

# 软件需求说明: 基于 T<sub>E</sub>X 的题库整理与试卷构建系统 (0.1 版本)

Mathlang 组织

2019 年 11 月 19 日

## 目录

<b>1</b>	<b>修订历史</b>	<b>3</b>
<b>2</b>	<b>简介</b>	<b>3</b>
2.1	项目目的	3
2.2	文档约定	3
2.3	目标读者与阅读建议	4
2.4	项目范围	4
2.5	参考资料	4
<b>3</b>	<b>总体描述</b>	<b>4</b>
3.1	产品总览	4
3.2	产品功能	5
3.3	用户类别及其特征	5
3.4	操作环境	5
3.5	产品设计及其实现中的约束	6
3.6	用户文档	6
3.7	假设与依赖	6
<b>4</b>	<b>外部接口的要求</b>	<b>6</b>
4.1	用户接口	6
4.2	硬件接口	6
4.3	软件接口	6
4.4	通信接口	7
<b>5</b>	<b>系统其他非功能需求</b>	<b>7</b>
5.1	系统性能需求	7
5.2	系统安全需求	7
5.3	系统保密需求	7

目录	2
5.4 系统质量指标 . . . . .	7
5.5 授权规则 . . . . .	8
5.5.1 教育授权 . . . . .	8
5.5.2 商业授权 . . . . .	8

## 1 修订历史

维护者	时间	变更日志	版本
梁钰栋	2019/11/19	<ul style="list-style-type: none"> <li>初步定稿软件需求说明 (本地版本)</li> <li>在 <a href="#">GitHub</a> 新建仓库托管代码</li> </ul>	0.1

## 2 简介

### 2.1 项目目的

基于  $\text{T}_{\text{E}}\text{X}$  的题库整理与试卷构建系统 (下称本项目) 旨在将试卷的格式与数据分离 (即分离为前端, 后端及数据库): 使用 `Json` 数据格式或数据库 (如 `SQL`) 来存储题目数据; 使用  $\text{T}_{\text{E}}\text{X}$  系统对题目进行排版, 数据与版式之间通过 `Python` 编程语言操作数据库以生成  $\text{T}_{\text{E}}\text{X}$  源代码.

本项目的思想来自于使用特定工具解决特定的问题, 同时从工程的角度审视问题, 寻求易于拓展的解决方案: **(1)**  $\text{T}_{\text{E}}\text{X}$  擅长排版, 但用作运算 (如计数器的数学运算等) 的用户接口较弱, 而支持嵌入式脚本的原生引擎 `Lua1` $\text{T}_{\text{E}}\text{X}$  则需要更复杂的逻辑实现, 不利于以后对程序进行改进与维护; **(2)** `Python` 为“胶水语言”, 功能库众多, 可以根据特定需求做特定事情 (比如从数据库筛选题目, 制作网页应用等), 但是排版所用库 (例如 `PyLaTeX`, `Python-docx`) 提供的功能不如  $\text{T}_{\text{E}}\text{X}$  丰富且学习成本较大; **(3)** 数据库仅用来存储数据与筛选数据, 所以在本项目中不能单独使用. 而 `Python` 存在 `ORM` (`Object Relational Mapping`, 例如 `SQLAlchemy1`), 可用来操作数据库. 但是本项目一开始是作为本地版本开发, 数据应追求可读性, 所以使用纯文本记录数据 (例如数据格式 `Json`) 可以更方便地修改数据.

### 2.2 文档约定

“基于  $\text{T}_{\text{E}}\text{X}$  的题库整理与试卷构建系统”的文档 (下称本文档) 参考 [Wikipedia<sup>2</sup>](#) 与 [GitHub<sup>3</sup>](#) 的提示进行编写, 其中文档采用  $\text{T}_{\text{E}}\text{X}$  系统进行排版, 除数据外的代码等部分均托管在 [GitHub](#) 上. 需要注意的是, 本文档的源代码中包含大量的交叉引用, 如无必要, 请勿对文件进行复制粘贴, 直接转载到其他网站上.

名词	解释
本地版本	使用 $\text{T}_{\text{E}}\text{X}$ , <code>Python</code> , <code>Json</code> 的版本, 旨在在本地进行题库的管理, 后期可能会在在线版本中提供题库上传与导出的功能, 用以管理个人的题库.
在线版本	使用 $\text{T}_{\text{E}}\text{X}$ , <code>Python</code> , <code>SQL</code> 的版本, 旨在在网页进行题库的管理, 后期可能会在在线版本中提供题库上传与导出的功能, 用以管理团体的题库, 同时在线合作完善题库.
版本控制	对软件开发过程中各种程序代码, 配置文件及说明文档等文件变更的管理.
$\text{T}_{\text{E}}\text{X}$	本项目使用的 $\text{T}_{\text{E}}\text{X}$ 发行版为 <code>T<sub>E</sub>XLive 2019</code> , 除此之外的所有发行版 (例如 <code>C<sub>T</sub>E<sub>X</sub></code> ) 均不保证模板的正常运行, 更多资料请参考 <a href="#">Read the Docs</a> .
<code>SQLAlchemy</code>	托管在 <a href="#">PyPi</a> 上的 <code>Python</code> 库, 具体版本需要见程序中的依赖声明 ( <code>requirements.txt</code> , <code>Pipfile</code> 等), 此外在本地版本中不会使用.

## 2.3 目标读者与阅读建议

本文档的目标读者为本项目的开发人员及在初期使用本项目的用户。以下将按照不同的读者类型分别列举其阅读建议。此外,如果您不属于此处列举的目标读者,但是仍想获取阅读建议,请及时提出 [Issue](#),我们会在注意到后及时补充至本文档中。

读者类型	阅读建议
开发人员	建议着重阅读第 4 节, 第 ?? 节及第 5 节, 如果对本文档存在疑问请及时提出 <a href="#">Issue</a> .
初期用户	建议浏览第 2 节及第 3 节, 如果对本文档存在疑问请及时提出 <a href="#">Issue</a> .

## 2.4 项目范围

本项目重点为题库整理与试卷构建。在前期项目 [Exam](#) 中,我们完全使用  $\text{T}_\text{E}\text{X}$  编写题库,通过  $\text{T}_\text{E}\text{X}$  的 `\input` 机制将不同时间的题目分离,与此同时产生一系列问题: (1) 题目看似分离,但是在根据已有题目来构建新的试卷时,还需要将  $\text{T}_\text{E}\text{X}$  的冗余部分重新书写一遍; (2) 为了追求注释部分 `\input` 后仍可以运行,导言区需要将所有的依赖记录下来,从而造成性能上的损失; (3) 前期项目提供众多的 API (尤其解题框),构建题库的时间跨度较大导致使用 API 不统一。

而本项目在设计时改进了上述的缺点,即将格式与数据完全分离。题库存储的时候不需要考虑  $\text{T}_\text{E}\text{X}$  的格式,试卷构造的时候会针对不同的题目进行导言区的构造(耗费时间可以忽略不计,  $\text{T}_\text{E}\text{X}$  宏包加载的时间远大于程序运行的时间),在提高题库自定义化的同时,更好地管理题库的数据。

如果后期添加了在线版本的支持,则可以将用户们的题库进行整合,在标签明确的前提下,可以快速构建试卷。同时,可以邀请一部分与试卷无利害关系的用户,对试卷的内容解析等进行更好地优化,同时记录修改的历史记录,用以量化用户的贡献。

## 2.5 参考资料

- [1] BAYER M. SQLAlchemy[G/OL]//BROWN A, WILSON G. The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks. [S.l.]: [aosabook.org](http://aosabook.org), 2012. <http://aosabook.org/en/sqlalchemy.html>.
- [2] Wikipedia contributors. Software requirements specification — Wikipedia, The Free Encyclopedia[Z]. [https://en.wikipedia.org/w/index.php?title=Software\\_requirements\\_specification&oldid=925231559](https://en.wikipedia.org/w/index.php?title=Software_requirements_specification&oldid=925231559). [Online; accessed 19-November-2019]. 2019.
- [3] Jean-Philippe Eisenbarth. A latex template for a Software Requirements Specification that respects the IEEE standards.[Z]. <https://github.com/jpeisenbarth/SRS-TeX>. [Online; accessed 6-April-2016]. 2016.

# 3 总体描述

## 3.1 产品总览

本项目首先开发本地版本。如果有需求,则继续开发在线版本(即套用离线版本,但使用数据库存储数据,同时添加类似 [GitHub](#) 的用户管理);如果有需求,则继续开发本地版本的桌面版本,避免用户命令

行编译 (即将本地版本 API 综合起来, 添加 GUI 支持).

本项目的简介请参考第 2 节. 本地版本主要分为三个部分: (1) 模板文件, 主要为本地版本的默认模板, 如果想自定义, 可以手动修改提供模板 (目前为硬编码在 Python 文件中, 实现初步功能后代码重构会解决此问题); (2) 题库数据文件, 主要以 Json 数据结构存储题目信息, 针对不同的题型设计了不同的键值来存储, 同时继承了相同的键值 (例如题目类型, 描述, 分数, 难度, 解析), 数据模型划分为不定项选择, 不定数填空, 判断正误; (3) 代码逻辑部分不建议非开发人员修改, 而开发人员不需要阅读此条.

在线版本除了在线自定义模板文件及题库数据文件外, 增加存储题库, 合作编辑及版本控制的功能, 此处可以参考 [GitHub](#) 的形式. 主要分为六个部分: (1) 注册用户; (2) 创立公有, 私有题库; (3) 添加合作者至题库; (4) 上传, 下载题库; (5) 在线编辑题库 (数据及样式); (6) 在线生成试卷.

### 3.2 产品功能

版本	功能
本地版本	<ul style="list-style-type: none"><li>使用 Json 数据格式存储题目以构成题库, 纯文本的格式方便用户查看与修改. 但是格式需与在线版本的数据库结构一致, 方便数据库迁移;</li><li>使用 TeX 系统构建试卷, 提供默认样式, 同时支持用户自定义样式;</li><li>使用 Python 连接数据与格式, 抽象题库的筛选算法;</li><li>最初提供命令行编译版本, 后期可以提供不同平台的 GUI 程序 (使用程序均可跨平台).</li></ul>
在线版本	<ul style="list-style-type: none"><li>注册新用户, 上传与下载 Json 格式的题库, 并且在线对题库进行编辑;</li><li>根据标签对题库进行筛选, 构造试卷, 同时控制分数及难度;</li><li>用户管理类似 <a href="#">GitHub</a>. 公有题库可以通过用户页进行查看, 私有题库则需要通过分享链接进行修改 (或添加为合作人员), 同时进行版本控制;</li><li>在线版本的数据库存储需进行加密, 防止后台出现问题.</li></ul>

### 3.3 用户类别及其特征

用户类别	特征
开发人员	可以熟练运用命令行或终端, 可以理解产品的代码及架构, 看完文档之后便可以运用本项目.
编程入门者	可以初步使用命令行, 看完文档之后便可以尝试着在脱离 GUI 的前提下使用本项目.
无编程经验者	没有接触编程, 在看过文档之后不能运用本项目, 此时应提供批处理脚本或 GUI, 可以适当提供线下辅助的使用学习.

### 3.4 操作环境

本项目的本地版本主要采用可跨平台的 TeX, Python 编写, 所以可以运行在主流操作系统中, 相应地需要配置编译环境, 具体可参考 [Read the Docs](#) 及 [Python](#) 官网进行配置. 而在线版本则只需要有可以联网并且安装好了浏览器即可.

### 3.5 产品设计及其实现中的约束

产品设计见第 2 节及第 3 节. 实现中的约束包括: (1) 本地版本用户可能无法配置好  $\text{\TeX}$ , Python 环境; (2) 本地版本用户可能对数据库的修改存在误差, 导致结果出现错误, 但是本项目中前期不采用捕获异常操作, 可能导致错误不易理解; (3) 在线版本用户可能设置权限出现问题, 导致私有题库外泄.

### 3.6 用户文档

目前本项目尚未作出初步版本, 请先参考本文档, 等到项目完成后, 将替代此部分.

### 3.7 假设与依赖

假设用户可以遵循数据库的设计, 可以参考文档配置出高质量的模板文件. 本地版本的成功运行依赖于以上对用户的假设, 而在线版则需考虑服务器安全问题, 假设用户会制造出不遵循规范的数据, 所以在写入数据库之前需要对数据进行严格的校验.

## 4 外部接口的要求

### 4.1 用户接口

本地非桌面版本需要考虑用户接口, Python 代码需使用 `click` 库添加命令行使用的参数. 目前本项目尚未完善, 所以暂时空缺此小节.

### 4.2 硬件接口

本项目不涉及对硬件进行操作, 所以无硬件接口.

### 4.3 软件接口

所有的代码需要遵循 [PEP 8](#) 协议, 但是为确保向前兼容, 故不推荐使用类型推导式 (即 `num: int = 1` 与 `def f() -> int: return int()`), 此外所有的接口需参照以下函数的定义 (Python):

```
1 def api_name_must_be_explicit(arg_one, arg_two='default_value', *args, **kwargs):
2     '''API name must be explicit, this is a summary of this function.'''
3
4     :Argument:
5         - arg_one: <type>, <explanation>
6         - arg_two: str, <explanation>, default is 'default_value'
7         - args: list, <explanation>
8         - kwargs: dict, <explanation>
9
10    :Return:
11        - <case_one>, <explanation>
```

```
12         - <case_two>, <explanation>
13         - <...>
14
15     :Example:
16         >>> api_name_must_be_explicit(None)
17
18     :Reference:
19         - [<ref_name>](<ref_link>)
20     '''
21     <function_definition>
```

#### 4.4 通信接口

本项目的本地版本不涉及通信, 只有在线版本提供通信接口. 目前在线版本尚未完善, 所以暂时空缺此小节.

## 5 系统其他非功能需求

### 5.1 系统性能需求

鉴于家用笔记本可以处理本项目本地版本的计算量 (最大的运算量分配在  $\text{\TeX}$  编译上), 所以在项目构思阶段对系统的性能没有要求. 但是在线版本需要考虑系统的性能. 假设在线版本供一个学校使用, 则需要使用高效的服务器架构 (例如 `Nginx`), 如果经费有限, 则考虑使用排队策略处理请求.

### 5.2 系统安全需求

系统安全方面的问题交由专业人员处理, 目前能做的是服务器关闭不用端口, 设置长密码及申请 HTTPS 安全证书.

### 5.3 系统保密需求

为保证用户数据的保密性, 则在线版本的数据库存储需进行加密. 常规来讲, 用户的所有数据 (尤其密码) 更需要进行加密处理.

### 5.4 系统质量指标

衡量系统的质量主要有以下三个指标: **1** 本地及在线版本的编译时间; **2** 服务器处理请求每秒的数量; **3** 在线版本的异常捕获能力.

## **5.5 授权规则**

### **5.5.1 教育授权**

教育授权 (目标为学校) 有待商榷.

### **5.5.2 商业授权**

商业授权 (目标为教育机构) 有待商榷.