程设 QT 大作业

组号：19　　组名：你说对不队　　指导老师：刘家瑛老师

程序功能介绍

这份大作业的核心目的是帮助大学生进行日程以及任务管理的工作。因此，功能主要涵盖了以下几个方面：

1. 选择计划：通过一些提前设计并内置的计划选项供用户参考或直接使用，并通过打卡的方式对已经完成的任务进行记录
2. 多模式计时器：内置多模式的计时器可帮助用户更沉浸更专注的学习，并建立良好的时间管理观念内置模式包括了耳熟能详的番茄钟以及长短休息等，同时也允许用户自定义时长。
3. 制定计划：用户可通过填写表格来制定其专属的学习计划，功能与内置计划完全一致。
4. 历史记录：历史记录页面支持查看所有曾经进行过的任务以及未完成的任务，都会有标识来区分是否完成，且按任务进行的时间排序以提升查看历史记录的便利性。每个记录还记录了用户参与同一样任务的次数和时长并且支持点击后恢复到上次未完成任务的页面，提升效率。
5. 时间表：支持用户导入个人的上课时间表或日程表，支持更换时间表颜色、为每个日程添加备注添加地点以及支持日程来临 10 分钟前进行提醒。
6. 数据统计：对用户的学习历史记录等各方面的数据进行统计，并用各种类型的数据图将数据分析的结果可视化，如用户累计打卡时长、用户每月在特定任务的打卡时长、连续打卡天数等等。

7. 个人首页：支持用户自定义昵称和头像等功能，以社交媒体常见的打卡小火焰来显示连续打卡天数并以每月日历展示当月打卡情况。

项目各模块与类设计细节

1. 学习计划打卡页面：

这段代码定义了一个 overlay 类，它是一个自定义的 QWidget，用于学习计划打卡页面。以下是其功能和设计细节的总结：

设计细节：

初始化和 UI 设置：

构造函数使用父对象初始化小部件，并通过 Ui::englishoverlay（由 Qt UI 设计器生成）设置 UI。

“返回按钮”连接到一个 lambda 函数，当点击时会弹出一个确认对话框，询问是否退出页面。

```
#ifndef ENGLISHOVERLAY_H
#define ENGLISHOVERLAY_H

#include <QWidget>

namespace Ui {
class englishoverlay;
}

class englishoverlay : public QWidget
{
    Q_OBJECT

public:
    explicit englishoverlay(QWidget *parent = nullptr);
    ~englishoverlay();

    void setTasks(const QStringList &tasks);
    void setName(const QString &name);
    void setTasksWithState(const QList<QPair<QString, bool>> &taskList);


private:
    Ui::englishoverlay *ui;
};

#endif // ENGLISHOVERLAY_H
```

返回按钮行为：

确认（选择“Yes”）后，代码会将当前计划的任务及其状态保存到文件（unfinished_plans.txt，位于 Documents 目录中）。

任务以文本和复选框状态（0 为未选中，1 为选中）保存，根据所有任务是否勾选标记为 [FINISHED] 或 [UNFINISHED]。

同名计划会被覆盖，窗口随后隐藏。

任务管理：

setTasks(const QStringList &tasks)：清空任务列表，并使用 TaskItemWidget 实

例填充新任务，设置固定项高度。

setName(const QString &name)：更新 UI 中显示的计划名称。

setTasksWithState(const QList<QPair<QString, bool>> &taskList)：类似于 setTasks，但还会根据提供的 QPair 设置每个任务的复选框状态。

2，制定计划：

```
private:
    void setupUI();
    void updateDisplay();
    void playAlarm();

    QTimer *countdownTimer;
    int remainingSeconds;
    int initialSeconds;
    int timeSpent;                  // Total time spent in seconds
    qint64 startTimestamp;          // Timestamp when timer was last started

    QLabel *timeDisplay;
    QComboBox *modeSelector;
    QLineEdit *customInput;
    QPushButton *startButton;
    QPushButton *pauseButton;
    QPushButton *resetButton;

    QMediaPlayer *player;
    QAudioOutput *audioOutput;

    bool isRunning;
};

#endif // TIMERWIDGET_H
```

这段代码定义了一个 TimerWidget 类，继承自 QWidget，实现了一个计时器界面，支持多种时间模式和自定义时长。

设计细节：

1. 初始化和基本设置：

构造函数初始化计时器（QTimer）、媒体播放器（QMediaPlayer 和 QAudioOutput）以及相关变量，设置默认状态为未运行。

计时器每秒更新一次，媒体播放器用于播放警报音，音量默认设为 50%，警报文件路径为特定本地文件。

```
#ifndef TIMERWIDGET_H
#define TIMERWIDGET_H

#include <QWidget>
#include <QTimer>
#include <QMediaPlayer>
#include <QAudioOutput>

class QLabel;
class QPushButton;
class QComboBox;
class QLineEdit;

class TimerWidget : public QWidget {
    Q_OBJECT

public:
    TimerWidget(QWidget *parent = nullptr);

signals:
    void timerFinished(int duration); // Emitted when timer reaches zero, duration is time spent
    void timerPaused(int duration);   // Emitted when timer is paused, duration is time spent

private slots:
    void updateCountdown();
    void startTimer();
    void pauseTimer();
    void resetTimer();
    void modeChanged(int index);
```

2. UI 布局和组件：

setupUI 创建垂直布局，包含时间显示标签（timeDisplay）、模式选择下拉框（modeSelector）、自定义输入框（customInput）和三个按钮（开始、暂停/继续、重置）。

时间显示采用居中、大字体样式；按钮具有自定义样式（黑色背景、悬停和点击效果）。

模式包括番茄钟（25 分钟）、短休息（5 分钟）、长休息（15 分钟）和自定义时长。

3. 模式切换和时间设置：

modeChanged 根据选定模式设置初始时间（initialSeconds），自定义模式启用输入框，其他模式禁用输入框并使用预设时间。

自定义输入框支持实时更新时间（分钟转秒），非运行状态下立即反映。

4. 计时器控制：

startTimer：开始计时，初始化剩余时间（remainingSeconds），记录开始时间戳。

pauseTimer：暂停或继续计时，暂停时计算并记录已用时间（timeSpent），并发出 timerPaused 信号。

resetTimer：重置计时器，恢复初始时间并停止计时。

updateCountdown：每秒减少剩余时间，计时结束时停止计时、发出 timerFinished 信号并播放警报。

5. 警报和显示更新：

playAlarm 播放警报音并弹出 AlarmDialog，警报在 60 秒后自动停止，或通过对话框关闭。

updateDisplay 将剩余时间格式化为“MM:SS”显示。

这段代码实现了 taskRecord 类的功能，用于记录任务、计划及时间消耗，并

维护统计数据。以下是其功能和设计细节的总结：

3.历史记录

设计细节：

1. 初始化：

构造函数设置日志文件路径为 task_log.txt，位于文档目录（使用 QStandardPaths），并输出路径调试信息。

```
#ifndef TASKRECORD_H
#define TASKRECORD_H

#include <QObject>
#include <QFile>
#include <QTextStream>
#include <QDateTime>

class taskRecord : public QObject
{
    Q_OBJECT
public:
    explicit taskRecord(QObject *parent = nullptr);
    void logTask(const QString &planName);
    void logPlanWithTasks(const QString &planName, const QStringList &tasks);
    void logTimeSpent(const QString &planName, int duration);
    void updateSummaryStats();
    void refreshStats();
    QPair<QString, int> getCurrentStats(); // 返回 <今日时长字符串，连续天数>

signals:
    void statsUpdated(const QString &todayHours, int streak);


private:
    QString logFilePath = "task_record.txt"; // 日志文件路径
    QString m_todayHoursText;
    int m_currentStreak;

};

#endif // TASKRECORD_H
```

2. 统计更新： updateSummaryStats()：读取日志文件，解析每日时长（dailyDurations），计算今日总时长（秒转小时）和连续完成天数（streak）。更新文件首行统计信息（格式：#TOTALS|日期|时长|连续天数），并发射 statsUpdated 信号，更新内部变量 m_todayHoursText 和 m_currentStreak。
3. 任务记录：

logTask(const QString &planName)：记录计划名称，包含日期和时间信息，更新已有记录的出现次数（第 10 字段），初始时长为 0。记录后调用 updateSummaryStats()。

logPlanWithTasks(const QString &planName, const QStringList &tasks)：记录计划及其任务列表，格式同上，任务以“- 任务名 (Pending)|日期”形式追加。记录后更新统计。

logTimeSpent(const QString &planName, int duration)：更新指定计划的时长

（秒），若无记录则创建新条目，累加已有时长。记录后更新统计。

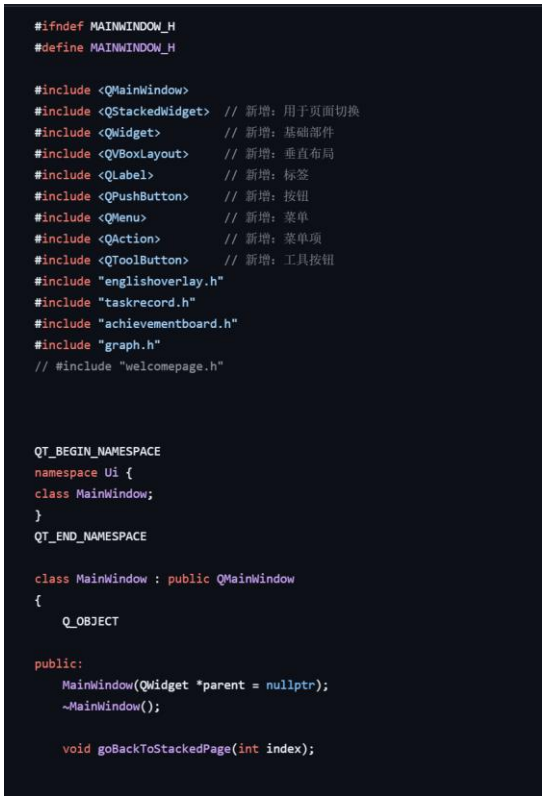
4. 统计刷新和获取：

refreshStats()：读取日志文件首行（统计信息），解析今日时长和连续天数，更新内部变量并发射信号。若无统计数据，默认设为 0。

getCurrentStats()：返回当前统计数据（m_todayHoursText, m_currentStreak）作为 QPair。

4.功能统合

这段代码实现了 MainWindow 类，定义了一个主窗口界面，集成了任务管理、计时器、统计图表和历史记录等功能。



```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QStackedWidget>  // 新增：用于页面切换
#include <QWidget>         // 新增：基础部件
#include <QVBoxLayout>     // 新增：垂直布局
#include <QLabel>          // 新增：标签
#include <QPushButton>     // 新增：按钮
#include <QMenu>           // 新增：菜单
#include <QAction>         // 新增：菜单项
#include <QToolButton>     // 新增：工具按钮
#include "englishoverlay.h"
#include "taskrecord.h"
#include "achievementboard.h"
#include "graph.h"
// #include "welcomepage.h"



QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    void goBackToStackedPage(int index);
```

设计细节：

1. 初始化和 UI 设置：

构造函数设置 UI（通过 Ui::MainWindow），初始化 taskRecord 实例用于日志记录，配置按钮连接和图像显示。

底部导航按钮切换 stackedWidget 页面（首页、时间表、任务统计、个人资料），部分按钮打开新窗口（如 Timetable 和 Profile）。

“退出”按钮关闭应用，工具按钮显示计划选择菜单（学习英语、编程、剪辑），并加载对应任务和计时器。

2. 计划管理和计时：

选择计划（如“学习英语”）时，记录任务，设置 englishoverlay 显示任务列表，居中显示 TimerWidget，并连接计时器信号（完成/暂停）记录时间。

“制定计划”按钮打开 FormDialog，用户输入计划和任务后，加载到 englishoverlay 并启动计时器。

“历史记录”按钮打开 HistoryWindow，选择历史计划后加载到 englishoverlay 并启用计时。

```
private:
    Ui::MainWindow *ui;

private slots:
    void onStatsUpdated(const QString &todayHours, int streak);

protected:
    void showEvent(QShowEvent* event) override;
    void resizeEvent(QResizeEvent* event) override;
    void showOverlay(QWidget* overlay);
    void loadUnfinishedPlans();
    void loadSelectedPlan(const QString &planName);
    void generateQuote();
    void loadTodayTasksToChart();

private:
    englishoverlay *englishOverlay;
    QWidget* overlayWidget;
    taskRecord* TaskRecord;
    taskRecord taskLogger;
    achievementboard *achievementBoard;
    graph *Graph;

    taskRecord *m_taskRecord;
    QLabel *m_hoursLabel;      // 指向UI中显示小时数的标签
    QLabel *m_streakLabel;     // 指向UI中显示连续天数的标签

};
#endif // MAINWINDOW_H
```

3. 数据统计和图表：

loadTodayTasksToChart() 读取 task_log.txt，统计今日任务时长，生成饼图（QPieSeries）显示任务分布，样式包括颜色和标签设置。

onStatsUpdated() 更新界面上的今日时长和连续打卡天数标签，响应 taskRecord 的 statsUpdated 信号。

4. 其他功能：

generateQuote() 随机显示励志语录。

loadUnfinishedPlans() 读取 unfinished_plans.txt，显示未完成计划供用户选择。

showOverlay() 和事件处理（showEvent、resizeEvent）动态调整 englishoverlay 位置和大小。

小组成员分工情况

组长：吴洁允 2400094807

组长负责了与助教沟通，同时为组员分配任务。其中，在大作业进行期间，组长主要负责前端 ui 大部分的设计工作以及各个组员间代码的整合工作，并主要的负责了演示视频的剪辑工作。路演期间也参与了部分的 ppt 设计以及报告演示工作。

组员：陈雯宜 2400094814

雯宜主要负责了大作业的部分功能的后端代码实现以及部分前端的 ui 设计工作，后端代码方面主要参与了时间表、数据统计以及个人首页部分，前端设计主要参与了时间表和个人首页部分。她也参与了演示视频的讲解以及路演上台报告的工作。

组员：刘诗政

诗政主要负责大作业的功能实现的后端代码编写部分，主要完成了多模式计时器的代码与 ui、制定计划的代码、历史记录的代码与 ui 部分，并参与了代码整合的部分。他也参与了演示视频的讲解、路演的 ppt 设计、路演的上台报告并主要完成了大作业报告的工作。

项目总结与反思

本项目旨在为大学生提供一个全面、实用的日程与任务管理工具，以提高学习效率和时间管理能力。项目设计围绕“计划制定—任务执行—数据反馈”这一完整闭环展开，通过所介绍的几样核心功能使得本项目具备了一定的实用性和完整性。尽管项目整体功能较为完备，用户体验初步达成预期，但在开发与测试过程中也暴露出一些不足之处：

1. 界面交互细节仍需优化：部分功能页面（如计划填写、数据统计界面）在初次使用时缺乏足够的引导或提示，容易造成理解困难，影响用户上手效率。
2. 功能集成度略显分散：目前各模块相对独立，功能之间的联动性不强，例如日程与打卡记录之间缺乏自动关联，影响整体使用的连贯性。
3. 缺乏智能推荐机制：系统尚未根据用户历史数据自动推荐学习计划或时间管理策略，仍需用户手动选择，智能化程度有待提升。
4. 提醒功能不够灵活：日程提醒时间固定为 10 分钟前，尚不支持用户自定义提醒时长，限制了实用场景的多样性。
5. 数据统计维度有限：虽然已初步实现打卡时长与频率的可视化，但若能引入更多如学习效率评估、任务完成率对比等统计维度，分析功能将更具深度。