PKUStudyHelper技术报告

by Bug别和我做队

[TOC]

摘要

PKUStudyHelper 是一个面向北京大学学生的综合学习管理系统，集课程管理、任务规划、学习资源整合于一体，通过Qt框架实现了跨平台的桌面应用开发，，帮助北大学生高效管理学业生活，提升学习效率。

程序功能介绍

主要功能:

- **注册时：根据北大门户账号和密码** 自动导入课程信息、日程表信息等
- **课程信息：** 可以查看课程名称、上课时间、教室等信息
- **任务和ddl管理：** 显示各种任务，自动计算距离ddl的时间，在时限将近时提醒用户完成
- **待办事项管理** ：如每日习惯等

这里按照不同界面来详细介绍

1. 登录界面

- 输入账号和密码，点击登录键以登录
- 点击注册键进入注册界面
- 界面有提示机制，当异常输入的时候会发出错误信息

2. 注册界面

- 输入昵称，账号，重复输入两次密码即可注册
- **注意** ：为了更好的用户体验，这里账号和密码需要是北大门户的账号密码
- 程序会根据用户注册时输入的账号和密码爬取用户的课程信息、课程表和gpa信息

3. 主页

- 界面的顶部会显示问候语
- 左栏是我的课程，显示这一天的课程，同时会有光标追踪目前所属的时间段
- 右栏是我的任务，是任务栏的一个微缩视图
- 左侧导航栏的按键可以跳转到任务界面和课程界面

4. 任务界面

- 可以创建任务，任务的创建需要截止日期和任务名称两项信息，截止日期可以自己编辑，也可以使用日历视图快速选定
- 任务会根据截止日期进行排序，越临近的任务越靠前
- 根据截止日期可以将任务分为今天截止、三天内截止、一周内截止、其他
- 右键单击任务可以标定已完成和删除，已完成的任务会被划线并且立刻移至最末端

5. 课程界面

- 课程界面主要分为详细的课程表和考试轴建立
- 课程表是基于用户的门户课程信息创建的，可以快速浏览课程信息
- 考试轴是用来管理考试日程的，用户可以设置具体的考试时间、地点来创建一个考试项目。
- 考试项目会按照时间排序，如果考试在今天，还会有红色效果加以警告

6. 课程主页

- 每门课程都有属于自己的课程主页
- 课程主页里包括三个主要控件：课程信息、课程备忘录、课程作业
- 课程信息是根据用户的门户信息整理得来的，是系统自动导入的
- 课程备忘录则是一个可以由用户编辑的文本框，可以方便用户储存一些重要信息
- 课程作业功能其实是任务功能的一个子集，因为所有的作业都会被加入任务栏进行管理，同时课程主页也会显示属于这门课程的作业

项目各模块与类设计细节

1.系统模块划分

- 用户认证模块：处理用户登录和注册
- 主页模块：显示用户信息、今日任务、课程表概览
- 课程管理模块：显示课程详情、课程安排
- 任务管理模块：管理学习任务和DDL
- 爬虫集成模块：在QT中调用python爬虫程序，与北大校内门户交互，获取课程信息，课程表和成绩

2.核心类设计

2.1 数据模型类

User 类

- **功能** ：存储用户信息
- 成员变量：
  - `QString account` ：学号
  - `QString password` ：加密存储的密码
  - `QString username` ：用户昵称

ParsedScheduleResult 结构体

- **功能** ：存储解析后的课程表结果
- 成员：
  - `QMap<QString, QVector<CourseInfo>> name_index` ：按课程名索引的课程信息
  - `QMap<QString, QVector<CourseInfo>> time_index` ：按星期索引的课程信息

- `QVector<QString> name_list` ：课程名列表（去重）

2.2 界面类

LoginDialog 类

- **功能**：实现用户登录界面，支持用户输入账号密码并验证，支持跳转注册页面并响应注册结果。
- **界面组件（通过 `ui_logindialog.ui` 管理）**：
  - `QLineEdit *account` ：账号输入框
  - `QLineEdit *password` ：密码输入框（应设置 `EchoMode` 为密码）
  - `QPushButton *loginButton` ：登录按钮
  - `QPushButton *registerbutton` ：注册按钮
  - `QLabel *label` ：用于显示提示信息（登录成功、失败、注册结果等）
  - `QLabel *label_image` ：头像/图标区域，设置了阴影特效
- **关键功能实现**：
  - 阴影效果：使用 `QGraphicsDropShadowEffect` 为 `label_image` 添加阴影，增强视觉效果
  - 登录验证逻辑：
    - 检查账号和密码是否为空
    - 首次查询账号是否存在： `SELECT * FROM user WHERE account = :account`
    - 若存在，再验证密码是否正确： `SELECT * FROM user WHERE account = :account AND password = :password`
    - 登录成功后设置全局变量 `current_user` ，并调用 `accept()` 关闭窗口
  - 注册弹窗逻辑：
    - 弹出 `Register` 窗口（模态），监听注册成功与取消信号
    - 注册成功后提示“注册成功，请登录”
    - 注册取消提示“注册已取消”
- **信号与槽**：
  - `on_loginButton_clicked()` ：执行登录验证流程
  - `on_registerButton_clicked()` ：打开注册窗口并连接信号
  - `on_register_success()` ：注册成功后提示绿色文字
  - `on_register_cancel()` ：注册取消后提示黑色文字

Register 类

- **功能**：实现用户注册界面，支持输入用户名、账号、密码并完成注册，同时自动爬取课程表并存入数据库。
- **界面组件（通过 `ui_Register.ui` 管理）：
  - `QLineEdit *newname` ：用户名输入框
  - `QLineEdit *newaccount` ：账号输入框
  - `QLineEdit *newcode` ：密码输入框
  - `QLineEdit *confirmpassword` ：确认密码输入框
  - `QPushButton *yes` ：确认注册按钮

- `QPushButton *cancelButton` ：取消按钮
- `QProgressBar *progressBar` ：注册进度条，初始隐藏
- `QLabel *statusLabel` ：状态提示标签，初始隐藏
- `QLabel *message` ：提示信息标签（如错误提示）

- **关键功能实现** ：
  - 按钮连接：
    - 确认按钮连接 `on_confirmButton_clicked()`
    - 取消按钮连接 `on_cancelButton_clicked()`
  - 注册流程：
    - 校验输入不能为空，密码和确认密码必须一致
    - 显示进度条和状态标签，禁用按钮防止重复点击
    - 异步执行数据库检查和插入，避免界面卡顿
    - 检查用户名是否已存在，若存在提示错误
    - 检查账号是否已存在，若存在提示错误
    - 插入新用户账号和密码到 `user` 表
    - 调用爬虫函数 `fetchSchedule(account, password)` 获取课程表
    - 解析并逐条插入课程表数据至 `courses` 表
    - 最后完成注册，进度条更新至100%，延迟关闭窗口并发出 `registerSuccess()` 信号

- **信号与槽** ：
  - `registerSuccess()` ：注册成功信号，通知登录窗口更新状态
  - `registerCancel()` ：取消注册信号

- **外部依赖** ：
  - `extern QSqlDatabase db` ：数据库连接
  - `fetchSchedule(account, password)` ：外部课程爬虫函数，返回课程表数据
  - `ParsedCourseInfo parseCourseInfo(QString)` ：外部解析函数，用于分解课程详情字符串

MainWindow 类

- **功能** ：应用程序主窗口，负责搭建整体界面结构，包括左侧导航栏与右侧功能页面的切换。
- **核心成员** ：
  - `QListWidget *navList` ：左侧导航栏，用于选择“我的主页”、“我的任务”、“我的课程”及各课程主页
  - `QStackedWidget *stackedWidget` ：右侧页面切换区域，按导航栏项切换对应页面
- **核心方法** ：
  - `void setupUI()` ：构建主界面，包括导航栏、主页、任务页、课程页和课程主页
  - `void on_item_clicked(QListWidgetItem *item)` ：点击导航栏项后，根据索引切换 `stackedWidget` 页面
- **初始化页面** ：
  - `HomePage *homePage` ：展示问候语、今日任务和DDL表
  - `TaskPage *taskPage` ：任务添加与管理页面
  - `ClassPage *classPage` ：课程总览页面

    - `CoursePageTemplate *coursePage` ：每门课程的主页（动态添加）
- **数据库交互** ：
    - 通过 SQL 查询 `courses` 表中当前用户的课程名称，为每门课程生成一个 `CoursePageTemplate` 页面
- **信号连接** ：
    - `TaskPage::tasksChanged` 连接到 `HomePage::refreshTasks` ：任务更新后刷新首页
    - `CoursePageTemplate::assignmentsChanged` 分别连接到 `TaskPage::refreshTasks` 和 `HomePage::refreshTasks` ：课程作业更新后刷新任务和主页显示
- **特点** ：
    - 左侧导航栏图标+文字风格统一，右侧使用 `QStackedWidget` 实现不同页面之间的无缝切换
    - 所有课程页面在启动时动态从数据库加载生成，支持扩展
    - 当前选中的页面会被高亮，并默认进入“我的主页”

HomePage 类

- **功能** ：展示用户主页，包括欢迎语、今日课程表和任务一览。
- **核心成员** ：
    - `QTextBrowser *greetingBrowser` ：显示欢迎语与每日鸡汤名言
    - `QTableWidget *courseTable` ：展示“今日课程”的时间、课程名和教室
    - `MiniTaskPage *miniTaskPage` ：嵌入任务模块，展示简化版任务表
    - `QString userName, userAccount` ：当前用户信息
- **核心方法** ：
    - `QString getGreetingText(const QString& username)` ：根据当前时间生成“早上好/下午好/晚上好”的问候语
    - `QString getRandomQuote()` ：从固定语录集中随机返回一句每日鼓励
    - `void loadTodayCourses()` ：从 SQLite 数据库中加载当前星期的课程信息，并展示在课程表中，当前课程高亮
- **布局结构** ：
    - **顶部** ：欢迎语（QTextBrowser）
    - **中部横向分为两栏** ：
        - 左侧：今日课程表（QTableWidget）
        - 右侧：任务展示（MiniTaskPage）
- **样式特点** ：
    - 背景使用半透明纸张图像
    - 所有控件均设有美化的字体和圆角背景
    - 当前正在进行的课程自动高亮
- **数据库交互** ：
    - 通过用户账号 `userAccount` 查询 `courses` 表，筛选今日课程
    - 课程信息包括节次（period）、课程名（course_name）、教室（classroom）
- **扩展性** ：
    - 可根据数据库返回的课程动态更新课程表
    - 欢迎语和任务区模块可独立扩展样式与逻辑

TaskPage 类

- **功能** ：任务管理页，允许用户创建、查看、分类、完成和删除任务。
- **核心成员** ：
  - `QDateEdit *dateEdit` ：任务截止日期输入框
  - `QLineEdit *nameEdit` ：任务名称输入框
  - `QPushButton *addButton` ：添加任务按钮
  - `QListWidget *taskList` ：任务列表展示区域，支持按状态分类
  - `QString userAccount` ：当前登录用户账号，用于数据库操作
  - `QVBoxLayout *mainLayout` ：主布局，包含输入区域与任务列表
- **核心方法** ：
  - `void addTask()` ：向当前用户的任务 JSON 中添加一条新任务并刷新显示
  - `void loadTasks()` ：从数据库中加载当前用户任务数据，分类展示，并发送 `tasksChanged()` 信号通知外部
  - `void saveTasks(const QJsonArray& array)` ：将任务数据保存至数据库（JSON 格式）
  - `void markTaskDone()` ：将选中任务标记为完成
  - `void deleteTask()` ：删除选中的任务项
  - `QJsonArray getSortedTasks(const QJsonArray &array)` ：按完成状态和截止时间对任务排序
  - `QString deadlineStatus(const QDate &date)` ：根据当前时间判断任务的时间紧迫状态
- **信号** ：
  - `void tasksChanged()` ：当任务添加、删除或修改后触发，用于通知主页刷新任务列表
- **槽函数** ：
  - `void showContextMenu(const QPoint &pos)` ：右键点击任务时弹出菜单（“已完成”“删除”）
- **特点** ：
  - 任务列表支持按“今天截止”、“三天内截止”、“一周内截止”、“其他任务”、“已完成”自动分类
  - 支持 strike-out 效果标记已完成任务，并用红色高亮“今天截止”的任务
  - 所有数据以 JSON 格式存储在用户表的 `json_tasks_content` 字段中，具有良好的扩展性

ClassPage 类

- **功能** ：展示当前用户的完整课程表与考试时间轴，支持考试信息的添加与删除。
- **构造参数** ：
  - `const QString& account` ：当前用户账号，用于数据库筛选
- **主要组件** ：
  - `QTableWidget *courseTable` ：展示 7 天 × 11 节课的课程表
  - `QListWidget *examListWidget` ：考试时间轴卡片区域（水平排列）
  - `QPushButton *addExamBtn` ：添加考试按钮
- **界面布局** ：
  - 顶部：课程表（带节次与星期标签）
  - 中部：考试时间轴标签 + 添加按钮

- 底部：考试卡片（按时间升序排列）

- **主要方法** ：
  - `void loadCourseTable()` ：从 `courses` 表加载所有课程数据，填充课程表并合并相邻单元格
  - `void mergeCourseCells()` ：自动合并连续相同课程的表格单元
  - `void loadExamList()` ：从 `exams` 表中加载考试信息，并以卡片形式展示
  - `void onAddExamClicked()` ：点击“添加考试”后弹出对话框
  - `void showAddExamDialog()` ：考试添加对话框，支持选择课程、日期、时间与地点，表单校验后写入数据库
  - `void onExamItemContextMenuRequested(const QPoint &pos)` ：右键考试卡片支持删除操作（带数据库更新）

- **数据库表依赖** ：
  - `courses` ：用于获取课程的 `weekday` 、 `period` 、 `course_name` 和 `classroom`
  - `exams` ：用于存储并展示考试的 `exam_date` , `start_time` , `end_time` , `location`

- **扩展性** ：
  - 可根据实际需要扩展支持编辑考试信息或添加作业
  - 卡片样式与交互逻辑可独立升级，支持动画、筛选、颜色标签等功能

CoursePageTemplate 类

- **功能** ：为某门课程生成独立主页，展示课程信息、用户备忘、作业管理等内容，支持作业添加、完成与删除操作。

- **构造参数** ：
  - `const QString& userAccount` ：当前登录用户的账号
  - `const QString& courseName` ：课程名称（带空格前缀）

- **主要组件** ：
  - `QTextEdit *courseInfoEdit` ：显示课程信息（教师、备注、考试信息）
  - `QTextEdit *memoEdit` ：支持用户手动记录的课程备忘录（自动保存）
  - `QPushButton *addAssignmentBtn` ：添加作业按钮，弹出表单对话框
  - `QListWidget *assignmentListWidget` ：当前课程的作业任务列表

- **界面布局** ：
  - 顶部：大标题显示课程名（加粗加大居中）
  - 中部：左为课程信息（不可编辑），右为备忘录（支持编辑、失焦保存）
  - 底部：左为添加作业按钮，右为作业列表展示

- **核心数据库交互** ：
  - 从 `courses` 表中加载课程详情信息
  - 从 `assignments` 表中加载、写入、更新、删除作业信息
  - 从 `user` 表中读取备忘录并保存更新
  - 支持将作业信息同步追加至 `user.json_tasks_content` 中（Json 数组格式）

- **关键方法** ：

- `void loadCourseInfo()` ：加载教师、备注、考试信息并显示
- `void loadMemo()` ：从 `user` 表加载课程备忘信息
- `void saveMemo()` ：在失焦时自动将备忘内容写入数据库
- `void loadAssignments()` ：加载作业数据，构造 `Assignment` 列表并排序展示
- `void refreshAssignmentList()` ：刷新作业展示区域（带完成状态视觉标记）
- `void onAddAssignmentClicked()` ：弹出对话框添加作业，写入数据库并同步 Json
- `void onAssignmentContextMenuRequested(const QPoint &)` ：右键菜单支持“完成/删除”
- `void onMarkAssignmentDone()` ：将作业标记为完成（字体变灰 + 斜体）
- `void onDeleteAssignment()` ：从数据库中删除该作业并刷新列表

- **信号** ：
  - `assignmentsChanged()` ：用于通知外部组件（如主页或任务页）刷新任务数据

2.3 工具类/函数

ScheduleParser 工具类

- **功能** ：提供课程表解析功能
- 方法：
  - `static ParsedScheduleResult parseAndCategorizeSchedule(const QVector<QMap<QString, QString>>& scheduleEntries)` ：解析课表数据
  - `static int parseStartPeriod(const QString& periodStr)` ：解析起始节次
  - `static QString mergePeriods(const QVector<QString>& periods)` ：合并连续的节次

爬虫python函数course.py

- 通过爬虫获取JSON格式的课程表、课程信息和GPA

crawler.h–在QT中调用python函数并将结果捕获

- `QJsonDocument callPythonScript(const QString& functionName,const QString& stuId, const QString& password,bool verifySSL)` 公共函数：调用Python脚本获取数据,并将数据从JSON解析为Qt可用的QJsonDocument对象
- `QMap<int, QMap<int, QString>> fetchSchedule(const QString& stuId, const QString& password, bool verifySSL)` ：以{周几：{课时：课名，}，}格式获取课程表
- `float fetchGPA(const QString& stuId,const QString& password,bool verifySSL,bool* ok )` ：以float形式获取GPA
- `QMap<int, QString> fetchCourses (const QString& stuId,const QString& password,bool verifySSL)` ：以{选的第几门课：课程名}格式获取课程列表

其他函数

`int parseStartPeriod(const QString& periodStr);` 从节次字符串中提取起始节次（数字）

`QString mergePeriods(const QVector<QString>& periods)` 合并连续的节次信息

`QVector<CourseInfo> parseSchedule(const QVector<QMap<QString, QString>>& scheduleEntries)` 将原始数据结构转换为结构化的课程信息列表

`ParsedScheduleResult parseAndCategorizeSchedule(const QVector<QMap<QString, QString>>& scheduleEntries)` 将原始数据结构转换为具有索引结构的课程信息列表

2.4 全局对象

- `current_user` ：当前登录的用户对象
- `db` ：sqlite数据库

小组成员分工情况

- 胡文彬（队长）：处理程序前端，包括设计程序的业务流程和图形界面。完成部分后端，实现注册登录逻辑，sqlite数据库存储用户信息（账号密码、课程信息、任务信息等），任务管理（创建、排序、交互）。完成视频拍摄与讲解。
- 翟凌飞：处理程序后端，尤其是在QT中调用course.py接口的问题。参与后端设计，实现注册时加载用户信息的功能。完成技术报告的撰写。
- 王睿恒：处理网络业务。实现了一个功能强大的course.py爬虫，为程序提供了网络资源的基础。

项目总结与反思

1.项目概述

PKUStudyHelper 是一个面向北京大学学生的综合学习管理系统，通过Qt框架实现了跨平台的桌面应用开发。项目核心功能包括：

1. **课程管理** ：课程表展示、课程详情查看
2. **任务跟踪** ：DDL提醒、任务完成状态管理
3. **门户集成** ：与北大校内门户对接获取课程数据

2.技术实现亮点

- **门户信息集成** ：通过python爬虫直接在北大校内门户获取用户信息，无需手动添加课程，为用户提供个性化体验
- **模块化设计** ：清晰划分用户界面、业务逻辑和数据访问层
- **数据库技术** ：本程序使用sqlite数据库对用户的数据进行存储与维护，可以支持长期运行使用

3. 总结与反思

PKUStudyHelper较为成功的是充分利用了Qt框架优势，包括模块化设计、信号槽机制简化组件通信、以及利用Qt中UI库实现了界面美化。利用数据库实现用户信息的存储与维护也极大地增加了程序的实用性。同时，实现跨c++和python两种编程语言，充分利用python爬虫获得用户课程信息也是设计上的创新。

然而，我们的项目仍有许多不足：

- **功能完善性反思** ：

- 在我们最初的设想中，PKUStudyHelper是集合大量学习辅助功能的应用，其中包括笔记功能，支撑Markdown等多种语言的实时渲染，以及笔记文件的结构性整理（如Obsidian的双向链接等）但我们低估了这一技术的实现难度，因此最终没有完善笔记功能。

- **技术实现反思**
  - 在以C++语言为主的一个程序中调用python对我们来说是一件比较棘手的事。我们尝试过很多方法，甚至包括将python直接翻译成C++这样无知的操作。造成困难的主要原因是对技术的掌握不足，我们一开始并不熟悉数据库的使用，更是很少接触json文件，所以屡屡碰壁。直到后面深入了解了更多知识后才解决这个问题。这启示我们设计项目的过程启示就是学习的过程，只有不断去学习新的知识，才能获得解决问题的灵感。很多时候一个问题解决不了可能并不是因为问题本身有多难，而不过是这个问题需要的技术恰好没掌握罢了。
  - 对于PKUStudyHelper这样的多功能集成的项目，在开发的过程中注意功能的封装是尤其重要的，否则到后期函数功能很容易混淆，也很不利于调试和维护