Qt 大作业报告——DdlHub

石杨子然 孙天宇 邓俊逸

一、程序功能介绍

1. 身份管理：

- 提供登录/注册功能、支持记住密码和账号安全验证

2. 学习管理系统：

- 可视化的周课表展示、支持课程增删查改与冲突检测

3. 任务管理系统：

- DDL 管理（作业/考试/论文）与智能分类（今日任务/7 天任务）
- 周期性任务支持
- 完成状态实时追踪

4. 效率工具：

- 番茄钟计时器：
- 可配置的工作/休息计时、可视化环形进度条、自动阶段切换

5. 成就系统：

- 图标化成就展示、完成状态动态切换、解锁条件提示

6. 交互界面：

- 自定义主题控件：
- 汉堡菜单按钮、滑动式侧边栏、九宫格导航按钮、带动画效果的进度条和弹窗

二、项目各模块与类设计细节

前端类

AchievementWidget

1. 功能职责

可视化展示成就系统信息，支持显示成就图标、标题和副标题并根据成就完成状态动态切换显示效果（完成/未完成）。支持响应鼠标点击事件以弹窗展示成就的解锁条件说明。

2. 核心方法

构造函数：默认构造与带参构造

设置函数：

void setIcon(const QPixmap&)：更新成就图标，触发重绘。

void setTitle(const QString&)：设置主标题，自动重绘控件。

void setSubtitle(const QString&)：设置副标题，自动重绘控件。

void setCompleted(bool)：更新完成状态，切换背景色/图标色彩，触发重绘。

void setConditionText(const QString&)：设置成就解锁条件文本（不触发重绘）。

重写事件处理：

void mousePressEvent：左键点击时弹窗显示 conditionText，空文本显示默认提示。

void paintEvent(QPaintEvent*)：圆角背景/状态色/图标处理/文字分层渲染。

QSize minimumSizeHint() const：返回建议最小尺寸(300x100)，防止布局压缩。

ChangePasswordDialog

1. 功能职责

提供一个对话框界面用于修改用户的用户名和密码，显示并允许编辑当前用户的用户名和密码，并将修改后的用户数据通过 DataManager 持久化存储。

1. 核心方法

ChangePasswordDialog：构造函数，初始化界面并绑定用户数据

QString getUsername() const：获取当前输入框中的用户名

QString getPassword() const：获取当前输入框中的密码

void handleSave()：处理保存操作，更新用户数据并持久化存储

CircularProgressBar

1. 功能职责

用于番茄钟应用的时间管理可视化，直观展示计时状态（工作、休息或暂停），可以显示剩余时间、切换工作/休息模式、支持暂停/继续操作，并在阶段完成时触发回调通知。

2. 核心方法

构造函数：初始化 UI 尺寸、创建内部计时器并建立信号槽连接

setValue/setMaximum：分别设置当前进度值和阶段最大时长（秒级控制）

setUser：绑定用户对象用于阶段完成时的回调通知

setDurations：配置工作/休息时长（自动转换为秒存储）

start/pause/reset：控制计时状态（启动计时器/进入暂停/完全重置进度）

onTimeout：处理每秒的进度更新，自动切换阶段模式并在完成时回调

paintEvent：动态绘制环形进度条和剩余时间文本，根据状态使用不同配色方案

CustomMenuButton

1. 功能职责：触发全局导航菜单，显示三条水平线作为视觉表现，可悬停、点击交互。

2. 核心方法

构造函数:

CustomMenuButton(QWidget *parent=nullptr)：初始化按钮基础属性，包括固定尺寸为32x32、设置手型光标以及应用透明无边框样式。

重写函数 paintEvent：绘制菜单图标，使用抗锯齿渲染。

DurationDialog

1. 功能职责

基于 QDialog 的自定义对话框，用于设置工作和休息的时间。用户可以通过 QSpinBox 输入工作时长和休息时长，并通过 "OK" 和 "Cancel" 按钮确认或取消。

2. 核心方法：

DurationDialog(QWidget *parent = nullptr)：构造函数，初始化对话框 UI，设置默认值

int workMinutes() const：获取用户设置的工作时长（分钟）

int breakMinutes() const：获取用户设置的休息时长（分钟）

ScheduleButton

1. 功能职责

显示带有 3×3 白色方格（九宫格样式）的交互按钮。作为"日程表"、"菜单展开"或"网格视图"等功能的入口，支持点击、悬停，并通过自定义绘制在按钮中央添加视觉标识。

2. 核心方法

构造函数 explicit ScheduleButton：初始化按钮并设置固定大小为 40×40 像素

重写 paintEvent：先绘制按钮默认样式，再在中央用抗锯齿渲染绘制白色方形网格。

SlideMenu

1. 功能职责

可滑动的侧边菜单控件，主要功能包括：支持显示/隐藏的平滑滑动动画效果，允许通过布局添加任意子控件，以及采用半透明黑色背景样式。

2. 核心方法

构造函数：SlideMenu(QWidget *parent = nullptr)：初始化菜单尺寸、样式和动画参数。

toggle()：切换菜单显示状态。

showMenu()和 hideMenu()：显示和隐藏菜单并播放滑动动画。

addWidget(QWidget *widget)方法：向菜单布局中添加自定义控件。重写的
resizeEvent(QResizeEvent *event)方法：在窗口大小变化时调整菜单尺寸。

SlideMenuButton

1. 功能职责

滑动菜单按钮，实现圆角背景、悬停颜色变化以及手型光标等，提升用户交互体验。

2. 核心方法

构造函数：SlideMenuButton 无参构造与创建带有指定文本按钮的带参构造

事件处理函数：

paintEvent(QPaintEvent *event)：处理按钮的自定义绘制，包括圆角和悬停效果

enterEvent(QEnterEvent *event)：鼠标进入时激活悬停状态

leaveEvent(QEvent *event)：鼠标离开时取消悬停状态

CourseDialog

1. 功能职责：获取用户的课程信息
2. 核心方法

   构造函数：CourseDialog(QWidget *parent = nullptr)初始化添加课程对话框界面

   QString getCourseName() const：获取课程名称

   int getDay() const：获取课程星期（返回 0-6，代表周一到周日）

   int getStartSection() const：获取开始节数

   int getEndSection() const：获取结束节数

   QString getLocation() const：获取地点

CourseSchedule

1. 功能职责：呈现课程表管理界面，支持用户查看、添加和删除课程。
2. 核心方法

   构造函数 CourseSchedule：初始化课程表窗口，接收数据管理器和用户对象引用。

   void createActions()：配置添加和删除课程的交互动作。

   void populateTable()：用当前课程数据填充课程表界面。

   bool hasTimeConflict：检测新课程与已有课程的时间冲突。

   void loadUserCourses()：从数据管理器加载当前用户的课程数据。

   addCourse()：处理添加课程按钮点击，弹出添加界面。

   showCourseDetails(int row, int column)：点击表格项时显示课程详情。

   showContextMenu(const QPoint &pos)：右键点击表格项时弹出删除菜单。

   deleteCourse()：执行选中课程的删除操作。

TopMenu

1. 功能职责：展示用户头像、修改个人信息、跳转到课程表界面、打开侧边菜单
2. 核心方法

   构造函数 TopMenu：初始化界面布局，创建并添加控件

   成员变量初始化：

   layout（QHBoxLayout）：管理水平布局

   profileButton（QPushButton）：显示用户头像并触发个人信息修改

   schedulBtn（QPushButton）：导航至课程表界面

   customBtn（QPushButton）：控制侧边菜单的显示/隐藏中

AddDDLDialog

1. 功能职责

   提供对话框界面，用于添加或编辑 DDL 项，支持设置任务名称、截止时间、类型和周期性

2. 核心方法

   构造函数 AddDDLDialog(QWidget* parent = nullptr)：初始化对话框 UI

   DDLItem getItem() const：获取用户输入的 DDL 信息

   void setItem(const DDLItem& item)：设置对话框内容（用于编辑已有 DDL）

   static QString encodeCycle(const QList<QCheckBox*>& boxes)：将复选框状态编码为周期字符串

   static QString displayCycle(const QString& cycle)：将周期字符串显示为可读文本

DDLWidget

1. 功能职责

   单个 DDL 项表格的显示及交互（支持编辑、删除操作）

2. 核心方法

   构造函数: DDLWidget：初始化 UI 组件

   数据获取: DDLItem getItem() const：返回当前 DDL 项数据

   数据更新: void setItem(const DDLItem& newItem)：刷新 DDL 数据并更新显示

DeadlineWidget

1. 功能职责

   主管理界面，显示 DDL 列表、提供筛选和搜索功能，处理 DDL 的增删查改操作

2. 核心方法

   构造函数：DeadlineWidget(User& user, QWidget* parent = nullptr)：初始化界面

   内置处理函数：

      void refreshDDLList(QVector<DDLItem> list)：刷新 DDL 列表显示

      QVector<DDLItem> sortDDLs const：对 DDL 按截止时间排序

   槽函数：

      void showAddDDLDialog()：显示添加 DDL 对话框

      void onDDLUpdated(const DDLItem& newItem)：处理 DDL 更新事件

      void onDDLDeleted(DDLWidget* widget)：处理 DDL 删除事件

      void onDDLEditRequested(DDLWidget* widget)：处理 DDL 编辑事件

      void filterDDL(const QString& type)：按类型筛选 DDL

void searchDDL(const QString& keyword)：按关键词搜索 DDL

TodayTaskMenu

1. 功能职责

   显示和管理用户今日任务（周期型任务）和长期任务（7 日内截止的非周期任务）

   滚动显示，可根据完成状态实时更新 DDL 界面

2. 核心方法

   TodayTaskWindow：构造函数，初始化 UI 并导入 DDL 数据。

   void loadTasks()：加载任务数据，筛选今日任务和即将截止的任务并更新 UI 显示。

   void setupUI()：配置 UI 组件布局及样式。

   void addTaskWidget：将单个任务添加到任务列表控件中。

   void onTaskCompleted：处理任务完成事件，更新已完成任务状态并刷新任务列表。

RegisterDialog

1. 功能职责：支持用户输入用户名、密码和确认密码进行注册功能
2. 核心方法

   构造函数：RegisterDialog(QWidget* parent = nullptr)：初始化 UI 界面

   数据获取：

   QString getUsername() const：返回当前输入的用户名

   QString getPassword() const：返回当前输入的密码

   槽函数：void onRegisterClicked()：触发注册验证逻辑，若输入有效则提示成功

LoginDialog

1. 功能职责

   用户登录对话框，用户键入用户名和密码以登录，支持记住账号功能

2. 核心方法

   LoginDialog(QWidget* parent = nullptr)：初始化登录界面，并自动调用

   loadSettings()：加载保存的登录信息

   QString getUsername() const：获取用户名文本

   QString getPassword() const`：获取密码文本

   void loadSettings()：加载已保存的用户名、密码和记住状态，并自动填充到界面

   void saveSettings()：根据"记住账号"勾选状态决定是否保存/清除用户名、密码信息

   信号 void registerRequested()：当用户点击注册按钮时触发

后端类

Course

1. 功能职责：表示用户的课程信息
2. 成员变量：

   name：课程名称（QString）

   starttime/endtime：课程开始/结束时间（QTime）

   date：上课日期（星期几，QString）

   location：上课地点（QString）
3. 核心方法：

   构造函数：默认构造和带参初始化

   operator==：重载相等运算符，通过名称、时间、日期判断课程是否相同

   时间冲突检测：在 addcourse()中自动检测新课程是否与现有课程时间重叠

Task

1. 功能职责：管理用户的任务/DDL
2. 成员变量：

   title：任务名称（QString）

   deadline：截止时间（QDateTime）

   estimatedTimeMinutes：预计完成时间（分钟）

   isCompleted/failed：任务状态标志

   type：任务类型枚举（Assignment/Paper/Exam）

   cycle：任务周期设置（QList<int>）

   daystocomplete：预期所需天数

   priority：优先级（1-10 级）
3. 核心方法：

   三种不同参数的构造函数：适应不同初始化需求

   operator==：重载相等运算符，判断任务是否相同

   hoursRemaining()：动态计算剩余时间（小时）

   reducedaystocomplete()：减少预期天数并自动标记完成状态

Achievement

1. 功能职责：管理用户成就系统
2. 成员变量：

title：成就名称（QString）

unlocktime：解锁时间（QDateTime）

description：成就描述（QString）

3. 核心方法：

多种构造函数：简略构造（仅标题）和完整构造

operator==：判断成就唯一性

User

1. 功能职责：整合用户所有数据和行为
2. 成员变量：

username/password/lastlogindate：用户名，密码，上次登录时间

courses/tasks/achievements：三个 vector 存储用户的课程，任务，成就

tomatoclocktime/taskcompletetime：用户的番茄钟使用次数与任务完成次数

3. 核心方法：

verify()：验证用户身份

课程管理：

addcourse()：添加课程（带时间冲突检测）

三种移除方式：按对象/名称/时间

任务系统：

addtask()：添加任务（带重复检测），同时判断与任务数量有关成就的完成

removetask()：移除任务

completetask()：完成任务，自动进行相关成就获取

checktask()：自动检测过期任务

getShownTasks()：筛选后返回需要显示给用户的任务列表。

成就系统：

addAchievement()：添加成就（带重复检测）

tomatoclockfinished()：番茄钟统计与成就触发

is0To4()：判断任务完成时间是否符合成就“夜猫子”的要求

checkAchievement()：监测成就完成情况

DataManager

1. 功能职责：持久化存储管理

2. 核心方法：

saveUser()：结构化保存用户数据，存储课程/任务/成就/统计量，实现数据格式化

loadUser()：按[COURSES]等分区标识模块化读取数据，类型转换处理，错误处理

LoginSystem

1. 功能职责：用户认证管理
2. 核心方法：

userExists()：通过文件存在性检测用户

registerUser()：创建带基础信息的用户文件，检测用户名冲突

login()：文件读取用户身份

三、小组成员分工情况

石杨子然（前端）	AchievementWidget ChangePasswordDialog CircularProgressBar DurationDialog CustomMenuButton ScheduleButton SlideMenu SlideMenuButton
邓俊逸（前端）	CourseSchedule DailyMisson AddDDLDialog DDLWidget DeadlineWidget LoginDialog RegisterDialog TodayTaskMenu TopMenu
孙天宇（后端）：	Course Task Achievement User DataManager LoginSystem

四、项目总结与反思

1. 开发过程中我们遇到了一些较为典型的困难：

-前端与后端接口不熟，模块之间协作效率低：起初在前后端的数据传递上存在理解偏差，导致部分

功能失效。通过统一接口文档、明确数据结构定义、加强沟通与协作，逐步打通了各个模块的逻辑流转。

-对象与引用混淆，数据未能正确更新：在多个模块传递用户信息或任务列表时，初期常用对象传值，导致数据修改未能同步回数据源。经过调试后意识到必须使用引用或指针来实现真正的“共享数据”，之后统一了接口传递方式，大大减少了数据不同步的问题。

-界面复杂度增加带来的维护困难：随着功能增多，窗口和控件逐渐复杂，导致布局混乱、维护成本提升。我们后来进行了模块封装、逻辑抽离，把核心组件提炼为独立类，提升了代码复用性和可维护性。

-时间管理不足导致进度滞后：项目初期未能精准估算各模块的开发时间，导致后期某些模块（如成就系统和番茄钟）开发时间被压缩。通过阶段性评估进度、合理调整任务分工，我们最终仍保证了项目的完整性。

2. 项目收获与未来优化方向

通过本次大作业我们不仅加深了对 Qt 框架的理解，更锻炼了团队协作与项目管理能力。

-熟练掌握了 Qt 的核心机制，包括信号槽、布局管理、自定义控件、资源管理等;

-学会了模块化设计与接口协作，提高了程序的拓展性和工程化水平;

-提高了调试能力与问题定位能力，能灵活应对各种运行时 bug 与逻辑冲突;

-在项目推进中也培养了团队间的沟通与协作意识。

当然，我们也认识到项目仍有不少可提升空间:

-UI 界面尚可优化，目前功能完整但视觉设计较为基础，可进一步美化布局、统一风格;

-数据缺乏持久化支持，当前数据存储只存在内存中，后续可接入数据库实现长期保存;

-成就系统可进一步拓展，例如增加图标奖励、连续完成奖励等机制，提高游戏化体验;

-任务分析模块可引入可视化，如每日完成情况折线图、番茄钟使用时长统计等图表，提升反馈性与沉浸感。