《程序设计实习》Qt 大作业
作业报告

李明泽[1], 王晨旭[2], 王度[3]

2024 — 2025 学年第二学期

[1] 信息科学技术学院, 李明泽, 2400013220;
[2] 信息科学技术学院, 王晨旭, 2400013225;
[3] 地球与空间科学学院, 王度, 2400012446;

目录

1 程序功能介绍

我们开发的软件名为 PKUCourseRadar，是一个帮助用户选择、规划旁听课程的软件。

我们的软件包含以下核心功能：

导入课程信息 本项目支持自定义课程信息的导入。这使得软件可以支持使用不同学期甚至不同学校的动态课表，具有很强的适应性和实用性。

多维度课程检索 导入用户课程信息后，可通过搜索特定课程名称、授课教师、教学楼和特定教师查找相应课程，也可根据一个或多个特定标签筛选课程，找到自己心仪的课程。

规划旁听课表 点击课表单元格即可看到在当前时间段的所有课程，支持查看详细课程信息、收藏、选中（置顶）等操作。用户可在"我的课程"中集中管理所有已收藏课程，并对同一时间段的课程进行置顶、取消收藏等操作。

多用户支持与课程评价系统 内置登录注册与课程评价系统，用户可查看其他用户的课程评论，并支持对评论进行点赞与点踩，从而构建互动的课程评价社区。

多语言支持 在登录界面可切换中英文显示，满足不同语言用户的需求。

2 项目各模块与类设计细节

2.1 CourseManager

`CourseManager` 是一个采用单例模式的全局对象 (`theManager`)。这意味着在整个应用的生命周期中，只有一个 `CourseManager` 实例存在。这种设计是为了提供一个统一的数据访问入口，避免在众多窗口之间层层传递数据对象的复杂性。此外，每个用户的全部信息都可以用一个 `CourseManager` 对象表示。因此，实现用户的登录时，只需从文件中把用户信息加载到 `CourseManager::theManager` 中，即可直接通过 `CourseManager` 进行操作。而用户退出登录时，再把 `CourseManager` 的信息写入数据库即可。

下面介绍 `CourseManager` 中值得一提的实现细节。

`QMap<QUuid, Course> AllCourses`

`QMap<QUuid, Course> AllCourses` 是存储所有课程信息的数据结构。选用这样的结构，可以将所有 `Course` 对象存储于这个结构中，而在程序的其它部分均使用 `QUuid` 来代表一门课程。这样做的好处是，当程序内部需要传输课程信息时，可以统一使用值传递 `QUuid`，而不会带来过大的复制开销，并且保证了 `AllCourses` 中作为值的 `Course` 对象是代表这一课程的唯一 `Course` 对象；相反，如果使用 `QSet<Course>` 或 `QVector<Course>` 存储课程，将会导致传递课程信息时，需要考虑选择值传递还是引用传递、指针传递的问题；同时，如果中间有一步不慎使用了值传递，而后续需要对课程信息进行更新，则无法作用到最初的 `Course` 对

象，造成课程信息的不统一。同时，由于存在多个 Course 对象，哪个 Course 是“正统”的也无从得知。使用 QMap 规避了上述问题。

我们还通过在 Course 对象中加入课程 UUID 属性，实现了课程与 UUID 的一一对应和快速查找。

QMap<QUuid, CourseComment> AllComments 也采用了类似的结构，具备以上所述优点。

2.2 数据实体类 (Course, CourseComment, CourseTime)

这些类封装了相关对象的属性和一些辅助方法。下面介绍这三个类中值得一提的实现细节。

Course::description()

这个函数可以返回课程的可读描述，将复杂的字符串格式化部分在 Course 内部即实现，避免了多行长字符串对需要课程详情的相关代码可读性的干扰。同时也具备了良好的可扩展性。

CourseComment::format() 具备类似的功能。

CourseComment::likes

与其简单粗暴地使用 int 类型存储 likes ，我们使用了 QSet<QUuid> 来存储点赞信息。这样的好处是，确保每个用户只能点一次赞，并且用户取消点赞时，也可以以较低的复杂度将用户从点赞列表中删除。

CourseComment::dislikes 同理。

2.3 数据库读取接口 DataAccessor

DataAccessor 是一个抽象类,实现了四个接口:用户登录、用户注册、从文件读取 CourseManager、将 CourseManager 写入文件。

事实上,整个程序中,我们使用的均为 JsonDataAccessor。那为什么还要实现 DataAccessor 这个抽象类呢? 这样做的好处是，利用 C++ 的多态，只要一个类继承自 DataAccessor，就可以作为这个程序读写数据库的接口。这保证了程序良好的可扩展性：例如，如果未来要把这个项目接入网络，实现真正的多用户交互，实现服务端的相关代码后，我们只需实现一个 WebDataAccessor，然后在 main 中把 DataAccessor::theDataAccessor 初始化为一个指向 WebDataAccessor 的指针即可。也可以多种数据库读取方法相结合。

DataAccessor 主要在 LoginWindow 中调用，以处理用户管理与用户登录后对数据库的读取；在 HomeWindow 关闭后，也会调用 DataAccessor，将 CourseManager 写入数据库。

2.4 UI 与交互逻辑

本项目并不依靠大量窗口的堆叠取胜，而是对每个窗口进行精心设计与布局，展示尽可能详尽的信息，并且充分利用鼠标的单击、双击、右键等事件，尽可能简化交互所需的操作次数，实现更简捷的用户体验。

2.4.1 `LoginWindow` - 用户注册与登录

本窗口为程序入口。输入用户名和密码后，程序将会对密码进行 MD5 编码并与数据库中的数据进行比对：如果用户名不存在，就直接注册这个账号；否则，如果用户名存在但是密码不对，则弹出提示框，提示密码错误；如果用户名和密码均正确，则正常登录。

本窗口具有导入、导出 Json 数据库的功能，这保证了程序的适应性，可以支持使用不同学期甚至不同学校的动态课表，也可适用于不同的用户团体或组织。

本窗口还有一个切换显示语言的按钮。具体请参见2.6。

2.4.2 `HomeWindow` - 导航窗口

此窗口为程序的导航窗口，通向两个主要功能：“查找课程”和“我的课程”，以及“关于我们”和“退出登录”。窗口左侧为两张北京大学的图片，使用 `.qrc` 资源文件嵌入程序，彰显北大特色。

2.4.3 `RadarWindow` - 课程发现与筛选

本窗口是本项目的核心：通过课程的具体信息和标签来筛选课程，帮助用户快速检索到旁听课程。

窗口左侧是筛选条件，右侧是一个单元格组成的可操作课程表，实时显示每个单元格的可用课程数量，点击单元格便可弹出 `CourseInfoWindow` 展示这些课程的详细信息，并进行更多操作。只要筛选条件发生任何改变，均会通过信号-槽触发 `reSearchCourses` 方法。这个方法会根据新的筛选条件，计算有多少门课程占用了某个单元格，并将这个数量设置为该单元格的显示文本，帮助用户快速查看各时段的课程数量信息。

值得一提的是 tag 浏览器的一个实现细节：当 tag 从一个列表移动到另一个列表时，两个列表均会根据两个 `QSet` 完全重新加载一次。这样做的目的是，保证一个 tag 在两次操作，回到原来所属列表时可以回到自己所在位置，而不是插入到列表的结尾。这使得 tag 的选取和删除没有割裂感，提高了用户体验。

2.4.4 MyCoursesWindow & CourseInfoWindow - 课表展示与修改

MyCoursesWindow 负责展示用户的课表。syncCells 方法为其核心,用来把 CourseManager 中的信息显示在课表上。点击课表的单元格即可对课程进行收藏、取消收藏、选中（置顶）等操作。

CourseInfoWindow 则是一个在程序中多次使用的信息面板，用于展示特定时间段的课程详细信息。得益于上面提到的 AllCourses 的结构，它在初始化时接收一个课程 ID 列表和时间信息，而不必复制课程对象。

由于课程的选中（置顶）的必要条件是该课程被收藏，我们实现了如下的逻辑:

当用户右键单击列表中的某个课程项时，会触发 onItemRightClicked 事件，该事件会直接修改 CourseManager 中对应课程的收藏状态 (marked 属性)。如果取消收藏的课程恰好是某个时间段的已选课程，则会将其从 selectedCourses 中移除。随后，sync_list() 方法会被调用，刷新列表以显示或隐藏收藏标记，并通过 emit changed() 信号通知父窗口数据已更改，需要刷新。

双击课程项则会触发 on_l_list_itemDoubleClicked 事件。该事件首先检查课程是否已收藏，未收藏的课程不能被选择。然后，它会修改 CourseManager 中对应时间段的已选课程，实现选课或取消选课的功能。同样，sync_list() 方法会被调用以刷新列表（显示或隐藏选课标记），并通过 emit changed() 信号通知父窗口。

sync_list() 方法还承担着根据课程均分对课程进行排序的工作，将高分课程排在前面。如果课程没有评价，我们将 4.2 分（百分制下的 84 分）作为课程的默认评分。

这样，我们便在不添加额外按钮的情况下，简洁地完成了两种操作的实现。同时，课程名称前面会显示一个对应的 emoji 来代表某门课程的状态（是否收藏、是否选中），提供视觉反馈。另外，本窗口也作为评论浏览器的入口，选中课程点击右上角按钮，即可查看该课程的课程评价。这样的结构，统一了 RadarWindow 和 MyCoursesWindow 的课程信息展示窗口，避免了重复实现。

2.4.5 CommentBrowserWindow & AddCommentWindow - 课程评论

评论浏览器 (CommentBrowserWindow) 允许用户查看课程评论并进行互动。这里也使用了类似的使用 sync_list() 方法的处理方式。并且此处的 sync_list() 也会根据评论的“净推荐值”（点赞数减去点踩数）进行降序排序。

用户可以通过点击“赞”和“踩”按钮，直接修改 CourseManager 中对应评论的 likes 和 dislikes 集合。数据修改后，sync_list() 会被立即调用，从而重新获取数据、重新排序、并重新显示列表，为用户提供即时的视觉反馈。

AddCommentWindow 是用来撰写课程评价的窗口。窗口上方使用 QSlider 实现了课程 rating 的选取。

2.5 自定义 UI 组件

我们的项目使用了一些自定义 UI 组件，以实现一些更美观、更符合项目需求的控件。

CourseTableWidget

CourseTableWidget 是一个核心的容器组件，使用 QGridLayout 创建了一个 7x12 的 CourseCell 网格。它的主要职责是创建并布局所有的 CourseCell，并将每个 CourseCell 的 clicked 和 rightClicked 信号连接到自身的 handle... 槽函数。这些槽函数的作用是从 sender() 获取发出信号的 CourseCell，然后将这个信号连同该单元格的坐标 (x, y) 再次 emit 出去，供 RadarWindow 或 MyCoursesWindow 使用。它起到了一个信号中继和坐标附加的作用。

CourseCell

CourseCell 是一个可用于显示各种信息的课程单元格。它内部维护了 disabled、canDisable 等状态，并使用 paintEvent 负责绘制背景和边框。背景色 currentColor 是一个动态变量，通过动画效果实现平滑的颜色过渡。enterEvent、leaveEvent、mousePressEvent 等事件处理器不直接改变颜色，而是设置并启动一个 QVariantAnimation。这个动画的目标值是 hoverColor、pressedColor 等。动画的 valueChanged 信号连接到 onAnimationValueChanged 槽，该槽负责更新 currentColor 并调用 update() 请求重绘，从而实现了平滑的颜色过渡效果。

BetterButton

BetterButton 是一个较复杂的自定义绘制的按钮组件。此组件存在多种特殊效果：

线性过渡 按钮未悬停、未点击时，会显示一个从左上角到右下角的线性过渡效果。

径向过渡 当鼠标悬停或点击按钮时，会显示一个以鼠标位置为中心的径向渐变效果，创造出光泽跟随鼠标的效果。

阴影效果 在 paintEvent 中，createBlurredShadow 被调用两次，一次生成浅色阴影（偏移到左上），一次生成深色阴影（偏移到右下）。这两个阴影用于模拟光照效果，增强按钮的立体感。阴影颜色会根据按钮的状态（默认/悬停/点击）进行调整。

文本位移 按下时，按钮文本会有一个微小的位移，模拟物理按压感。

动画效果 颜色和阴影的过渡使用动画，使得按钮状态变化更加平滑自然。动画时长可以自定义。

可配置性 按钮的背景颜色、圆角大小、内容内边距、渐变颜色、阴影颜色、阴影偏移、模糊半径等参数均可配置，以满足不同的界面设计需求。

`BetterMessageBox`

设计这个类的目的是用来替代 Qt 自带的 `QMessageBox`。主要原因是 `QMessageBox` 采用的是默认的 `QPushButton`,与其它界面广泛使用的 `BetterButton` 风格不符。`BetterMessageBox` 中手动实现了一个采用 `BetterButton` 的信息提示框。

2.6 多语言支持

我们利用 Qt Linguist 提供了程序的中英文切换功能。程序中大多数类都继承自 `QObject` 并包含 `Q_OBJECT` 宏，因此可以使用 `tr` 函数标记需要翻译的字符串。值得一提的是，我们使用了 `xxd` 来将二进制的`.qm` 文件转换为头文件，并在需要切换中英文的部分包含这些头文件。这使得我们的程序在发布、传输时，不需要额外携带翻译文件，并且可以在一定程度上防止翻译文件被恶意篡改。

设计上，我们在 LoginWindow 中即提供语言切换按钮，并且使用地球图标加以指示，以便非中文母语者在默认中文的情况下确定切换语言按钮的位置。

3 小组成员分工情况

王度　数据库结构的设计；`DataAccessor`、`JsonDataAccessor` 的实现；`CourseManager` 及其内部结构的设计与实现；`LoginWindow` 的设计与实现；`HomeWindow` 的实现；上述四个自定义控件的设计与实现；`RadarWindow` 的设计与实现；`CourseInfoWindow` 的设计与实现；“我的课程”功能的优化；`AboutUs` 窗口的设计与实现；课程评论、评分功能及两个相关窗口的设计与实现；多语言支持的实现；作业报告第二部分“项目各模块与类设计细节”的撰写。

王晨旭　“我的课程”部分的构思与初步实现；“开始”和“我的课程”部分中 bug 的修复；项目效果及隐含问题检测。

李明泽　项目构型；设计 HomeWindow 界面；提取、处理北大全校课表信息；对接联络助教；路演参选和最终演示两个视频的录制；作业报告初版的撰写。

4 项目总结与反思

本项目成功构建了一个功能完备的旁听课程选择与规划软件 PKUCourseRadar。通过实现课程导入、多维度检索、个性化课表规划、以及用户社交评价等核心功能，我们极大地便利了用户发现、选择和管理旁听课程。

在技术实现上，我们通过精心设计的模块划分、数据结构（如采用 `QMap<QUuid, Course>` 管理课程数据，有效避免了数据冗余和一致性问题）和抽象接口（如 `DataAccessor`），使得项

目在代码层面具备了良好的可维护性和可扩展性，为未来集成更多数据源或实现网络化服务打下了基础。自定义 UI 组件（如 `BetterButton` 和 `CourseCell`）的应用也提升了用户界面的美观度和交互体验。

在项目开发过程中，我们也积累了宝贵的经验，主要体现在以下几个方面：

前期规划的重要性与前瞻性 项目初期，我们没有预见到后期对多用户、评论体系等功能的需求，导致数据库结构设计时未预留相关字段。这直接造成了后期的数据库结构调整、数据迁移以及相关读取模块的重写，耗费了额外的时间和精力。这提醒了我们：在项目启动之初，进行充分详尽的需求分析、功能清单罗列和具备一定前瞻性的架构设计是至关重要的，它能够有效规避后期可能出现的大规模重构风险，提高开发效率。

技术选型与权衡 在自定义 UI 组件的开发上，例如 `BetterButton` 的复杂动画和阴影效果实现、`CourseTableWidget` 等，虽然投入了较多时间和精力，但最终增强了项目的视觉吸引力和用户交互感。这启发我们思考软件开发过程中技术实现深度与用户体验提升之间的权衡取舍。有时为了达到更优质的用户体验，投入额外的时间在细节打磨上是值得的。

复杂交互逻辑的实现与状态管理 `RadarWindow` 和 `MyCoursesWindow` 中涉及的课程筛选、课表同步、收藏/选中逻辑以及 `CourseInfoWindow` 的数据传递，都要求 UI 层与数据管理层（`CourseManager`）之间保持高度同步。实现这些复杂交互时，我们通过信号-槽机制和 `syncCells`/`sync_list` 等同步方法，维护了数据和 UI 状态的一致性，但也认识到复杂多窗口程序中，对数据流和 UI 刷新机制进行精细化管理是一个较大的挑战，需要严谨的逻辑设计，才能保证数据展示的一致性。

展望未来，本项目仍有广阔的开发空间。例如，可以进一步优化课程推荐算法，使其更智能化地为用户匹配感兴趣的课程；可以接入 LLM 进行自动化的 tag 生成和课程简介生成；将抽象的 `DataAccessor` 实例化为一个新的 `WebDataAccessor`，真正对接至后端服务，实现多用户数据的云端存储和实时同步，从而构建一个真正的多用户社区；利用 Qt 的跨平台特性，开发移动端版本，以适应更广泛的使用场景。我们相信，通过持续的迭代和优化，PKUCourseRadar 将能更好地服务于北大学子乃至全国大学生，成为他们高效规划旁听课程的得力助手。