# QT大作业智能课程表助手作业报告

63队 深藏blue队

小组成员：龚鹤宁 鲍彦泽 兰艺慧

## 一、程序功能介绍

### 1、课程表管理

**手动录入课程**

支持添加/编辑/删除课程信息

必填字段：课程名称、星期几、节次、教室

可选字段：教师姓名、课程颜色标记以及自定义字段，用来存储上课老师提到的一些关键信息。

**可视化展示**

周视图表格展示（周一至周日）

按节次分行的时间轴布局

自定义课程颜色区分不同类型课程

### 2、上课提醒

**基础提醒功能**

课前自定义分钟时系统托盘通知

显示课程名称、教室和节次信息

静音模式开关

**当前课程显示**

主界面高亮显示当前时间段课程

显示下一节课信息

### 3、作业管理

**作业记录**

添加作业（关联课程、作业名称、截止时间）

标记完成状态

按截止日期排序

**DDL 提示**

列表显示未完成作业

显示剩余天数（"今天"、"已过期 x 天"、"还剩 x 天"）

临近截止日期(3 天内)特殊颜色标注

考试时间显示（前一周高亮显示）

### 4、功能入口

工具栏按钮

添加课程

编辑课程

删除课程

查看作业

添加相关作业、DDL

自定义提醒时间

## 二、项目各模块与类设计细节

### 1. TaskManager (任务管理器)

- 用途：管理任务的生命周期，包括添加、删除、状态更新和持久化存储
- 关键功能：
  - addTask()/removeTask()：任务增删
  - setTaskCompleted()：标记完成状态
  - saveTasks()/loadTasks()：文件读写

### 2. Settings (设置管理)

- 用途：管理应用程序配置和用户偏好
- 关键功能：
  - 存储提醒时间(reminderMinutes)
  - 管理静音状态(isMuted)
  - 保存窗口布局(windowGeometry)

3. MainWindow（主窗口）
   - 用途：应用程序主界面，集成所有功能模块
   - 关键功能：
     - 课程表/任务列表显示
     - 系统托盘集成
     - 用户操作处理(添加/删除课程任务)
4. Task（任务模型）
   - 用途：表示单个任务的数据模型
   - 关键功能：
     - daysRemaining()：计算剩余时间
     - priorityColor()：动态优先级颜色
     - statusText()：生成状态描述
5. CourseDialog（课程对话框）
   - 用途：创建/编辑课程的交互界面
   - 关键功能：
     - 课程属性编辑(名称/时间/教室)
     - 颜色选择器
     - 输入验证
6. ScheduleManager（课程管理）
   - 用途：管理课程数据和时间计算
   - 关键功能：
     - getCurrentCourse()：获取当前课程
     - getNextCourse()：计算下一节课
     - 课程冲突检测
7. TrayIcon（系统托盘）
   - 用途：管理系统托盘图标和菜单
   - 关键功能：
     - 单例模式实现
     - 双击恢复主窗口
     - 气泡消息通知
8. Course（课程模型）

- 用途：表示单门课程的数据模型
- 关键功能：
  - hasTimeConflictWith()：冲突检测
  - displayText()：生成显示文本
  - 序列化支持

**9. Notification（通知系统）**

- 用途：处理课程和任务提醒
- 关键功能：
  - checkReminder()：课程提醒检查
  - showCourseReminder()：显示提醒
  - 静音控制

**10. TaskDialog（任务对话框）**

- 用途：创建/编辑任务的交互界面
- 关键功能：
  - 截止日期/时间选择
  - 关联课程选择
  - 考试类型标记

# 三、小组成员分工情况

龚鹤宁主要负责课程系统和统筹工作，关键贡献有冲突检测、当前课程计算，以及撰写报告；鲍彦泽主要负责任务系统和通知模块，关键贡献有优先级算法、提醒逻辑；兰艺慧主要负责 UI 和数据持久化，关键贡献有课程表渲染、文件存储和视频录制。

# 四、项目总结与反思

本项目开发了一款面向高校学生的智能课程表助手，核心功能包括课程管理、任务跟踪、智能提醒和系统集成。项目基于 Qt 框架开发，采用 C++11 标准，支持 Windows/macOS/Linux 多平台运行。三人团队历时 8 周完成，代码总量约 3500 行。

**1、核心技术实现：**

在课程管理模块，我们实现了冲突检测算法 Course::hasTimeConflictWith()，开发时间映射系统将实时时间映射到节次 ScheduleManager::getCurrentSection()，解决跨周课程计算难点 ScheduleManager::getNextCourse()中的双循环边界处理。

在任务系统模块，我们实现了动态优先级可视化，Task::priorityColor()根据紧急程度

动态生成颜色，智能状态描述 Task::statusText()自动生成"还剩 3 天/已过期"等提示，同时支持多类型区分作业/考试任务（m_isExam 标志位）。

在智能提醒模块，我们设置了上课前 N 分钟提醒：Notification::checkReminder()的分钟级精确计算，同时完成了跨周课程处理：通过(dayOfWeek - currentDay + 7) % 7 解决跨周边界问题。

在界面交互方面，我们实现了跨节次课程显示：QTableWidget::setSpan()实现单元格合并，整门课程选择：MainWindow::selectEntireCourseSpan()智能选中连续课程，同时数据自动保存：5 分钟定时器触发持久化（防止意外丢失）。

最后在代码健壮性上，我们进行输入验证：所有对话框实现 validateInput()防止非法数据，异常防护：文件操作全路径错误检查（QFile::open()状态验证），内存管理：显式删除任务对象（TaskManager::removeTask()中的 delete）。

**2、 团队协作经验：**

我们采用 Git 工作流，基于特性分支开发+PR 代码审查，同时明确定义模块间接口（如 ScheduleManager::getCoursesByDay()），在遇到关键问题时展开线下、现上讨论快速同步进度和阻塞问题。我们协作沟通的成果有：统一了数据格式，课程存储统一使用 quint32 保证跨平台兼容；开发 DateUtils 时间计算工具函数库；所有类方法使用 Doxygen 风格注释。这些规范使得我们的项目完成地更顺利。

**3、 反思与改进方向：**

**架构层面：**未实现 Model/View 解耦，UI 逻辑与数据管理耦合度高；通知系统依赖主窗口实例，限制模块复用性。

**功能层面：**缺少课程临时调整功能（如调课/代课）；未实现数据导入导出（Excel/课表图片识别）；无用户登录和云同步能力。

**工程实践：**单元测试覆盖率不足（仅核心算法有简单测试）；CI/CD 流水线未搭建（依赖手动编译打包）；缺乏性能分析（未评估百门课程时的效率）。

**团队协作：**前期接口定义不够清晰，导致集成阶段返工；未采用敏捷看板工具，任务进度可视化不足；技术方案讨论不充分（如序列化格式选择）。

**4、结语：**

总而言之，通过这次 QT 大作业的实践，我们为以后的科研和学习生活积累了大量经验，也完成了应该做什么、怎么做、怎么做好的初级探索。从最初的分工讨论，到中期的代码编写与整合，再到后期的测试与优化，每一个环节都离不开团队成员之间的密切配合与沟通。大家各司其职，充分发挥自己的优势，同时又相互支持、相互学习，共同攻克了一个又一个难题。感谢程序设计实习的教师和助教团队给予我们这个宝贵的机会，相信几年之后，回看这份报告，依然会觉得当年自己的努力和成果，值得骄傲。