 DashScope 模型生成总结；
5. 将生成的总结文本在界面上展示；
6. 用户可直接复制总结内容用于学习或备课。

## 4.5 稳定性与扩展性设计

* **稳定性**：所有关键节点均设计异常捕获与用户提示，避免“无响应”状态。
* **扩展性**：
  + 短期：支持批量上传、多文件合并。
  + 中期：支持 OCR，处理扫描件 PDF。
  + 长期：支持结构化导出（Excel/Markdown）、检索增强（RAG）、对话式问答。

# 5. 迭代与规划

为了使 AI 学习助手更好地服务于教育与学习场景，本项目在完成初始版本（v1.0）的基础上，制定了循序渐进的迭代规划。整体演进分为短期优化、中期功能拓展和长期发展愿景三个阶段，确保产品既能快速落地，又具备持续演进的潜力。

## 5.1 短期迭代（v1.1 ~ v1.2）

在初始版本的基础上，短期目标聚焦于**提升稳定性与用户体验**，确保工具能在常见教学和学习场景中可靠运行。具体优化方向包括：

1. **文件兼容性优化**
   * 增强对不同类型 PDF 文件的解析能力，包括部分复杂版式或带有图片的 PDF。
   * 改进错误处理机制，确保在解析失败时不影响系统运行。
2. **容错与提示**
   * 当用户上传非 PDF 文件或解析失败时，提供清晰的错误提示与解决建议。
   * 增强交互反馈，降低普通用户的使用门槛。
3. **界面与交互优化**
   * 改善 Gradio 界面细节，优化上传区域和结果展示的排版样式。
   * 提升操作流畅度，让用户更直观地完成“上传 → 生成 → 总结”的闭环。
4. **性能优化**
   * 在调用大模型时减少延迟，提高单份文件处理效率。
   * 优化代码结构，保证系统稳定性。

## 5.2 中期迭代（v2.0）

在完成基本可用的基础上，中期迭代聚焦于**扩展功能与实用性**，为更复杂的教学场景提供支持。重点包括：

1. **OCR 支持**
   * 集成 Tesseract 或 PaddleOCR，实现对扫描版 PDF 的文字识别。
   * 扩大系统的适用范围，支持更多样化的学习资料。
2. **批量处理能力**
   * 支持一次上传多个 PDF 文件，实现批量总结生成。
   * 提升处理效率，满足教师批量整理资料的需求。
3. **多格式导出**
   * 除在界面中展示结果外，支持将总结导出为 **Word / Markdown**。
   * 方便用户对总结内容进行后续编辑和存档。
4. **多模型接入**
   * 除通义千问外，支持接入更多 API 模型（如 GPT 系列）。
   * 兼容部分本地开源模型（如 ChatGLM），提升灵活性。

## 5.3 长期发展愿景

长期目标是打造一个**智能化、个性化的学习助手**，不仅具备文档总结能力，还能深度服务于学习全过程。愿景包括：

1. **交互式学习助手**
   * 除总结功能外，支持用户基于总结内容进行问答（QA 模式）。
   * 提供学习引导，而不仅仅是被动的结果输出。
2. **检索增强生成（RAG）能力**
   * 支持长文档拆分与知识库构建。
   * 通过 RAG 技术提升总结与问答的准确性和上下文连贯性。
3. **个性化学习支持**
   * 根据用户身份（如学生/教师），输出定制化总结：
     + 学生版侧重知识点梳理与重点难点提示；
     + 教师版则偏向教学讲义与考点提炼。
4. **学习数据沉淀**
   * 支持用户保存历史总结，逐步形成个人知识档案库。
   * 打造个性化的学习成长轨迹。