Quick Log / 速记日志

This document logs ideas occrrued to me when working with the program.

2025-10-04~13 | A Week Trail

Question Discovered - 04~13 Question Dolved - 12~13

1. Requirements & Issues

1. `Add Entry`: 删除Add Entry(与Batch insert雷同)
2. `Batch Insert`: 出处能否改为可以选择（搜索）之前录入过的出处（按照字母数字顺序）以及能否统一修改出处名称，这能在一个.py中做到吗？
3. `Home Admin` + `Admin Bulk Edit`: 能否合二为一？
4. `Home Admin`: 默认保存修改选项
5. `Batch Insert`: 默认改为en-zh 不拆分优先
6. `From Page`: 这个文章阅览功能能出现在新的一栏里（from page），单独构成一个.py文件。要求是：（1 一篇文章 = 一个出处；（2 输入一个出处后，可以按照先后输入顺序显示相应的所有这个出处中的句子；（3 但是也需要中途插入句子（这里指一个有id的句子，而不是在原句上改）
7. `From Page`: （1 希望选择栏是可以随下拉一直显示在屏幕中的窗口， Home的insert功能希望有个快捷键点击入库，而不是长拉到下面；（2 图中的功能希望也像choose一样显示在左边，最好可以调整收缩，可以收起来看不见。
8. `From Page`: 8中功能目测很难以使用。希望把功能变成这样：（1 choose的位置改到文章前面（sentences for 附近，如第二张图）（2 在每个句子旁有一个编辑按钮把图一的功能装载进去，并且可以进行句子的各种修改（来源，出处，时间等）；但平时这些修改和编辑功能看不见，除非你点击“编辑”按钮。
9. `Insert` 识别到不匹配的中英文入库，驳回 (错误导入机制的识别,详细见`Day_5_2`)
10. `Insert`仿照新的`Home Admin`收成了小窗口非常便捷
11. `Home Admin` 万一集体导入错误，有一个时间筛选删除功能,可以精确到秒(已追加)
12. `Home Admin`: 预览匹配出来的是否可以一个个就能改（同1合并Add Entry）
13. `Home Admin`: Home这里没法筛选出处去查找了。新增一个出处筛选，以及这个出处栏（1 不仅可以选；（2 而且可以输入。
14. `Home Admine`: 统一出处修改(已添加) 为Bulk Replace
15. .jpg输不进去（微笑.jpg），因为规则按照正则句号断开（使用一些现成的模型）
16. 能不能空一行就算录入一段（我记得之前有这个功能，但这一版没有了）

17. `Insert` 预先查看每条那请自动打开，不要预设折叠上！
18. `Insert`:注入键别在下面做成浮动，出现在方向旁边，所以方向也是浮动显示
19. `Insert` Batch入库记录看不到编号（已调整）
20. 如何对应中英进行集体修改？ `Insert`+`Home Admin`的克隆可以改id顺序（方法是通过克隆一行出现替代原有内容，而达到ip修改效果）
21. 功能合并为菜单栏

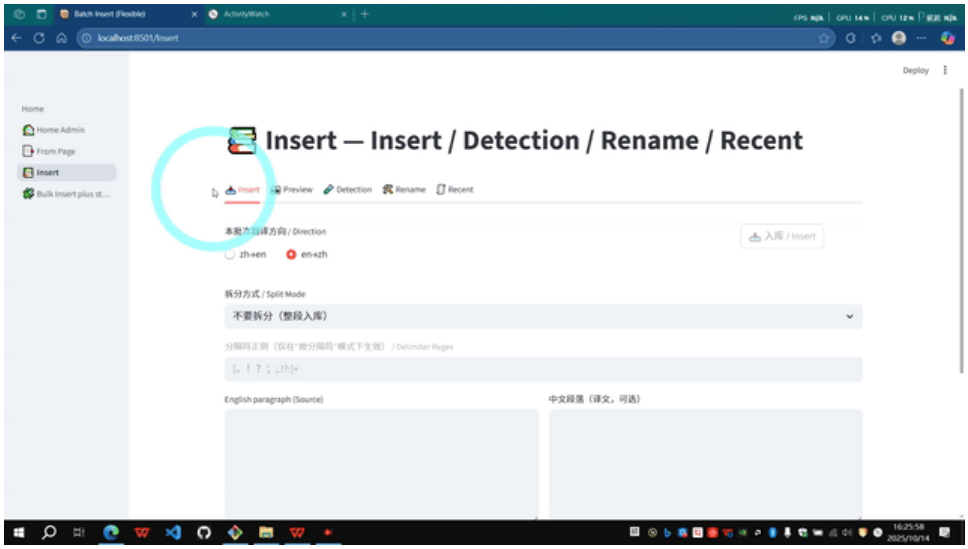
2. Full-stack Preparation

1. 为翻译软件新设一个开发工具箱(比如综合管理字体、网站等内容的调度)
2. `Insert`+`Home Admin`: 入库能不能改为浮动显示（相当于无论如何拖动都会显示出来，就像chat你里面那个代码的复制按钮一样）。这个功能目前在streamlit上不支持，所以希望可以脱离streamlit框架，改为正式全栈模式（上强度）
3. 输入数学公式怎么办？期望有可替代的模型。
4. 预览能不能在输入新内容时就刷新
5. 语言切换
6. 设计好几套字体系统/风格；格式进行统一调整、
7. 软件怎么设置快捷键，以及不与网页和各种端冲突？
8. 增加浮动窗功能 (目前streamlit不支持)
9. 添加为原始文档的索引功能，涉及到云盘和数据迁移；是一个全新的功能！！！

2025-10-14~16 | New Trial

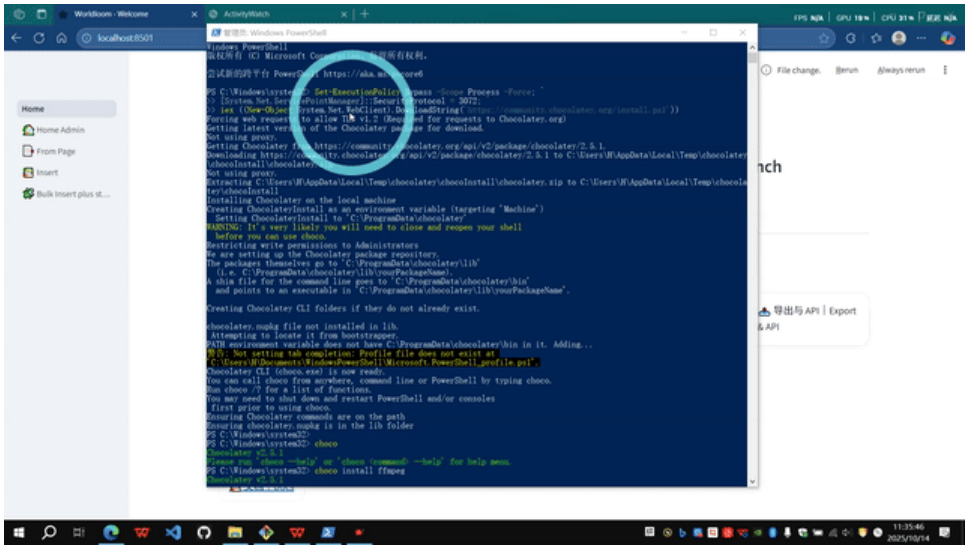
1. Requirements & Issues

1. `Insert`: 这条已显示能否摆在前面，否则录入时看不到，因为和录入键隔开太远；

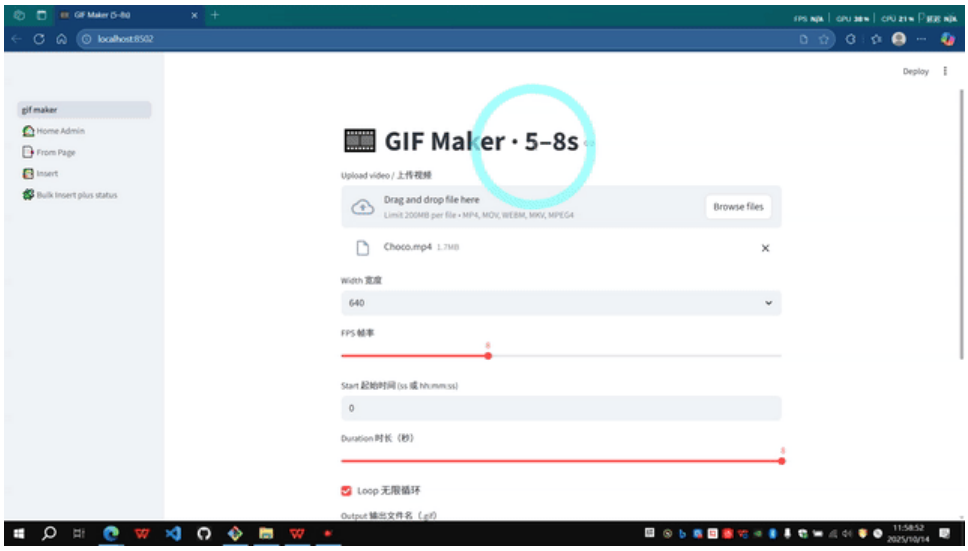


2. `Quick Log`: 为了辅助log管理，新增两个工具：（1 WPS录屏功能；（2 鼠标高亮和取词 PointerFocus;

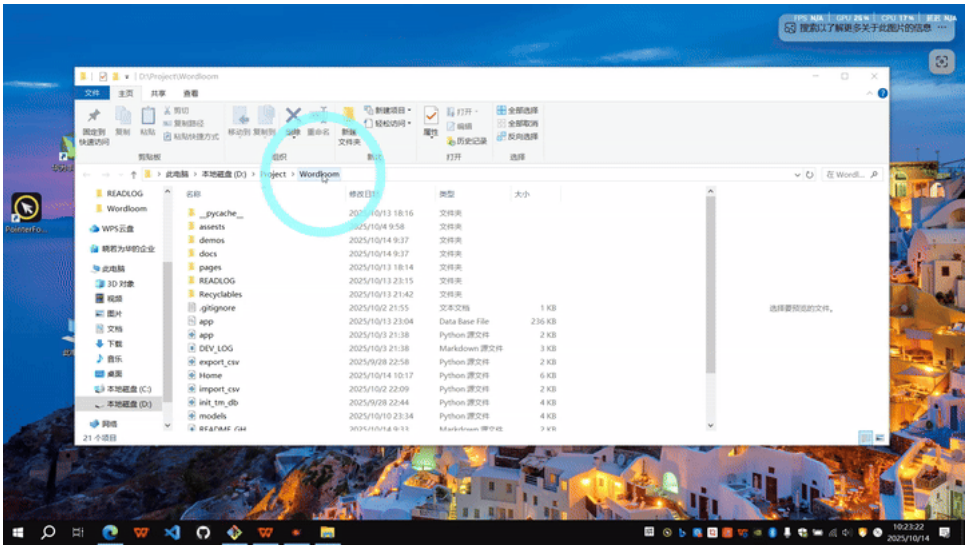
3. 因为今后组件过多，安装choco进行管理，安装以后可以用命令行工具安装文件



4. `Quick Log`: 考虑嵌入视频，但是markdown不支持，所以附带视频链接的同时，增加gif降低git阅览损耗，但也提供html版本。
5. `gif_maker`: 基于上述需求，制作小程序gif_maker来制作高质量的gif图片，以下开始就有演示

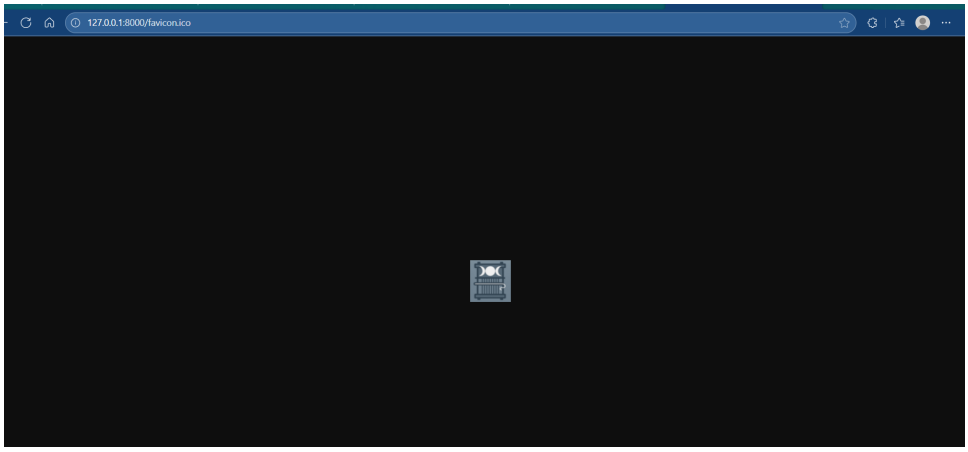


6. 路径名过于冗杂（包含中文，调用容易出错）换成了新的路径拥有新欢迎页面，并且拥有了名字 Wordloom!



7. 新增功能的detection在代码翻译输入上有问题。默认语言检测关掉。
8. `Gif-Maker`: 默认显示输出的文件名为插入的文件名，以及把输出同步到assets中一键管理
9. `DEV_log`和`Quick_log`: 关于日志管理，在日志中同时添加了可看的gif链接也添加了可进一步查看的视频MP4。为了文件夹方便干净，重新在asset中做了分类，而不放在Readlog里面；

10. 搜索A时出现span和渲染冲突问题；
11. 每次一旦文件夹迁移就导致markdown文件都要改动。这怎么一劳永逸的解决??? 这个问题在软件工程里面叫做路径耦合；
12. 现在我欠了很多技术债，我必须一点点了解我所有代码的逻辑，我要怎么去补效率比较高，以及我对这些英语也不够了解，我还需要一边学习这些内容的英语一边学习代码知识，头疼。
13. 工作流需要，我的频繁调出自己的目录树，计划开发小工具来导出-不用powershell自带。以后统一把有用的命令行归类
14. 为wordloom成功设计了一个小logo，然后经过调整弄进网站了！

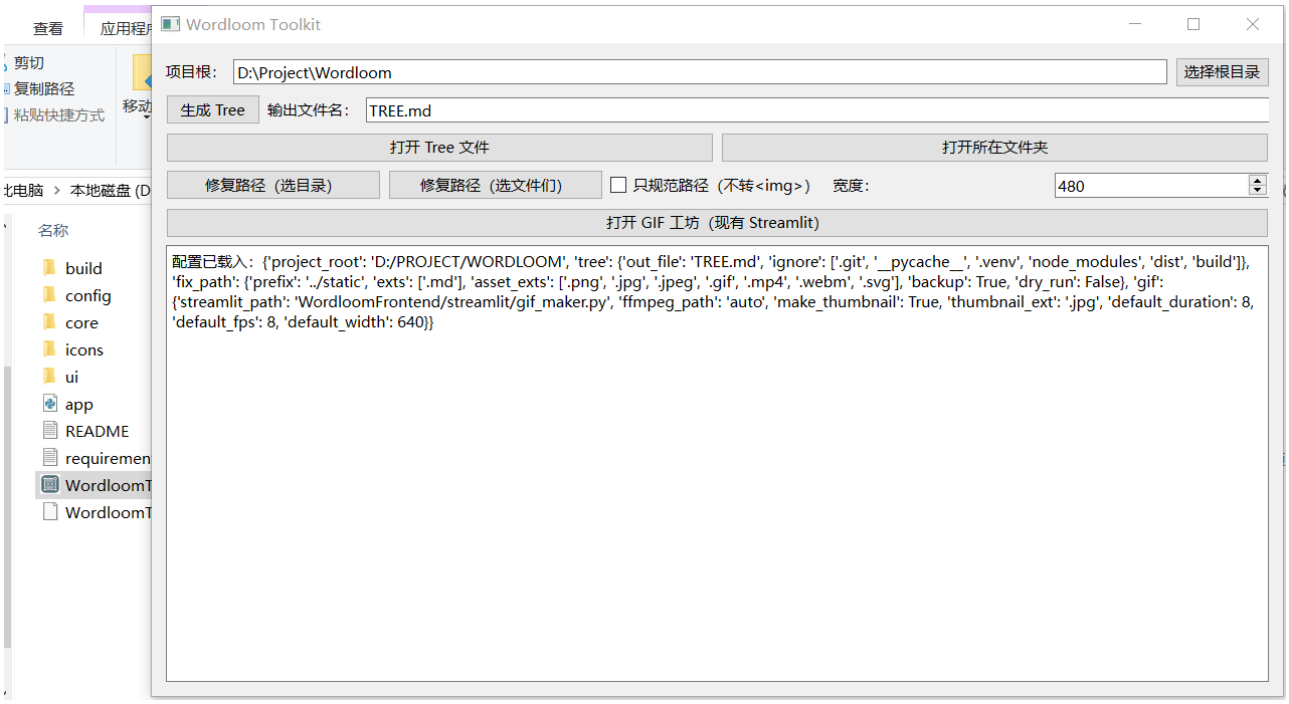



```
INFO:     127.0.0.1:5151 - "GET / HTTP/1.1" 307 Temporary Redirect
INFO:     127.0.0.1:5151 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:5151 - "GET /openapi.json HTTP/1.1" 200 OK
WARNING:  WatchFiles detected changes in 'app\main.py'. Reloading...
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [7116]
INFO:     Started server process [28860]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
WARNING:  WatchFiles detected changes in 'app\main.py'. Reloading...
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [28860]
INFO:     Started server process [28348]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
WARNING:  WatchFiles detected changes in 'app\main.py'. Reloading...
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [28348]
INFO:     Started server process [19220]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:5854 - "GET /favicon.ico HTTP/1.1" 204 No Content
INFO:     127.0.0.1:5854 - "GET /favicon.ico HTTP/1.1" 204 No Content
INFO:     127.0.0.1:6001 - "GET /favicon.ico HTTP/1.1" 204 No Content
INFO:     127.0.0.1:6001 - "GET /favicon.ico HTTP/1.1" 204 No Content
WARNING:  WatchFiles detected changes in 'app\main.py'. Reloading...
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [19220]
INFO:     Started server process [20284]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:7324 - "GET /favicon.ico HTTP/1.1" 200 OK
```

15. 数据库和API迁移。(1 成功与streamlit解耦 （2 解耦不完全，为了方便数据管理，以后都不带下划线，把文件树重新整理
16. 验证后端完全迁移成功：三部曲

- `GET /healthz` 返回 `{"ok": true}`
- `POST /auth/login` 用 `admin/admin123`，拿到 `access_token`（`devtoken123`）
- 带 `Authorization: Bearer devtoken123` 访问 `/entries/recent`、`/sources` 等接口

17. 增加swagger的小锁
18. 需要一个一键生成tree的小程序
19. 增加路径修正器，去掉历史的绝对盘符前缀，把路径层级调整为相对路径
20. 多标签文件管理问题
21. 把数据导入新的api的db中
22. 集成wordloom的开发工具箱（比如tree，比如路径修复，比如gif转化）采用GUI框架，并成功弄出一个小程序，会在后期设置中英双语。并增加一键打包脚本bat。



23. 设置jwt
24. 一切顺畅，准备将streamlit换为更强的前端
25. GIFmaker:默认生成位置问题，第二是点开目录问题

2025-10-17 | New Trial + 1

1. 以后把每次chat的聊天内容提前生成文本日志和Quick Log做对比
2. 设计结构化日志 + 校验 进行断言 - 合并进WordloomToolkit里（把断言脚本及时落地）
3. 关于回滚文件与安全返回点：
4. 前端设计
5. 学习相关术语与英语，以及及时把Git落实到框架上
6. Git学习完毕进入实操阶段，记得每次弄完commit以及发布新的CHANGELOG
7. 版本号追加 Wordloom/ │ ├─ VERSION ← 全局版本号 ├─ WordloomBackend/api/VERSION ← 后端版本 ├─ WordloomFrontend/streamlit/VERSION ← 前端版本 └─ assets/VERSION ← 资源版本
8. 启动Git LFS，管理大量的gif, image, 和视频文件
9. 准备pre-commit钩子，用于工程级别的遗忘操作：)(确实很正常)
10. CI/CD可以等以后再开发
11. 把版本发布的py合并到toolkit里面，另外以后做pre-commit后放到pre-push里面

2025-10-17~ | Frontend development

streamlit 模块与抽象化

1. 第一批文件发送 推荐分批顺序（每批 1–3 个文件即可）： Home.py repo.py + models.py（如果有 text_utils.py 一起） pages/0_🏠_Home_Admin.py pages/3_🛠_Bulk_Insert_plus_status.py pages/2_🗃_Insert.py pages/1_📄_From_Page.py gif_maker.py（确认缩略图/占位的调用点） 你就按这个顺序一批一批贴过来或上传文件，我读完一批就回你一批可直接替换的改造补丁。

2. 第二批文件发送 第 2 批：数据访问抽象（今天就铺好路，将来切 API 不会痛） 目标：让前端只认识"数据服务层"，不要认识 SQLite 细节。 抽象层落点 以 WordloomFrontend/streamlit/repo.py 为门面，models.py 定义数据结构，text_utils.py 做辅助。先适配本地 app.db，等你后端 API 稳定后，只改这一层

的实现即可。 最小契约 搜索、分页、按出处聚合、插入/更新/删除、批量导入（行迭代 + 事务），全部在这层封装。

3. 第三批文件发送

第 3 批：路径与样式的一次性“止血” 目标：防止“搬家就全崩”的相对路径问题 + 统一样式入口。 路径 沿用你现有的 assets/static/... 组织；前端组件里统一从相对根引用。你仓库里有 fix_md_paths.py（用于文档），前端也会放一个 url_for_asset() 小工具，避免硬编码。 样式 加一个 styles/base.css（或保留现状），让字体、颜色、行距等在一处管控；不强制换你已有字体/配色，只做“可配置”。

4. 第四批文件发送 目标：给你“跑一遍就能心安”的最小测试与烟囱脚本。 健康检查脚本：启动后端（如需）、检查前端关键页能否加载、抽样读写是否成功。 演示数据夹具：一小份“英文↔中文”样例，便于回归。 目录树导出：你工具箱里已有 tools/WordloomToolkit 与 tree_runner.py，保持使用。

弃船上岸方案

1. 做到第一批末尾，结果Streamlit已经无法兼容大多数页面功能，出现多次重复读取和白屏。所以现在目标是直接升级换代。

2. 项目骨架 src/ app/ layout.tsx # 全局布局 + 字体 page.tsx # 主页，可做导航/最近源 admin/ page.tsx # Home Admin from/ [source]/ page.tsx # From Page（按 source） insert/ page.tsx bulk/ page.tsx api/ # 仅限 next/server actions（可选） components/ ui/ # shadcn 组件 common/ Toolbar.tsx EntryCard.tsx EditDialog.tsx ConfirmDialog.tsx admin/ SearchForm.tsx ResultsList.tsx from/ SentenceList.tsx lib/ api.ts # axios 实例，拦截器 queryClient.ts # TanStack Query 客户端 repo.ts # 对后端的“服务层”：统一封装 entries/sources schema.ts # zod 校验（与后端对齐） fonts.ts # 本地字体注册 config.ts # WL_API_BASE 从环境读取 types/ openapi.d.ts # openapi-typescript 生成 styles/ globals.css tokens.css # 颜色/字号/间距变量

3. 批次

4. 关于中文输入法与VScode的不兼容问题调整