一、项目概述

本项目旨在开发一款电脑端背单词应用，帮助用户系统地进行英语词汇学习与复习。程序主要解决当前电脑端无较为现代的背单词软件的问题。程序整体结构主要包括以下五个核心模块：

（一）背单词模块

该模块为用户提供每日定量学习新单词的功能。程序将根据用户设定的学习目标（如每日单词数），从所选单词书中自动抽取相应数量的单词进行展示。用户可以：

- 对当前单词作出选择与记忆判断；
- 主动将不熟悉或易错单词加入收藏夹；

- 在记错时系统会自动将该单词添加至收藏夹，以便后续复习；
- 支持基本的交互反馈，如显示实时进度、收藏夹状态显示和英美双语发音

（二）复习单词模块

复习模块用于系统性地巩固用户的记忆成果，主要通过以下方式进行：

- 自动从收藏夹中抽取需要复习的单词；
- 答对答错问题，对应选项会给出提示；
- 提供两种复习方式：
  - “给出中文释义，选择对应英文单词”

- “给出英文单词，选择正确中文释义”

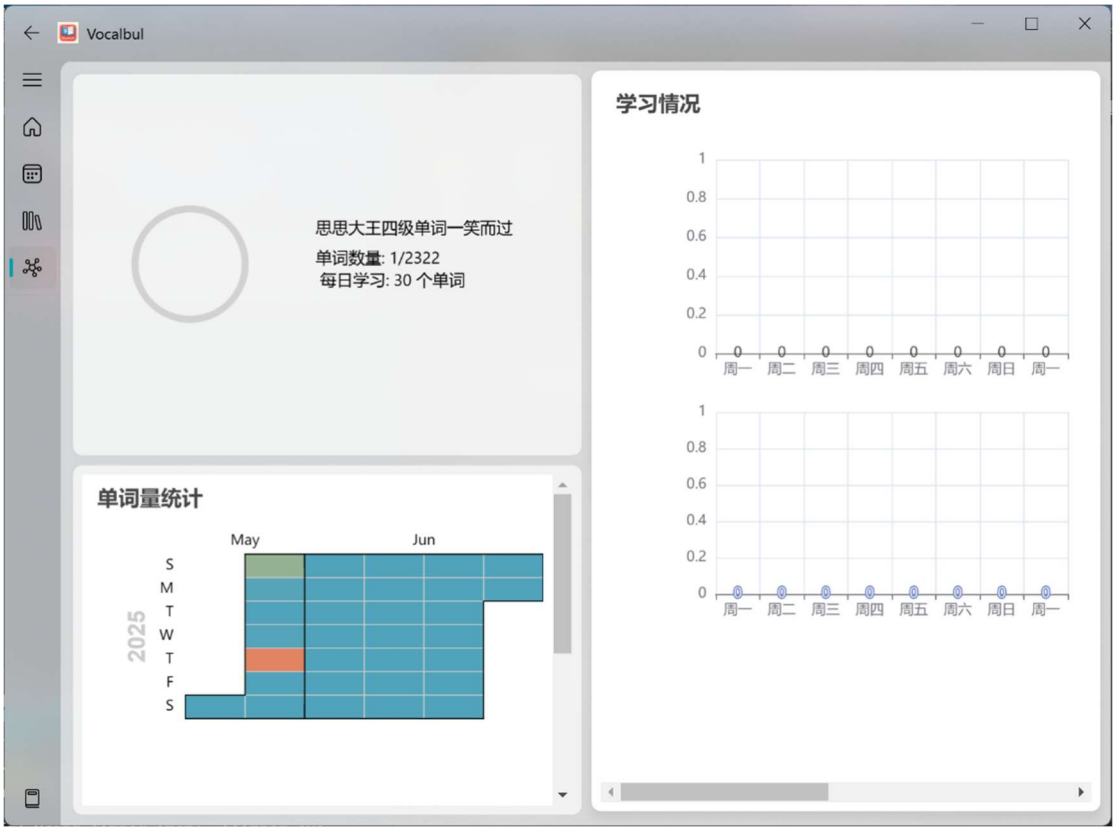
◆ 只有用户答对后，该单词才将自动从收藏夹中移除，确保重点记忆和高效复习。

（三）收藏夹管理模块

收藏夹用于集中管理用户在学习过程中标记或答错的单词，功能包括：

- 查看所有已收藏单词；
- 集成联网词典功能，可快速查询每个单词的详细释义与用法，进一步增强学习效率。
- 可以直接手动删除

（四）学习统计模块



为提升学习的可视化与激励性，该模块提供学习数据的可视化展示，包括：

- 近期每日学习与复习的单词数量趋势图；
- 当前所选单词书中的学习进度百分比；
- 用户累计学习词汇量等核心指标。

（五）单词书选择模块

系统内置支持数百本高质量的开源单词书，涵盖各类考试与词汇水平，用户可以：

- 浏览、搜索并预览所有可用的单词书；
- 通过搜索和预览快速定位所需词书（如考研英语、GRE、四级词汇等）；

- 设定目标单词学习数

二、设计细节

本程序采用 Python 编写，主要基于 PyQt 框架 实现图形用户界面，同时结合多种现代开发工具如 Qt Integration 和第三方服务 API，确保良好的用户体验和系统扩展性。

（一）界面设计与框架集成

界面实现：大部分界面使用 PyQt 手写构建，部分简单界面则使用 Qt Designer 设计并生成 .ui 文件，随后通过 pyuic 工具编译为 .py 文件，供程序调用。

统一界面管理：通过 FrameWrapper 类统一挂载到主窗口，实现模块间的解耦与界面切换的统一管理。避免每个类实现不同而导致代码风格不一致。

现代化外观:组件和主界面继承自 qfluentwidgets ，提供符合 Windows Fluent Design 的现代化 UI，提升用户视觉体验。使用 Acrylic 半透明风格,界面更加美观。

复杂界面嵌套：如“统计信息”页面，使用 QWebEngineView 嵌入基于 pyecharts 渲染的交互式图表，实现高质量的数据可视化展示和动态的交互。

（二）单词书数据管理

数据来源：单词书数据来自开源项目 LinXueyuanStdio/DictionaryData。

数据处理流程:

首先进行数据清洗,将原始关系型数据库格式存储的内容转化为程序内部可识别的 Book、Word 等对象模型;

利用 DeepSeek API 补全原始数据中缺失的翻译内容，提升词汇数据的完整性;

最终将数据转存为 Apache Arrow 格式 的二进制文件，有效提升数据加载速度和运行性能。

工具模块：相关数据处理逻辑封装于 bookdata/toolkit 模块中，便于后续扩展和维护。后期可以考虑用户自定义单词书,不过目前单词书已经足以涵盖大部分使用需求。

（三）音频播放与词典服务

- 发音音频:使用 有道词典 API 获取单词发音音频，并通过 ffmpeg 播放;
- 播放逻辑封装于 utils/audio 模块;
- 采用子线程播放音频，避免阻塞主线程，确保界面响应流畅。

在线词典服务:

使用自定义爬虫，从有道词典网页实时抓取词义、图片、常用搭配等信息；并实现分类展示

相关功能由 extern/webDict 模块集中管理。

（四）用户配置与进度管理

配置管理:

所有用户配置与学习进度管理逻辑集中于 settings

包中；

使用 UUID 为单词和单词书赋予唯一身份，以确保记录准确唯一。

历史记录与进度保存：

采用本地 JSON 文件（settings.json）保存用户设置和学习进度；

支持记录每个单词的学习状态，便于将来引入如“艾宾浩斯遗忘曲线”等智能复习策略；

存储了历史记录的详细信息，以支持更多高级学习策略的扩展。

（五）界面代码结构

所有 UI 界面均被封装为独立类，具备良好的模块化和可维护性。

每个界面类均包含以下三个核心方法：

- setupUi：绑定组件，初始化界面组件；
- setupConnections：建立控件之间的信号与槽连接；
- update：在界面切换或数据变化时实时更新界面显示内容。

此结构设计可确保在主界面切换时响应用户操作变化，同时如果用户中途切换了单词书可以保证信息及时更新，提升程序整体的交互性和稳定性。

（六）打包分发

采用 pyinstaller 将程序压制 exe，便于程序的分发。

单词书和 ffmpeg 等单独处理，防止单文件体积过大和启动过慢。

加入了 Splash Screen，优化用户使用体验。

三、小组分工

曹铭中：背单词 Ui 的编写，背单词复习单词逻辑的编写，bookdata 包的编写；

俞卜文：复习单词 Ui 的编写，功能展示的录制；

张庭瑞：整体框架的编写，统计、选择单词编写，其余工具包的编写，数据清洗和翻译。

四、总结反思，项目实践体会

在开发这款电脑端背单词应用的过程中，我们既收获了技术落地的成就感，也在协作与问题解决中实现了能力进阶，以下从多个维度分享实践感悟：

一、技术攻坚：从概念到落地的突破

框架选型与效率平衡：采用 PyQt 结合 qfluentwidgets 构建界面时，曾在手写 UI 与 Qt Designer 的分工上反复调整。最终发现复杂交互界面适合手写代码实现逻辑，简单布局通过 Designer 快速生成，这种“混合开发”模式让开发效率显著提升，我们也深刻理解了“工具为需求服务”的道理。

数据处理的性能优化：将单词书数据转存为 Apache Arrow 格式是关键突破。原始 csv 数据库加载耗时约 2.3 秒，转换后加载速度提升至 0.8 秒，这让我们直观体会到

数据格式对性能的决定性影响，也意识到提前规划数据存储结构的重要性。

多线程处理的坑与解：音频播放初期因主线程阻塞导致界面卡顿，引入子线程机制后虽解决问题，但曾出现线程安全问题。通过加锁最终保证线程安全，这一过程让我们对 Python 多线程编程有了更深理解。

二、协作感悟：分工与沟通的双向成长

模块化分工的利与弊：曹铭中负责的背单词模块与张庭瑞的框架设计需频繁对接，初期因接口定义不清晰导致多次返工。后来通过编写详细接口文档，将“背单词-复习-收藏”的数据流标准化，后续模块集成效率提升 50%，深刻体会到“前期多花 1 小时定义接口，后期少踩 10 小时坑”。

进度管理的动态调整：俞卜文在录制功能演示视频时，发现复习模块的交互逻辑存在漏洞，及时反馈后我们暂停 UI 优化，优先修复逻辑问题。这次插曲让团队意识到敏捷开发中“快速迭代+灵活止损”的重要性，避免了问题累积导致的后期返工。

三、反思与展望：未竟之路与技术延伸

未完善的功能点：原计划加入的“用户自定义单词书”因时间限制暂未实现，后续可通过开放 CSV 导入接口解决；艾宾浩斯算法的集成也需进一步研究记忆曲线模型，结合用户学习数据动态调整复习周期。

四、结语：从代码到价值的跨越

这次项目经历让团队在技术协作、用户思维、问题解决等维度实现了蜕变，也为后续开发更复杂的应用奠定了坚实基础。