AI 辅助的日程管理器
——程设 Qt 大作业报告

小组成员：张子苏（2100012732）杜宇凡（2100012734）袁子烨（2100016639）

一、 程序功能介绍

在当今快节奏的生活中，合理地安排日程、高效地完成任务，对于我们每个人来说都愈发重要。我们组构想的 AI 辅助的日程管理器正是在这样的背景下应运而生。这款基于 QT 的软件利用课程所学面向对象中继承、多态等手段建构了多种任务日程，实现了丰富的任务日程管理，同时使用 GPT 交互，以更科学智能的方式提升我们的日程管理效能，为我们的生活、学习和工作带来便利。

核心功能

1. 全面的事件任务管理

   本软件具有全面的事件管理功能，不仅支持基本的增、删、查、改、标签增删、归类的操作，我们更进一步考虑到事件的多样性和复杂性。我们根据事件的特性，抽象派生出各种具体事件类别，并为每类事件赋予独特的属性和方法。例如，周期性事件允许用户一次性设置，并提供周期性提醒。对于项目类事件，我们支持父子任务关系的设置，从而构建出清晰的任务管理网络图。对于 DDL 事件，我们明确了任务的完成时间，从而给用户更合理的提醒服务。对于标签类的管理，我们的任务视图更有层级特征。

2. 跨平台的同步

   通过利用 RESTful API，我们实现了前后端的分离，使得前端可以通过简单的 HTTP 动词进行所有的操作。这一设计使得前后端的分工更加明确，同时提升了系统的安全性。目前，我们的前端通过 Qt 实现，未来还有可能扩展到网页或移动应用上。另外，我们采用了远程数据库连接来存储信息，这使得信息在多设备间的共享和同步成为可能，也为后续可扩充的多用户协同任务做好了铺垫。

3. 简洁易用的用户交互界面

   我们的设计原则是简洁明了，旨在为用户提供既美观又易于操作的界面。通过直观的用户交互设计，我们致力于使用户的每一次操作都变得轻松愉快。

4. AI 辅助的任务管理

   我们采用了创新的人机交互模式，用户可以通过自然语言输入来让日程管理器执行相应的操作。例如，用户可以请求 GPT 帮助规划一周的健身计划，只需输入自然语言要求、一键点击，即可生成整周的计划，同时也可以用自然语言对已有的任务进行删改等操作。这种 AI 辅助的方式，无疑大大提升了任务规划和管理的效率。

二、 项目模块与类设计细节

（一）前端

该部分的源码位于 thezzisu/elastic-todo. 可以在Cloudflare Pages上看到由 CI/CD 管线实时部署的最新版本。

考虑到多端兼容性和代码复用目的，将前端作为 Web Application 单独实现，具体的原因可以见下文的总结部分。采取基于模块的解耦合和代码组织方式，对于界面骨架采用基于 Component 的拆分模式，不同的视图组件实现在对应的 SFC 源码中（`src/components`）。设计时充分考虑到了多后端（`Provider`）的支持，采用了基于类型抹除的多态方式，不同`Provider`的实现见`src/providers`。利用了提供持久化数据的 Web API 包括

`IndexedDb`与`WebStorage`等，提供了`Provider`配置和鉴权信息等应用状态的持久化。

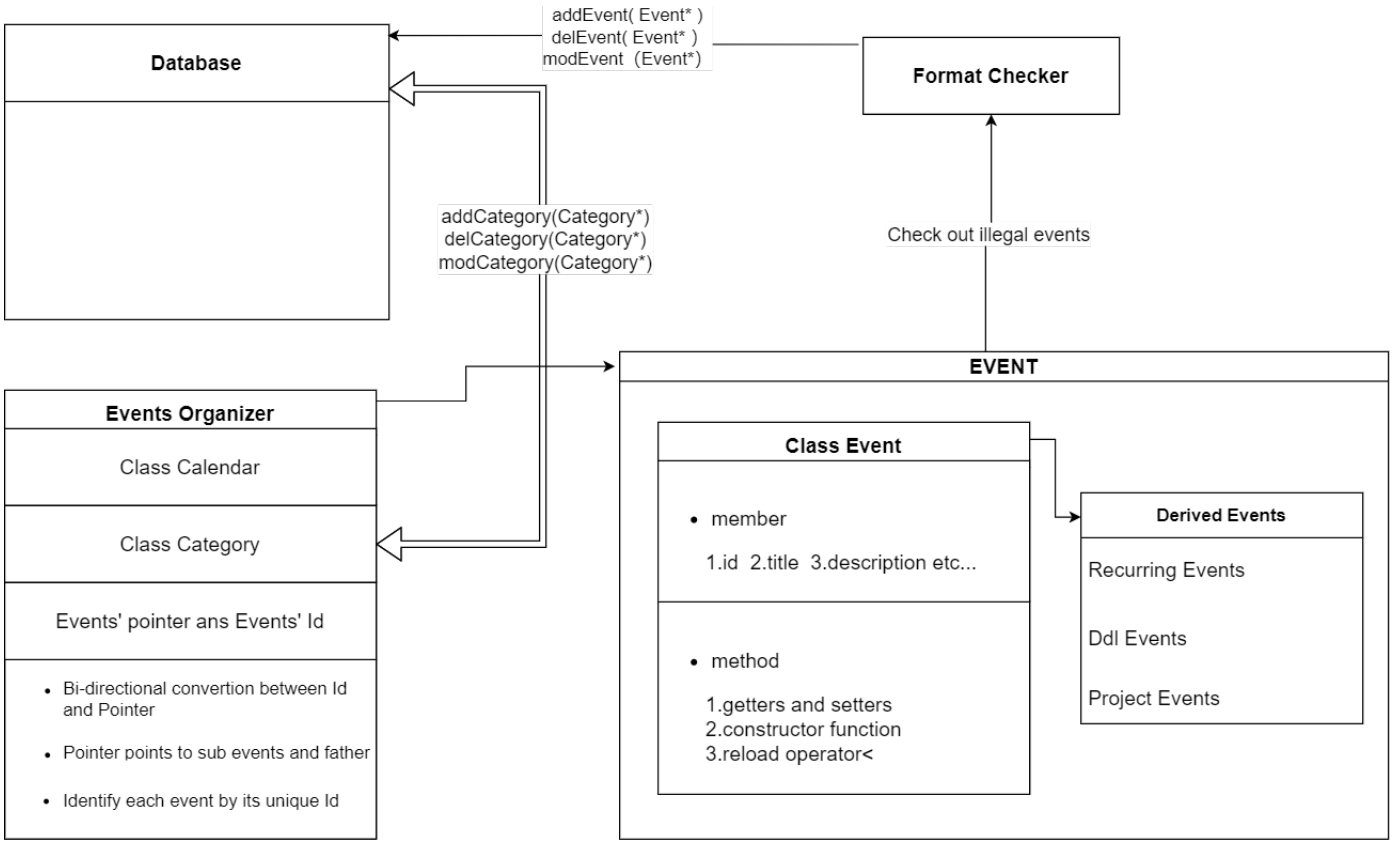
前端代码目录结构简介如下：

- `public`：静态资源文件
- `src`：源码文件
  - `components`：共用组件
  - `database`：基于 IndexedDb 的数据库 Wrapper
  - `pages`：页面组件，基于目录结构自动生成路由表
  - `plugins`：功能性插件
  - `providers`：ToDo 数据源提供方之实现
  - `router`：前端路由实现
  - `stores`：响应式应用程序共享状态
  - `utils`：辅助功能模块
  - `App.vue`：前端应用程序根组件
  - `main.ts`：前端应用程序入口
- `package.json`：Node.JS 项目描述文件

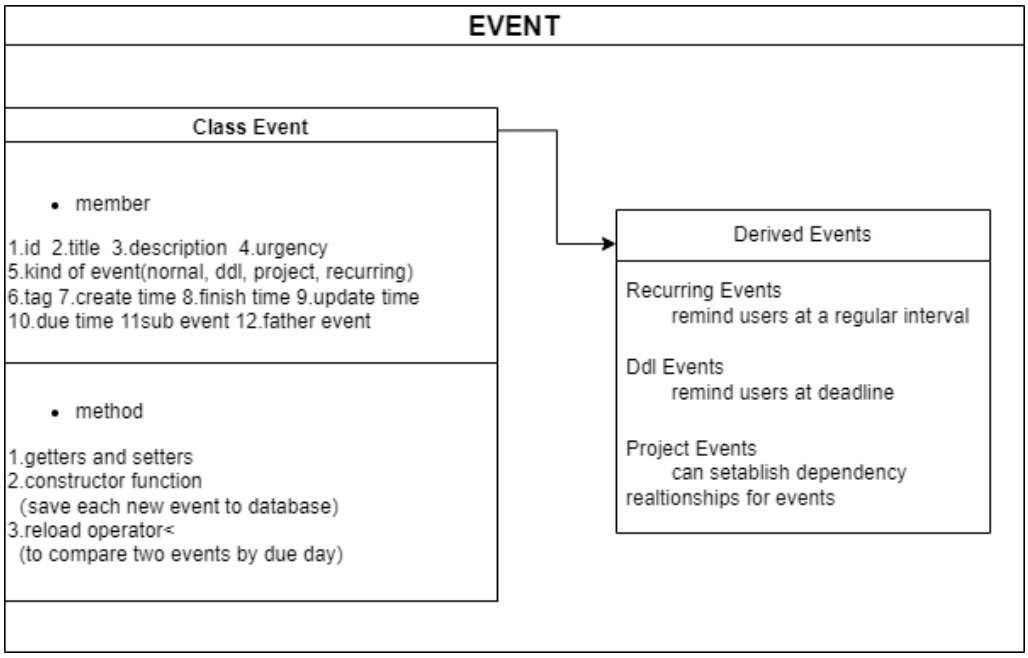
（二）具体面向对象类与方法

日程管理器的类主要分成两部分：一部分是具体的事件类（Events），另一部分是管理事件的Category、Calendar等管理类。

管理器的主要类结构如下：



1.关于事件类



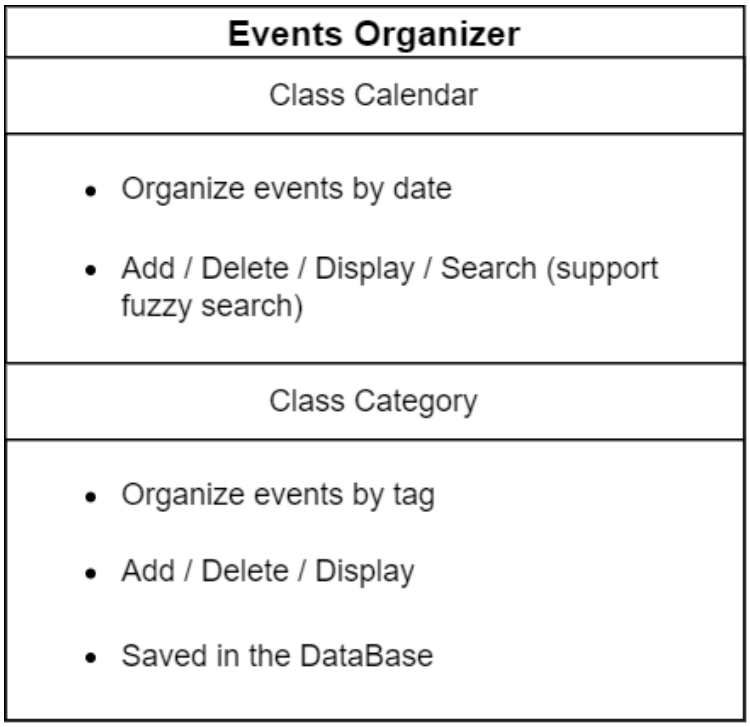
事件类结构图

对于事件类，我们首先实现了抽象基类Event，之后根据不同的用户需求和事件特点，派生出一系列具有不同特点的类。

例如，对于需要反复提醒定期循环的类，我们需要额外实现循环提醒与设置、更改循环周期等功能；对于具有前后依赖关系的具体项目，我们额外增加了用于记录父事件与子事件的两个队列，并分别实现了这两个队列的增删查改等功能。

除此之外，对于所有需要对事件做出改动（新增、改动、删除）的方法，我们需要同时让这些改动与管理类（Category、Calendar）以及数据库分别实现同步。

2.关于管理类



管理类结构图

管理事件的方式方法不是唯一的，我们可以根据事件分类来使用Category类管理一组事件，也可以用Calendar类来依据事件顺序来管理。

这些管理类除了将事件分类外，还要负责为事件增删查改提供接口。

同时，管理类本身也要记录在数据库中，所以同事件类一样，在进行数据改动时需要额外注意与数据库同步。

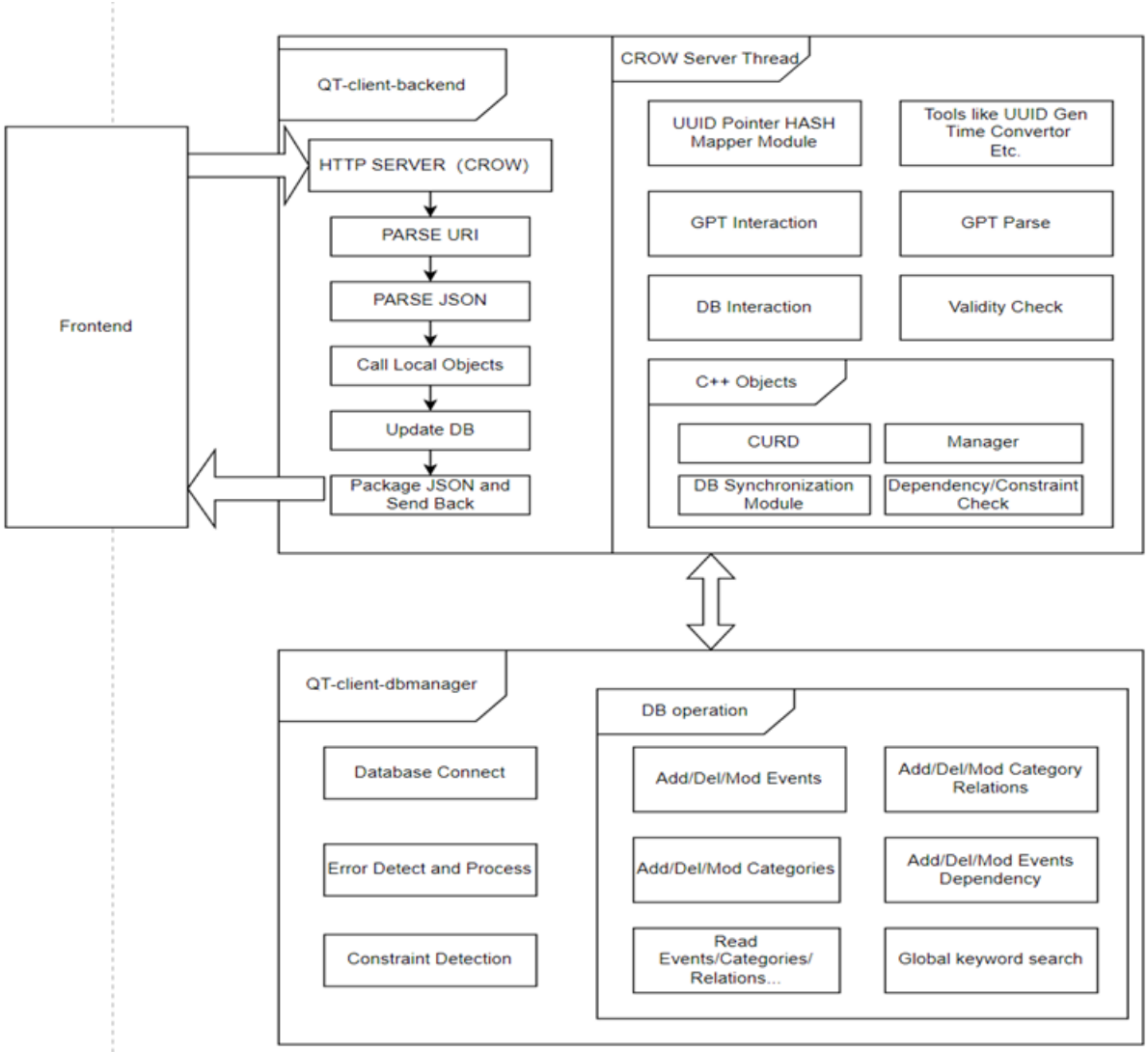
3.其他细节

为了更好地服务于事件管理，管理类实现了诸如“模糊查找”这样的方法。

另外，为了更好地服务于前端与数据库，我适当地重载了一些面向前端和数据库的接口，并在一些关键方法上设计了一些出错时的输出，便于整体项目的调试。

（三）QT 客户端模块结构综述

我们的后端架构完全基于 QtApplication 进行构建，该部分位于 thezzisu/elastic-todo-client。用Qt6.5.1构建，环境需要配置MySql8.0.33、MySql在Qt中的驱动文件和C++的Crow库进行编译。而MySql的地址、服务器端口地址等单纯为了本次大作业简化，放在一个头文件里面方便修改（实际软件不会这样，可能会带来很多问题）。主要包含如下几个模块：与前端交互的服务器模块、由各种功能模块组成的 C++类与对象、以及数据库交互模块。相关模块架构图如下：



Qt客户端示意图

服务器模块

我们将服务器的处理操作等抽象为一个 CrowServerThread 类，便于多线程的同时运作。

为了适应跨平台的需求，我们使用 Restful API 进行封装，并使用 Crow 库构建了一个轻量级的 HTTP 服务器以方便前端交互。主要流程包括 HTTP 服务器对请求的解析（如确认请求方法、获取请求中的 UUID 等信息）、对请求中 JSON 负载的解析（解析方式会根据请求方法的不同而不同）、利用哈希表映射获取本地缓存对象指针抑或是直接调用特定方法执行操作。在本地数据修改后，后端将实时同步数据至数据库以便保存，有助于跨平台的数据共享和未来的扩展。流程结束后，后端会向前端发送反馈，告知任务的完成状态（可能带上 JSON 负载）。

数据库模块

数据库模块被抽象为 DatabaseManager 类，主要负责数据存储与同步，提供各种方法，同时类的抽象有利于数据库操作多线程并发的实现。考虑到可能的多设备使用，我们采用远程连接数据库进行存储。我们为数据库提供了基本的增、删、改、查接口，并利用 MySql 的搜索便利性实现了全局和模糊搜索功能的封装。数据库包括四张表，分别存储事件、标签、事件间关系以及事件与标签的关系。我们选用 UUID（通用唯一识别码）作为主键，整体满足第四范式（4NF）要求，便于进行查找、修改和删除操作。下列是数据库表创建情况：

Field	Type	Null	Key	Default	Extra
id	char(36)	NO	PRI	None	
name	varchar(100)	NO	MUL	None	
kindofevent	int	NO		None	
description	text	YES		None	
createdAt	datetime	YES	MUL	None	
dueAt	datetime	YES	MUL	None	
finishedAt	datetime	YES	MUL	None	
updatedAt	datetime	YES	MUL	None	
inter_time	int	YES		None	
urgency	int	YES		None	
status	int	YES		None	

(a)事件表

Field	Type	Null	Key	Default	Extra
id	char(36)	NO	PRI	None	
name	varchar(100)	NO	MUL	None	
description	text	YES		None	

(b)标签表

Field	Type	Null	Key	Default	Extra
event_id	char(36)	NO	PRI	None	
category_id	char(36)	NO	PRI	None	

(c)事件标签关系表

Field	Type	Null	Key	Default	Extra
parent_id	char(36)	NO	PRI	None	
child_id	char(36)	NO	PRI	None	
relation_type	varchar(50)	YES		None	

(d) 事件关系表

数据库建表示意图

GPT 模块

为了提升与用户的交互丰富性，我们使用 GPT 的 API 进行用户需求的解析。我们增加 prompt 预设 GPT 返回时的分隔符和信息内容，然后使用字符串解析进行进一步的处理。过程中主要难点在于修改和删除特定事件时，GPT 无法获取我们数据库中的所有信息，因此我们需要根据 GPT 返回的信息在当前本地任务中进行模糊匹配，并选择匹配度超过阈值的最高值。获取到匹配关系后，为防止误操作，后端将 GPT 的相关指示转换为 JSON 信息，并发送给前端让用户确认或修改。用户确认后，前端会再次向后端发送具体的操作指令。

如用户发送以下请求`change my plan to go to the library from tomorrow morning to evening, and schedule a meeting with my advisor in the morning`，GPT 会返回`ADD$$$Meeting with advisor$$$.....`等信息，经过解析后发送回前端的结果如下（其中已完成事件的匹配）：

```json
{
    "value": [
        {
            "urgency": 2,
            "remove_id": "4c4e42b4-4603-40a0-938e-4a99fac81bcd",
            "name": "Go to the Library",
            "description": "To do research",
            "datetime": "2023-06-05T10:00:00",
            "action": "remove"
        },
        {
            "urgency": 2,
            "name": "Meeting with advisor",
            "description": "Discuss report progress",
            "datetime": "2023-06-05T9:00:00",
            "action": "add"
        },
        {
            "urgency": 2,
            "name": "Go to the Library",
            "description": "To do research",
            "datetime": "2023-06-05T14:00:00",
            "action": "add"
        }
    ]
}
```

GPT响应解析结果

辅助功能模块

该部分主要有时间转换模块和关系映射模块。前者主要负责不同模块时间信息兼容性的问题，后者主要解决了前端后端联系、通讯的问题。

三、小组成员分工情况

张子苏：实现前端。

杜宇凡：RESTFul API 服务器类和方法建立、数据库管理器类和方法、GPT 交互处理类等。

袁子烨：QT客户端模块（Event、Category等类的成员与方法）、与数据库相适配的接口。

四、项目总结与反思

关于一些技术选型与分歧问题（by 张子苏）：

- 为什么没有采用 Qt Widgets/QML 实现前端？ \
  笔者具有逾十年编程经验，使用过包括 Qt Widgets、QML、Java Swing、Win Forms、XAML、Dart/Flutter、H5 在内的各种 UI 方案，并对其优劣具有一定的评估了解。大体而言，UI 的实现方式可以分为两种：声明式和程序式。前者包括 QML、XAML、H5 等，以及 Flutter 也借鉴了声明式 UI（具体表现为其语法）。通过将布局单独表达为 Markup 文件，使得项目具有更好的可维护性，并可以程序化的通过语法分析实现高效的热重载与渲染优化。鉴此，笔者抛弃了 Qt Widgets 的实现。毕竟我们不需要在 Low-End 的嵌入式设备上跑大作业。 \
  至于 QML，其本质上是裁切后的 Web 技术的体现。通过 AOT 的编译期转换与绑定生成，QML 提供了比 Web 方案更好的性能和更低的资源占用。但在客户端性能快速增长的现实情况下，这些资源性能上的优势不值一提。 \
  使用 Web 方案，可以与现有完善的 CI/CD 管线打通，实现开发的高度自动化与代码的再利用。基于此，本次大作业不考虑使用 Qt 原生方法实现 UI，而是采用内嵌 WebView 的方式，嵌入 Web App 作为前端。 \
  当然，考虑到课程大作业的初衷可能为锻炼代码能力和 OO 设计能力，选择这个方案具有一定的风险性。但本着需求导向和代码复用原则，笔者仍然认为使用 Web 方案开发是最优选择。

关于管理器类与对象的设计问题（by 袁子烨）
在完成管理器的后端设计之后，笔者认为管理器的类的设计有两个比较重要的细节，这也是笔者大作业中的收获

一方面，管理器的类，无论是Events类还是Category等管理类，都需要强调出事件的层次结构。事件管理器本来就是为了让用户生活中的各种事件有条不紊，事件类的层级组织将直接影响到管理器的使用体验。

另一方面，这些类和对象都是为前端提供接口，所以项目开始前的组织沟通尤其重要，特别是后端接口的设计将影响到前端代码的编写。整个工程能否并行的关键就是提前协商好几个人之间接口，并保持这个约定不变。

通过本次大作业，笔者对软件的设计和编写有了以上初步的认识。

关于后端整体的想法与反思（by 杜宇凡）

为何不用指针来进行前端Qt Widgets和后端的连接？虽然程设课程所学的指针访问管理能起到类与对象很好的管理，但是对于更广泛需求而言，如果需要我们这个日程管理器应用于移动端、网页等地方，指针显然是不现实的。为了更好地提升统一性与扩充性，我们使用RestFulAPI交互，而这不恰恰就是C++在代码扩充与封装上优势思想的延续吗?

所以笔者认为，程设课程给我们带来的，不该仅仅在于学会了C++、虚函数、多态的基本写法，更应领悟一种思想，例如抽象、封装的美感与简约等等。而我们也是基于这种思想考虑，设计了更通用的API，同时考虑MySql的远程数据管理。另外，如GPT返回后的解析与匹配，我们使用了字符串和动态规划相关操作进行，融入课堂所学。

当然在大作业中也有许多值得改进的问题，如在写代码时更应先想好框架，然后逐步击破，而应该减少写代码时对功能的删改的考虑等等。