北大学习时空管家 - 作业报告

(1) 程序功能介绍

“北大学习时空管家”是一款专为北京大学学生设计的综合性桌面学习管理应用。它旨在通过整合日常学习生活的多种需求，提供一个统一、高效的管理平台，从而帮助学生更好地规划时间、利用资源。

该软件核心功能主要包括以下四个方面：

- 智能课表管理 ： 应用支持从北大“树洞”平台获取的课表 HTML 文件。用户只需将文件拖拽到课表窗口，程序即可自动解析并生成一个可视化的周课表。此外，该模块还集成了空闲教室查询功能，用户点击课表上的任意无课时段，即可查询指定教学楼在该时间的空闲教室列表。
- 自习时间记录 ： 为了帮助用户量化学习投入，软件内置了自习计时器功能。用户可以一键开始或结束自习，程序会自动记录自习时长。在学习统计模块中，用户可以查看以条形图或折线图呈现的每日学习时长，并且支持以秒、分钟、小时为单位进行切换。所有自习记录都可以被一键清空。
- 日程提醒系统 ： 应用提供了基于日历的日程管理功能。用户可以在主界面的日历上为特定日期添加、修改或删除任务。为了突出任务的紧急性，主界面的日历会对有任务的日期进行智能标色：距离当天 1-7 天的任务显示为红色，8-14 天为黄色，15 天及以上为绿色。同时，还有一个独立的“日程提醒”窗口，可以集中展示所有未来的日程安排。
- 实时天气查询 ： 软件主界面集成了一个天气查询模块。用户可以输入城市的英文名称，程序会通过调用 OpenWeatherMap API，实时获取并显示该城市的气温和天气状况。

(2) 项目各模块与类设计细节

项目整体采用 C++的 Qt 框架构建 GUI，使用 SQLite 进行本地数据持久化，并巧妙地通过调用外部 Python 脚本来处理网络数据抓取和 API 请求，实现了混合编程的优势。

2.1 整体架构

项目可以清晰地划分为三个主要层次：

1. 表现层 (UI): 完全由 C++和 Qt 构建, 负责所有用户界面的展示和交互。MainWindow 作为主窗口，是应用的入口和调度中心，其他功能如课表、统计、自习等则由独立的窗口或对话框（CourseScheduleWindow, StatisticsWindow, StudySessionDialog 等）承载。
2. 业务逻辑与数据持久化层：主要由 C++实现。核心是 DatabaseManager 单例类，它封装了所有与 SQLite 数据库的交互，实现了对日程任务和自习记录的增、删、改、查操作。各个 UI 窗口通过调用 DatabaseManager 的接口来存取数据。
3. 外部数据获取层：由两个独立的 Python 脚本构成，负责从网络获取特定数据。C++通过 QProcess 调用这些脚本, 并通过标准输入输出 (stdin/stdout) 与它们进行 JSON 格式的数据交换。
    - scraper.py: 负责解析用户提供的 HTML 课表文件。
    - query_free_room.py: 负责调用北大校内门户 API 查询空闲教室信息。

2.2 核心类设计详解

以下是项目中关键类的设计细节：

类名	主要职责	所在文件
MainWindow	作为应用程序的主窗口和入口，负责整体布局和核心调度。管理着日历、日程详情显示、天气模块以及到其他功能窗口（如课表、统计）的入口按钮，还实现了“打字机效果”的动态标题栏	mainwindow.h, mainwindow.cpp
DatabaseManager	采用单例模式设计的数据库管理核心，封装了对 tasks.db 文件的 SQLite 数据库操作，包括创建 tasks 表和 study_sessions 表，以及对这两个表的增、删、改、查接口	DatabaseManager.h, DatabaseManager.cpp
CourseScheduleWindow	课表管理窗口，支持通过拖拽 HTML/TXT 文件导入课表，启动 scraper.py 子进程解析文件并用 JSON 数据填充课表视图，点击空白单元格可查询空闲教室，课表数据会保存	CourseScheduleWindow.h, CourseScheduleWindow.cpp
DailyTaskDialog	添加、修改和删除每日日程的对话框，左侧是任务列表，右侧是任务详情编辑区，通过 DatabaseManager 与数据库同步数据	DailyTaskDialog.h, DailyTaskDialog.cpp
StudySessionDialog	自习计时器对话框，启动后显示计时器和动画 GIF，结束自习时将相关时间数据通过 DatabaseManager 保存到 study_sessions 表中	StudySessionDialog.h, StudySessionDialog.cpp
StatisticsWindow	学习统计分析窗口，从 DatabaseManager 获取自习记录并每日汇总，可在条形图和折线图间切换，可更改统计时间单位，无数据时显示提示页面	StatisticsWindow.h, StatisticsWindow.cpp
TaskReminderDialog	日程提醒对话框，用于集中展示所有未来的任务，从数据库获取任务日期，根据距离当前日期远近分类和着色显示	TaskReminderDialog.h, TaskReminderDialog.cpp

类名	主要职责	所在文件
SmartRoomWidget	独立的空闲教室查询窗口，支持用户选择教学楼、查询日期及具体上课节次，调用 query_free_room.py 脚本进行查询，并将结果以表格形式清晰展示	smartroomwidget.h, smartroomwidget.cpp

2.3 外部 Python 脚本

- scraper.py: 该脚本接收一个 HTML 文件路径作为命令行参数。它使用 BeautifulSoup 库解析 HTML 内容，定位到 ID 为 subtable 的课表元素，然后遍历表格，提取课程名称、教室、教师、上课时间等信息，并将其构造成一个 JSON 数组字符串，最后输出到标准输出流。
- query_free_room.py: 该脚本接收教学楼名称和查询日期（如"今天"）作为命令行参数。它使用 requests 库向北大的公共查询 API 发送一个 GET 请求，获取指定条件的空闲教室数据。脚本随后将返回的 JSON 数据进行处理，整合成一个更适合 C++程序使用的、以教学楼和日期为键的嵌套 JSON 对象，并输出到标准输出流。

(3) 小组成员分工情况：

李準珩：

- **课表与教室查询模块**：负责 CourseScheduleWindow 的全部设计与实现，包括课表的可视化、拖拽文件导入功能。
- **课表数据解析**：独立编写了核心的 scraper.py 脚本，使用 BeautifulSoup 库实现了对复杂课表 HTML 文件的精准解析。
- **学习统计与日程提醒**：负责 StatisticsWindow 的数据可视化图表以及 TaskReminderDialog 提醒窗口的开发。
- **辅助功能与优化**：负责 MainWindow 中天气预报模块的开发，并对项目整体 UI/UX 进行美化和调整，修复了多个模块的 Bug，提升了应用的稳定性与用户体验。

汤伟杰：

- **项目整体架构设计**：负责项目的底层框架搭建，确立了基于 Qt 的 C++主程序结构，并设计了与 Python 子进程通过 JSON 进行数据交换的通信方案。
- **数据持久化层**：设计并实现了 DatabaseManager 单例类，负责 SQLite 数据库的全部操作，为上层业务提供了稳定可靠的数据接口。
- **核心业务逻辑**：负责 DailyTask 数据模型和 DailyTaskDialog 的开发，实现了对每日日程的增、删、改、查核心逻辑。同时负责 StudySessionDialog 的开发，实现了学习计时器的核心功能。
- **API 数据对接**：负责 SmartRoomWidget 的设计与实现，并编写了 query_free_room.py 脚本，通过调用北大门户 API，为前端的空闲教室查询功能提供了完整的解决方案。

(4) 项目总结与反思

4.1 项目优点

1. 高度集成与实用性：项目成功地将课表管理、任务提醒、学习计时和校园资源查询等北大学生高频使用的功能整合到一个应用中，解决了需要在不同应用或平台间切换的痛点，具有很高的实用价值。
2. 清晰的模块化设计：项目的架构设计良好，职责分离清晰。UI 层、业务逻辑层和数据层解耦得当。特别是通过 DatabaseManager 单例类统一管理数据库访问，以及将复杂的网络请求和页面解析任务外包给 Python 脚本，使得 C++主程序逻辑更纯粹、更易于维护。
3. 良好的用户体验：应用提供了全图形化界面，操作直观。拖拽导入课表、点击查询教室、数据可视化图表等设计都极大地提升了用户体验。主界面中的 GIF 动画和动态标题等趣味性元素也让软件更具吸引力。
4. 混合编程的成功实践：项目巧妙地结合了 C++和 Python 的优点。C++ (Qt) 保证了桌面应用的性能和原生体验，而 Python 则凭借其强大的第三方库（requests, BeautifulSoup）快速实现了网络数据处理功能。这种务实的技术选型是项目成功的关键。

4.2 可改进之处

1. 环境依赖与部署复杂性：当前应用强依赖于用户本地的 Python 环境以及 requests、BeautifulSoup 等库的安装。这为应用的打包和分发带来了挑战。未来的改进方向可以是：使用 Qt 自身的网络和 XML/HTML 解析库（尽管可能更复杂）重写 Python 脚本的功能，或者将 Python 解释器和必要的库一同打包到应用中，实现绿色免安装。
2. 错误处理与鲁棒性：虽然 Python 脚本中包含了一些异常捕获，但 C++调用端对子进程执行失败（如 Python 未找到、脚本文件丢失）的处理可以更完善。例如，可以提供更明确的错误信息和修复建议给用户，而不是仅在调试控制台输出错误。
3. API 密钥的硬编码：项目将 OpenWeatherMap 的 API 密钥硬编码在了 mainwindow.cpp 中。这是一种不安全的做法，在开源项目中尤其危险。更好的做法是将其存储在外部配置文件或环境变量中，程序启动时读取。
4. UI 主题适配性：README.md 中提到软件可能不适配深色主题。可以通过使用更灵活的 Qt 样式表（QSS）或 Qt 6 中引入的样式系统来改进 UI，使其能更好地适应不同的操作系统主题。

4.3 设计与技术选型反思

本项目是对“如何构建一个功能丰富的现代化桌面应用”的优秀探索。技术选型非常合理：Qt 作为经典的 C++ GUI 框架，功能强大且稳定；SQLite 作为轻量级的本地数据库，无需配置，完美契合单机应用的需求；Python 作为“胶水语言”，高效地完成了它最擅长的数据处理任务。整个开发过程体现了清晰的工程化思想，从 UI 设计到后台逻辑，再到数据持久化，每个环节都有专门的类或模块负责。这种设计不仅使得开发过程更加顺畅，也为未来功能的扩展和维护打下了坚实的基础。总而言之，该项目是一个功能完善、设计合理且具有实际应用价值的成功作品。

此外后期我们加入了一些小巧思，如上方的打字机效果、动图的嵌入、天气的报道等等，发现原来看似十分复杂的内容，在学习之后也是可以用较少的代码完成的。

缺点是：由于小组人员的减少（田汇川中期退课）导致时间紧迫，没有完成预期的所有功能；此外，这篇报告大部分也是使用 ai 辅助撰写的，不过代码的原始内容是保证由小组队员协

作完成，并在后期使用 ai 进行格式的修正和规范。如果时间足够，后续我们会考虑加入智能推荐系统，并对 ui 进行进一步的完善。