作业报告

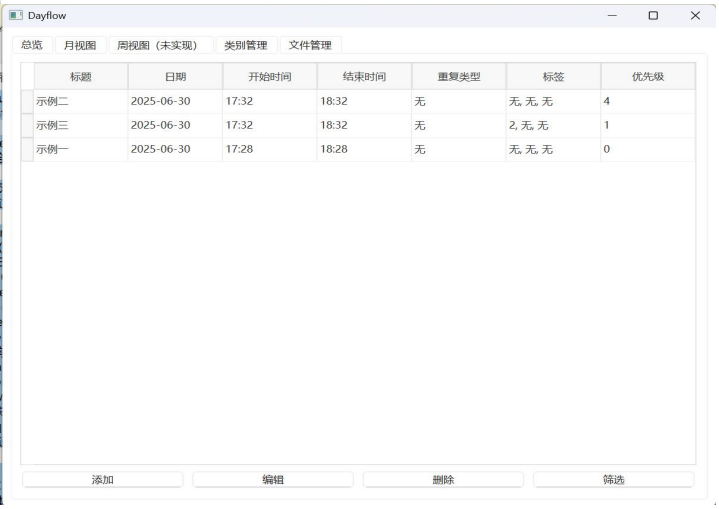
陈可 2400013167　刘玉龙 2300013162　肖博鑫 2300013121

一．功能介绍

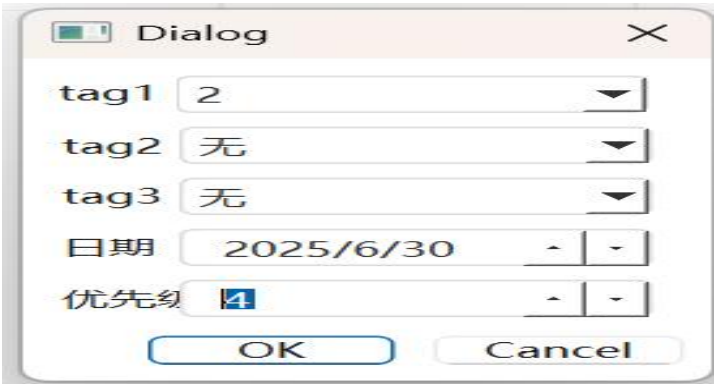
1.日程管理：

（1）日程编辑及筛选

用户可以在创建日程的时候通过程序提供的下拉选项框给创建的日程添加至多三个 tag 以及一个从 0 到 5 的优先级（数字越大，优先级越高），如果用户不修改的话，默认所有 tag 为无，优先级为 0，在日程总览界面所有日程将按照从高到低的顺序排列。

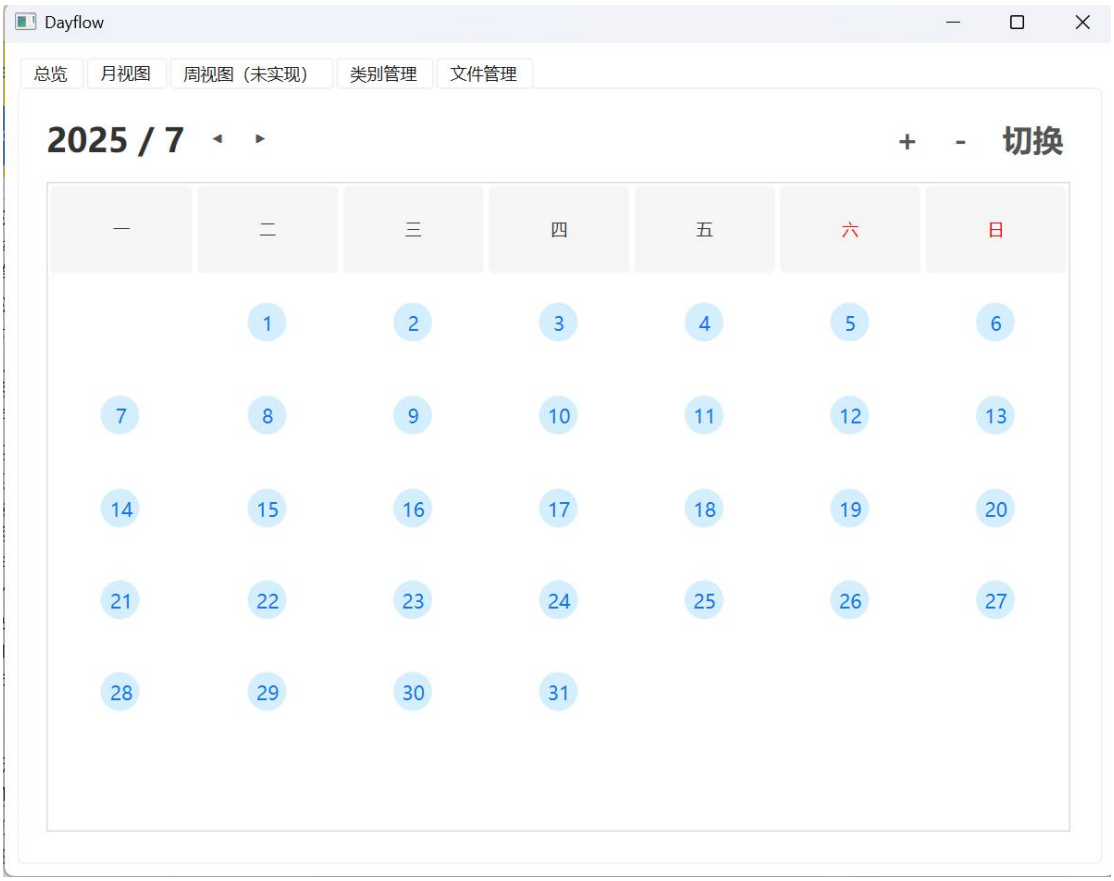


用户还可以通过点击总览界面的“筛选”按钮来筛选所有符合标准的日程。

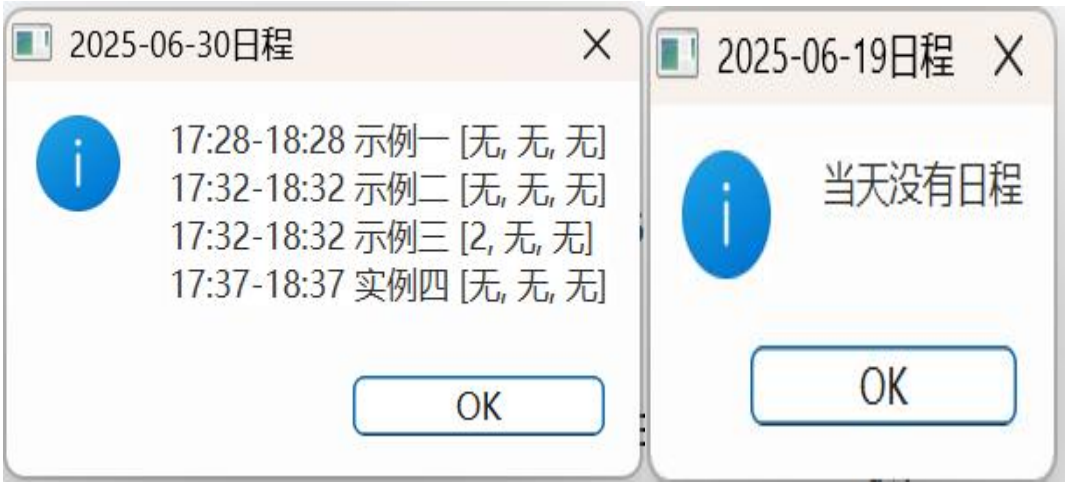


（2）视图切换

在月视图界面，用户可以通过点击左上角的“<”,“>”来实现切换月份。在月视图界面，只会显示当月的日期，工作日字体为黑色，双休字体为红色，当天日期的背景将会被设置为蓝色，字体设置为白色，如果某天有日程的话，这天的字体将会被一个淡蓝色“胶囊”包裹。



用户还可以通过点击某个具体的日期，获取某天所有日程。



2.分类管理：

（1）.创建、管理类别

在类别管理界面，用户可以新建、编辑多个类别，用于日程和文件分类管理。

创建界面：

管理界面：

（2）.日程分类管理

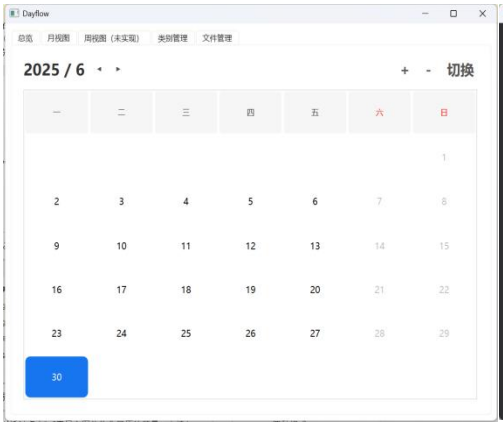
在进行日程编辑操作时，用户可以将其归类为任一已创建的类别。在类别管理页面可以查看日程的分类详情。

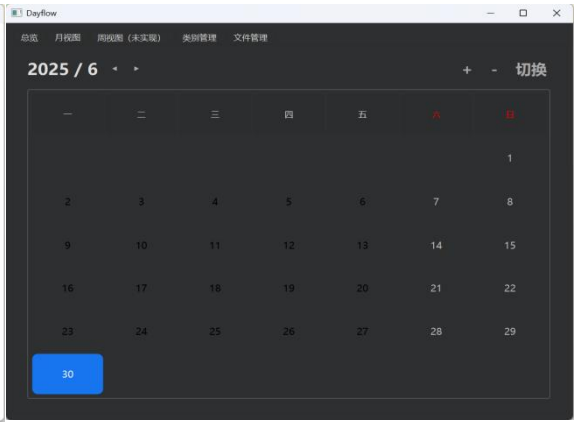
（3）.文件分类上传

创建的类别同样可以用于文件管理。用户可以上传文件到任一类别中，从而将文件存储与日程管理相联动。

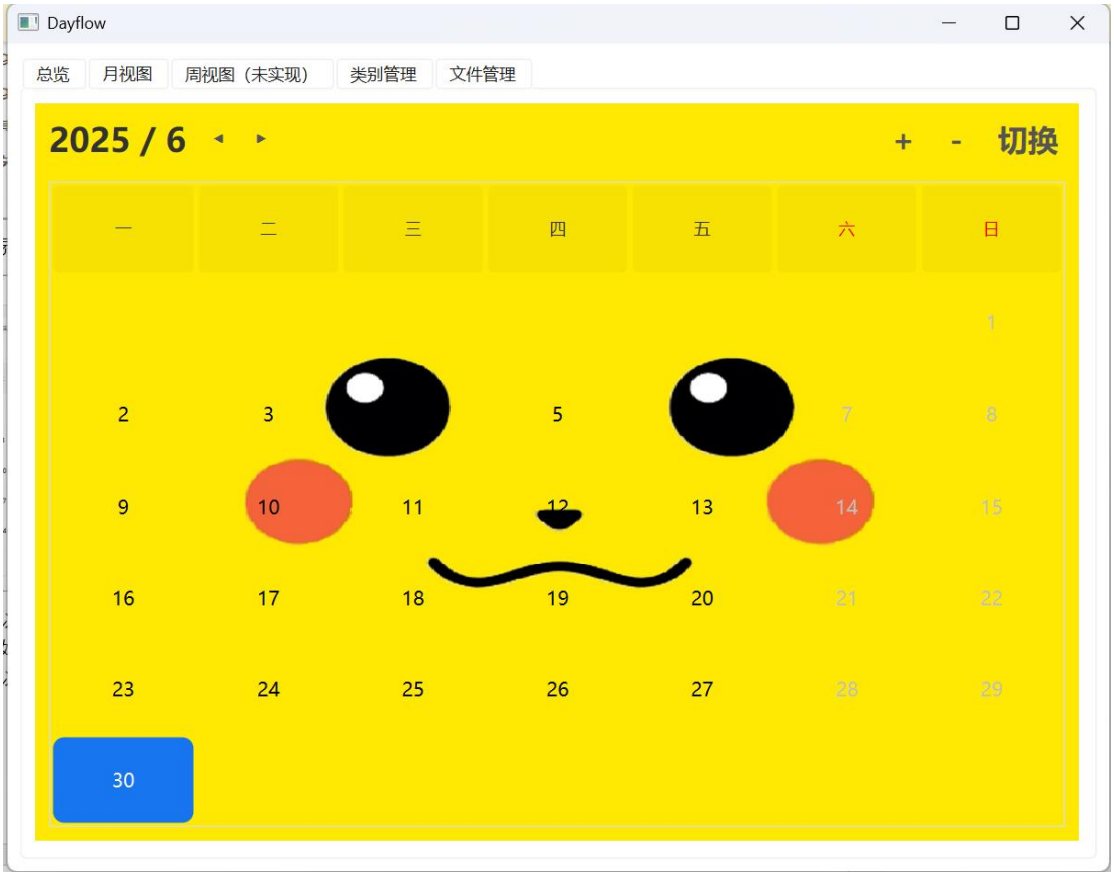
3. UI 风格切换

在月视图界面，用户可以通过点击右上角的“切换”来切换明，暗两种主题，默认情况下为明亮主题。



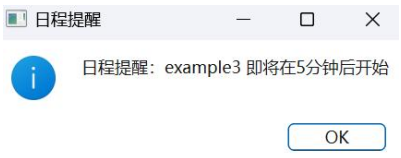


用户还可以通过点击“+”来导入图片作为日历的背景，支持 bmp，jpg，jpeg，png 四种格式的图片，如果想清除背景恢复默认的话，用户可以通过点击“-”来一键清除背景。



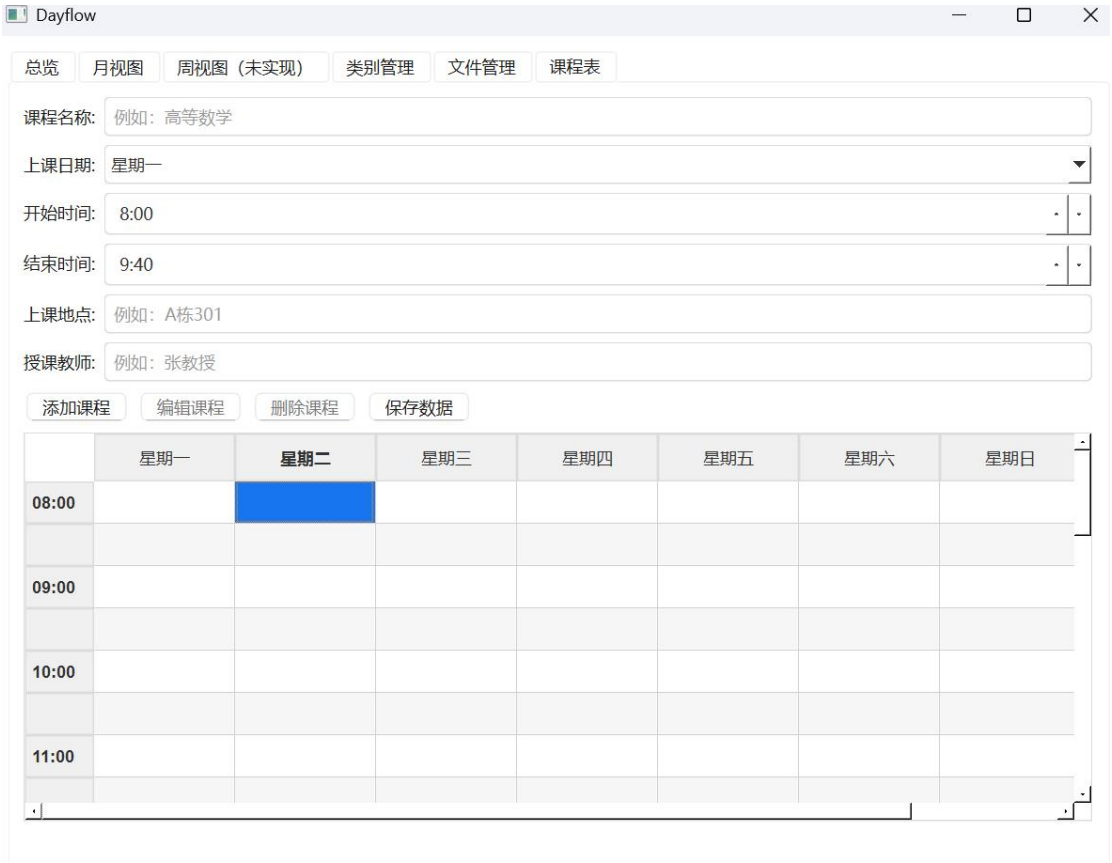
4. 通知系统

用户在创建日程时可以自定义通知时间。在预定时间会有弹窗提醒。



（4）.课程表功能

日历包含了一个相对独立的课程表，用户可以在课程表页面中编辑每项课程的名称、时间、地点、授课教师。该课程表实现了一周内从早上八点到晚上二十一点之间，每一个小时的课程时间的可视化表格。用户可以添加、修改课程信息，也可以删除某门课程的信息。每次添加、修改或删除表格信息后需要点击“保存数据”按钮进行保存。需要查看对应课程的信息，只需要双击表格中的相应课程，信息就会出现在上方的填入框中。



二．类设计细节

1，editdialog

a,在构造函数中添加 3 个 tag 的下拉选项框和优先级的调整框，并且为其设置初始值，从而可以在用户进行创建或者修改日程时提供相对应的图形界面

b,getScheduleItem()在这个函数中获取用户创建或修改的日程的信息，并将其填入要返回的 ScheduleItem 对象中，最后返回相对应的对象进行更新和存储

2，calendarview

a,在构造函数中，首先为整个日历界面设置对应的对象名，便于以后通过 QSS 样式表控制格式，之后申请一个竖直布局作为整个日历的底板并设置该底板内部件的间距，然后创建一个水平布局作为顶部导航栏，然后依次创建 month_label(月份标签)，“<”,“>”,空白区域，“+”，“-”，“切换”并为其设置对应的对象名，以便通过 QSS 样式表控制格式，然后将所有的部件加入之前的水平布局中作为顶部导航栏，之后用 Eventcalendar 创建日历的主体部分，并对其默认格式进行修改，来方便之后的美化。然后将导航栏和日历加入底板并将底板应用到整个窗口上。之后将信号与对应的槽函数相关联，并进行初始化，来实现月视图的显示

b,refresh()函数中调用 paintEventMarkers()来在日历上画出必要的和特殊的标记，调用完成后强迫日历重新绘制自身来实现日历的刷新

c,updateMonth(int year, int month) 用于日历的初始化和处理

QCalendarWidget::currentPageChanged 信号，获取更改后的月份和日期并为其设置格式，最后调用 paintEventMarkers()来重新绘制日历

d,paintEventMarkers()创建一个 QMap (dateEvents) 来存储日期和事件计数的映射,之后计算出当前日历视图的起始和结束日期，遍历模型中的每一个日程项，对每一个日程项，使用一个 while 循环来“展开”它的所有重复实例（每日、每周、每月），对于每一次展开得到的日期，如果它落在当前视图的月份范围内，就在 dateEvents map 中为其计数加一，循环结束后，将包含所有统计结果的 dateEvents map 传递给 EventCalendar 控件，让它根据这些数据来绘制事件标记，更新视觉表现。

e,handleDateClick(const QDate &date)处理 QCalendarWidget::clicked 信号的函数，接收到信号后，通过接口获取当日的所有日程，并决定输出的内容，之后创建一个 QMessageBox 并为其创建对象名和进行格式调整，最后设置其为模态弹窗输出应该输出的内容。

f,onSetBackgroundClicked()处理 addBtn:QPushButton::clicked 的函数，唤起文件选择界面，之后调用 setBackgroundImage(const QString& imagePath)设置背景，之后向 mainwindow 发送一个 backgroundImageChanged(imagePath)信号，通知其保存背景路径

g,setBackgroundImage(const QString& imagePath)设置背景的核心函数，检查是否可以用图片设置背景，可以的话通知日历进行重新绘制

h,onClearBackgroundClicked() 处理 clearBgBtn:QPushButton::clicked 信号的函数，向 onSetBackgroundClicked()发送一个空路径，并通知 mainwindow 清空保存的背景路径

i,paintEvent(QPaintEvent *event)在接受对应的信号后调整图片尺寸并将其填充为日历背景

3，eventcalendar

a,paintCell(QPainter *painter, const QRect &rect, const QDate date)用来执行具体的日期格的绘制，首先保存原初的 painter，之后对正在绘制的日期格进行条件检测，如果满足相对应的条件就进行特殊的绘制，最后形成整个日历的视觉表现

4，filterdialog

a，构造函数为筛选对话框设置应有的筛选条件的图形界面，为筛选搭建一个 UI

b，getCriteria()通过用户在筛选对话框中的输入，从而实例化一个 FilterCriteria（辅助筛选的结构体），并将该结构体返回以便 mainwindow 根据此结构体进行实际的筛选

5，mainwindow

a,构造函数，为 mainwindow 界面设置初始的主题，为代理模型设置源模型，并为信号们设置对应的槽函数，最后代理模型进行排序，来实现总览界面的按优先级排序。

b,setupTableView()不再使用源模型，改用代理模型以便进行日程的排序

c,onAdd()处理 addButton:QPushButton::clicked 信号，实例化一个 editdialog 对象，并将其返回的对象加入储存模型中。

d,onEdit()处理 edit:QPushButton::clicked 信号，实例化 editdialog 对象，将原有日程从模型中取出，并将新的对象加入模型中

e,onDelete()处理 delete:QPushButton::clicked 信号，将原有日程从模型中删除

f,onFilter()处理 filter:QPushButton::clicked 信号，通过返回的 FilterCriteria 对象从代理模型中筛选对应的日程，并更新显示

g,applyTheme(Theme theme)根据传入的参数决定该采用哪种主题，并从文件中加载相对应的 QSS 样式表并应用，最后记忆当前主题

h,onToggleTheme()处理 CalendarView::themeChangeRequested 的槽函数，接收到信号后确定

切换后的主题并执行切换，并记忆用户此次切换
i,setupTheme()在构造函数中调用的主题初始化函数，读取用户最后一次切换后的主题样式，并在 mainwindow 初始化时应用此主题样式
j,onBackgroundImageChanged(const QString& imagePath) 处理 backgroundImageChanged(imagePath)信号的槽函数，接收到信号后保存当下的背景路径
k,categoryExists(const QString &name)和 findCategoryByName(const QString &name),功能类似，分别用于按名称检查类别是否存在和寻找类别，作为实现类别编辑功能的子函数
l,loadJsonConfig() 、 loadCategoriesFromJson() 、 updateCategoryInJson() 、 removeCategoryFromJson()和 saveCategoryToJson()，用于从 json 文件中读取、存储、修改、删除类别信息，实现了类别的持久化存储。
m,setupFileManagementPage()，创建文件管理页，实现了文件上传功能：将用户选择的文件复制并存储到对应类别的目录之中。
n,saveFileMetadata()、removeFileMetadata()，将存储的文件信息于 json 文件中保存/删除，实现了文件的持久化存储
o,refreshFileManagementCategories()，refreshFiles()，在文件管理界面执行刷新操作时分别处理类别刷新和文件刷新，同时，refreshFiles()实现了打开文件和删除文件按钮的点击事件。
p,setupcategoryPage()，创建类别管理界面，实现了新建类别、编辑类别和日程管理三个子页面。在新建类别页面中，实现了新建类别操作，先创建类别信息存储路径，并将其保存到 json 文件中。在编辑类别页面中，实现了类别修改、删除功能。在进行类别修改时对新类别名称合法性进行检测，在此之后执行 updateCategoryInJson（）函数，将信息存储到 json 文件内。在进行类别删除时先删除其存储目录，在此之后执行 removeCategoryFromJson()函数，将新信息存储到 json 文件内。
在日程管理页面中，实现了按类别分类管理日程的功能。其中定义了函数 refreshScheduleView() 用于将类别和日程信息加载到界面中。同时处理 QTreeWidget::itemDoubleClicked 信号，在双击日程项后调用 showScheduleDetail()函数，显示日程详细信息。

6，schedulefilterproxymodel
a,filterAcceptsRow(int sourceRow, const QModelIndex &sourceParent)实际上对于每一行日程进行筛选匹配的的函数，其对于每一个日程根据筛选的 FilterCriteria 依次判断各项是否符合筛选要求，并据此决定相对应的行是否显示

7，light.qss,dark.qss 和 resources.qrc
resources.qrc 负责通知主程序明暗两个主题的 QSS 样式表储存在哪，而 light.qss 和 dark.qss 则是 QSS 样式表用来存储明暗主题的格式设置

8. Notification
用于编辑提醒弹窗页面及弹窗信号

9. Schedulereminder
a. ScheduleReminder(),计时器，每分钟触发对是否进行提醒的监测
b. checkSchedules()，对所有日程依次检测是否提醒，包括对时间的检测和对重复逻辑的检测

10. Scheduleitem

a. ScheduleItem()，日程对象，存储单个日程的基本信息
b. isValid()，在编辑日程时用于检查时间设置是否合法
c. toJson()及 fromJson()，将日程对象与 json 格式相互转化，用于持久化存储

11. ScheduleModel

a. data(),将日程对象可视化于总览界面
b. headerData()，在总览界面顶部显示名称，用于 ui 构造
c. addItem()、removeItem()、getItem()，分别进行日程的创建（编辑）、删除和展示
d. getSchedulesForDate()，处理日程的重复逻辑
e. saveToJson()、loadFromJson()、getStoragePath()，从 json 文件中读取\存储日程信息，用于日程的持久化存储

12. category

类别管理中的类别对象

a. toJson()和 fromJson()，将类别信息与 json 格式相互转化，用于持久化存储
b. FileInfo，存储文件信息的对象，包含文件名称、路径、大小、上传时间和所在类别

13.CourseSchedule

课程表的实现

a.setupUI:实现了课程表整体 UI 的构建，包括课程表的中心部件（课程表页面本身）、输入部件（需要填入的信息）、按钮布局及课程表格布局；同时，该成员函数还实现了信号槽的连接等功能。

b.setupTable:setupUI 中调用的成员函数，负责课程表格的详细构建，包含初始化表格的属性和调用 createTimeSlots 等函数，

c.add\edit\delete_course 等系列函数:实现了课程信息的添加、修改和删除和信息的本地储存

d. cellDoubleClicked:实现双击单元格在填写框中填充对应信息
e. saveCourses:实现了输入信息保存到 JSON 文件
f. loadCourses：从 JSON 文件中加载课程信息

三. 分工情况

1. 总览页面：包括日程总览、日程创建、编辑、删除：陈可、刘玉龙
2. 日程标签、优先级设置及筛选；月视图；UI 风格设置：刘玉龙
3. 类别管理：分类管理日程、文件分类上传及管理：陈可
4. 课程表功能：课程信息编辑及展示：肖博鑫

四. 反思与总结

这次的 C++大作业，我们选择开发一个名为《Dayflow》的个人日程管理应用。我们的目标是构建一个不仅功能完整，而且界面设计也向现代桌面应用看齐的程序，而不只停留在完成课业任务的层面。项目基于 C++/Qt 框架开发。我们通过继承 QAbstractTableModel 创建了 ScheduleModel 作为核心数据模型，负责封装所有日程数据的底层操作，这让我对数据与视图分离的设计思想有了深刻的实践体会。为了实现灵活的筛选和排序，我们还引入了 QSortFilterProxyModel，在不影响源数据的情况下对视图进行动态处理。在视图层面，除了

标准的 QTableView，我们还实现了一个自定义的 CalendarView。为了达到让界面简洁美观，我们又重构了 paintEvent 事件，手动绘制了日期单元格的背景、事件标记和文字颜色，这个过程让我真正理解了 Qt 的图形绘制体系。此外，我还利用 QSS 样式表和 Qt 资源系统，为应用构建了一套可一键切换的明暗双模式主题，并通过 QSettings 实现了存储。开发过程也并非一帆风顺。在处理用户自定义背景图片时，也遇到了 QSS 样式覆盖的难题，最终通过改为在 paintEvent 中直接绘制背景的方案才得以解决。在实现类别管理系统时，由于对代码量的错误估计，导致了大量的代码堆砌于 mainwindow.cpp 文件内，不利于之后的修改和维护操作。总而言之，这个项目让我将 C++的理论知识与 Qt 的实践应用紧密结合，虽然目前程序仍然有许多缺点，但它已经是我们尽力所为。这个大作业不仅锻炼了我的编程能力，还提高了我分析问题、调试程序的综合能力。