个性化心情管理助手项目报告

组名：代码都队 **组员**：郁凯文，耿子喻，禹慧俊

一、项目功能介绍

本小组开发的“个性化心情管理助手”核心目的在于帮助用户有效的管理、记录与调节自己的心情状态，以维持用户的心理健康，具体涵盖七大功能模块：

1. **心情日记**：支持通过 emoji 表情与文字相结合的方式，实时记录打卡每日心情状态，并可随时回溯历史日记内容，以进行编辑修改，从而为用户打造专属情绪记录空间。
2. **心理测试**：提供快速与标准两种问卷模式，用户可自主选择进行心理状态自测，系统将生成专业分析结果与改善建议。
3. **虚拟 AI 聊天**：深度接入 DeepSeek，为用户提供针对性建议与情绪价值支持，实现智能情感交互。
4. **沉浸式冥想与休息**：内置轻柔背景音乐与白噪音，配合呼吸训练及冥想引导，用户可灵活调节倒计时，享受沉浸式放松体验。
5. **情绪可视化**：基于心情日记数据，自动统计 7 天、30 天情绪变化趋势，以直观图表与数据呈现情绪波动规律。
6. **成就系统**：设置多元成就目标，通过达成成就赋予用户成就感，激发主动改善情绪的内在动力。
7. **首页界面设计**：界面设计简洁美观，设有“每日一句”栏目，通过励志哲理语录有助于激励用户。同时又通过引导语设计，有助于用户快速了解程序功能，快速上手。

二、项目各模块与类设计细节

1. 心情日记模块

（1）总体界面设计说明

在该模块中（MoodDiary），设计有两个界面：一为心情日记的显示与预览界面（calendarPage），二为心情日记的编辑与修改界面（editPage）。

在 calendarPage 界面中, 设计了一个能显示 emoji 的日历类(EmojiCalendarWidget), 以直观显示用户心情状态。（后文会详细介绍）

同时，我们亦设计了预览模块，当用户点击日历上的日期时，其相应的心情日记数据会显示在预览模块（dairybox）中。其中包含：

- dateLabel（QLabel 类）：显示选中日期
- moodLabel（QLabel 类）：显示当日心情 emoji
- timeRecordList（QListWidget 类）：显示当日不同时段的心情记录
- diaryBrowser（QTextBrowser 类）：显示当日的日记简要（只显示一行）
- editButton（QPushButton 类）：文字显示“查看/编辑当前心情日记”，当点击此

Button 时会切换界面到 editPage（编辑）页面。

在 editPage 界面中，用户可以选择心情，填写/编辑文字，以创建自己独特的心情笔记。具体包含：

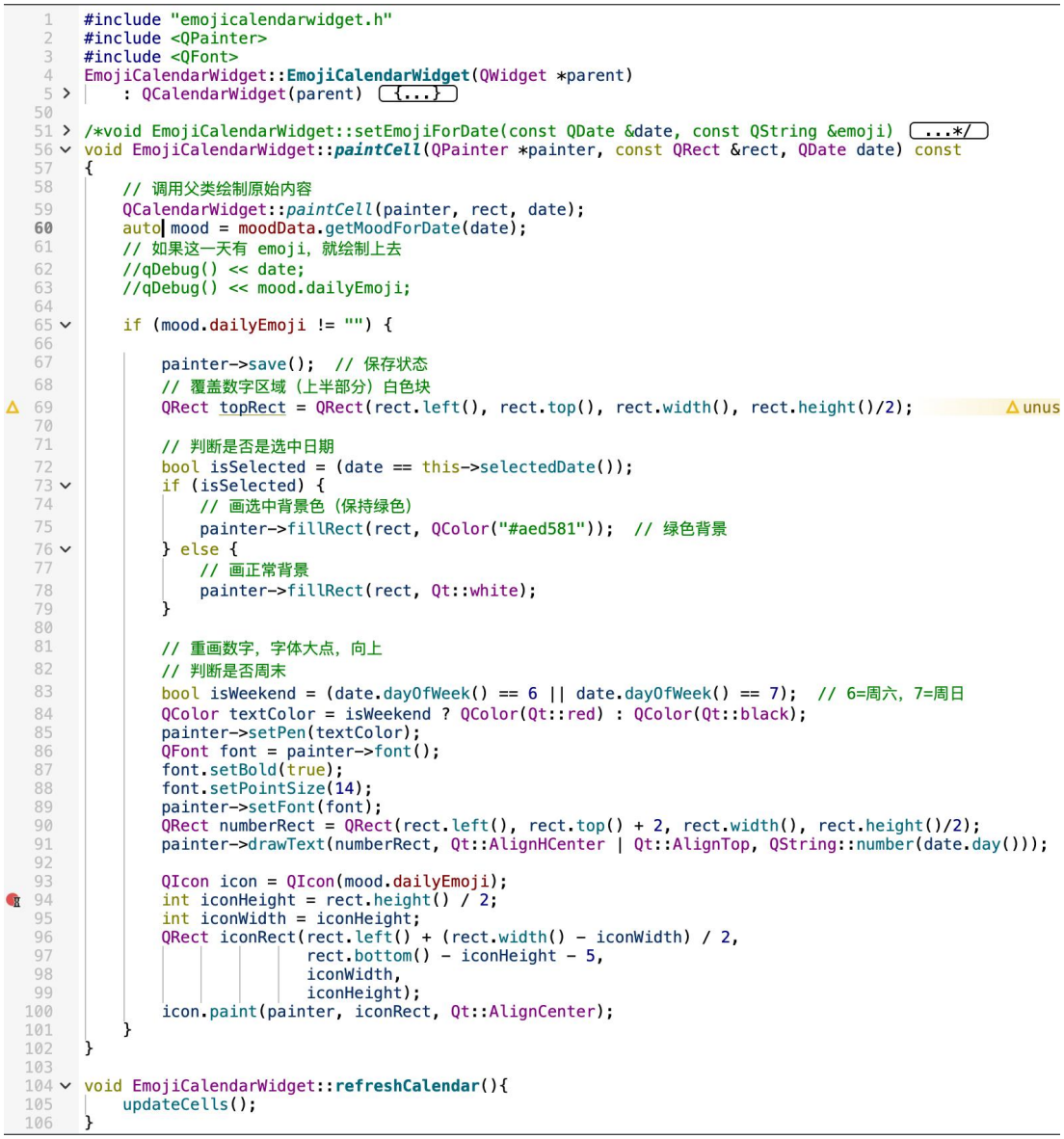
- dateLabel_2 (QLabel 类)：显示选中日期
- dailyEmojiBox（QComboBox 类)：为当天总体心情的选择框。用户点开下拉箭头会显示可选的心情选项，共有 13 项（分别为"开心","庆祝","机智","酷帅","好吃","惊讶","平静","伤心","生气","尴尬","可怜","疲倦","思考"，存储在 EmojiList 当中），用户可以自由选择自己的心情状态，默认为空白选项（无心情）
- morningEmojiBox/afternoonEmojiBox/eveningEmojiBox(QComboBox 类)：分别代表早上、下午、晚上的用户心情选择框，功能样式与 dailyEmojiBox 一致。
- morningEdit/afternoonEdit/eveningEdit（QTextEdit 类）：分别代表早上、下午、晚上的用户心情随笔，用户可以输入文字以修改编辑。（只有一行，为随笔简记）。
- diaryEdit（QTextEdit 类）：为用户当日的日记，用户可以输入文字以修改编辑。（为多行的长日记，记录一整天的总体感受感想）
- saveButton（QPushButton 类）：文字显示“保存”，点击时会保存心情日记，将心情日记写进 mood_diary.json 文件（后文会具体说明），并且会自动跳出一个提示窗口（QMessageBox），显示“你的心情记录已保存”。
- returnButton（QPushButton 类）：文字显示“返回”，点击后返回 calendarPage 页面。

同时，两个界面中还有大量显示具体引导文字的 Label，这里不加赘述。并且项目界面设计均为在 Qt Creator 内置的 Ui 设计模块中实现，故以上的界面组件与其样式表并无法在源代码中体现。

（2）含有 emoji 的日历类设计（EmojiCalendarWidget 类）

EmojiCalendarWidget 类为可以显示 emoji 的日历类，为 Qt 中的 QCalendarWidget 类派生而来。在构造函数中，我们为其设计了简洁美观的独特样式表（独特的背景、悬停和点击效果等）。

同时，我们主要重载了其中的 paintCell 函数。我们重新处理了带有 emoji 时的显示情况、鼠标选中的显示情况、周末的显示情况，以最终呈现出简洁美观的日历形式。具体设计代码如下图所示。



```cpp
#include "emojicalendarwidget.h"
#include <QPainter>
#include <QFont>
EmojiCalendarWidget::EmojiCalendarWidget(QWidget *parent)
    : QCalendarWidget(parent) {...}

/*void EmojiCalendarWidget::setEmojiForDate(const QDate &date, const QString &emoji) ...*/
void EmojiCalendarWidget::paintCell(QPainter *painter, const QRect &rect, QDate date) const
{
    // 调用父类绘制原始内容
    QCalendarWidget::paintCell(painter, rect, date);
    auto mood = moodData.getMoodForDate(date);
    // 如果这一天有 emoji，就绘制上去
    //qDebug() << date;
    //qDebug() << mood.dailyEmoji;

    if (mood.dailyEmoji != "") {

        painter->save();  // 保存状态
        // 覆盖数字区域（上半部分）白色块
        QRect topRect = QRect(rect.left(), rect.top(), rect.width(), rect.height()/2);

        // 判断是否是选中日期
        bool isSelected = (date == this->selectedDate());
        if (isSelected) {
            // 画选中背景色（保持绿色）
            painter->fillRect(rect, QColor("#aed581"));  // 绿色背景
        } else {
            // 画正常背景
            painter->fillRect(rect, Qt::white);
        }

        // 重画数字，字体大点，向上
        // 判断是否周末
        bool isWeekend = (date.dayOfWeek() == 6 || date.dayOfWeek() == 7);  // 6=周六，7=周日
        QColor textColor = isWeekend ? QColor(Qt::red) : QColor(Qt::black);
        painter->setPen(textColor);
        QFont font = painter->font();
        font.setBold(true);
        font.setPointSize(14);
        painter->setFont(font);
        QRect numberRect = QRect(rect.left(), rect.top() + 2, rect.width(), rect.height()/2);
        painter->drawText(numberRect, Qt::AlignHCenter | Qt::AlignTop, QString::number(date.day()));

        QIcon icon = QIcon(mood.dailyEmoji);
        int iconHeight = rect.height() / 2;
        int iconWidth = iconHeight;
        QRect iconRect(rect.left() + (rect.width() - iconWidth) / 2,
                       rect.bottom() - iconHeight - 5,
                       iconWidth,
                       iconHeight);
        icon.paint(painter, iconRect, Qt::AlignCenter);
    }
}

void EmojiCalendarWidget::refreshCalendar(){
    updateCells();
}
```

图 1: EmojiCalendarWidget 设计

(3) 心情日记数据记录模式

项目中为心情日记数据设计了独特的 moodData 结构。其存储结构大致如下：

- moodMap（QMap 类）一个 date 对应一个 DailyMood 结构
- DailyMood 包含三部分, dailyEmoji, dailyNote (当天心情与当天笔记) , timeSlots（记录每个时段的心情情况）
- timeSlots （QMap 类）一个时段（"morning", "afternoon" 和 "evening"）对应一个 TimeSlotMood
- TimeSlotMood 包含两部分，该时段的心情 emoji 和该时段的随笔笔记 note

同时, 我们设计了一些具体函数来便捷操作数据, 包括: load 函数 (加载 MoodData 到 moodMap），save 函数（保存 MoodData 到 json 文件），getMoodForDate（获

取某天的 Mood），setMoodForDate（设置某天的 Mood）。下图直观显示了 MoodData 类的结构与相关成员函数情况。

```
struct TimeSlotMood {
    QString emoji;
    QString note;
};

struct DailyMood {
    QMap<QString, TimeSlotMood> timeSlots; // "morning", "afternoon" 和 "evening'
    QString dailyNote;
    QString dailyEmoji;
};

class MoodData {
public:
    MoodData();

    bool load(const QString &filename);
    bool save(const QString &filename);

    // 获取某一天的MoodDiary
    DailyMood getMoodForDate(const QDate &date) const;
    //设置某一天的MoodDiary
    void setMoodForDate(const QDate &date, const DailyMood &mood);

//private:
    QMap<QString, DailyMood> moodMap; // key: date.toString("yyyy-MM-dd")
};
```

图 2　MoodData 的结构与相关成员函数

最后，我们用一个 json 文件（mood_diary.json）来存储心情日记的数据。下图显示了某天心情日记的存储形式

```
},
"2025.05.18": {
    "dailyEmoji": ":/emoji/伤心.jpeg",
    "dailyNote": "翻着我们的照片\n想念若隐若现～\n去年    的冬天\n我们笑得很甜～",
    "timeSlots": {
        "afternoon": {
            "emoji": ":/emoji/伤心.jpeg",
            "note": "想念若隐若现～"
        },
        "evening": {
            "emoji": "",
            "note": ""
        },
        "morning": {
            "emoji": ":/emoji/可怜.jpeg",
            "note": "翻着我们的照片"
        }
    }
},
```

图 3 某天心情日记记录

（2）其他相关函数说明

- addMoodRecord：用以增添预览框中 timeRecordList 中的心情记录（早上、下午、晚上）
- updateMoodPreview：每次选中日历上的日期会调用此函数，更新预览模块（dairyBox）上的日期与心情日记显示。（其中会调用 addMoodRecord 函数）
- updateCalendar：用表情来更新日历。当日历中某天的心情状态被填写，日历中那天

日期上会添加对应的表情 emoji。

- set_default：设置选择框的默认选项/编辑栏的默认文字
- show_editPage：在进入 editPage 后会调用此函数，以添加选择框的所有选项，同时，在其中会调用 set_default 函数，来设置选择框与编辑栏的默认状态。
- judge_achievement：用以判断成就。（在成就系统中详细介绍）

所有 MoodDiary 的相关变量与成员函数如下图所示。

```
class MoodDiary : public QWidget
{
    Q_OBJECT

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

private slots:
    //void onDateSelected(const QDate &date);
    //void onEditButtonClicked();

    void on_editButton_clicked();                                    ⚠Slots named on_foo_bar are error prone [clazy-c
    void on_calendarWidget_clicked(const QDate &date);               ⚠Slots named on_foo_bar are error prone [clazy-c
    void on_returnButton_clicked();                                  ⚠Slots named on_foo_bar are error prone [clazy-c
    void on_saveButton_clicked();                                    ⚠Slots named on_foo_bar are error prone [clazy-c

private:
    Ui::MoodDiary *ui;
    QDate selectedDate = QDate::currentDate();
    MoodData moodData;
    QList<QString> emojiList = {"开心","庆祝","机智","酷帅","好吃","惊讶","平静","伤心","生气","尴尬","可怜","疲倦","思考"};

    void updateMoodPreview();
    void updateCalendar();
    void show_editPage();
    void addMoodRecord(QListWidget *listWidget, const QString &timeLabelText, const QString &iconPath, const QString &noteText);
    void set_default(QComboBox* box, QString text);
    void set_default(QTextEdit* edit, QString text);
    void judge_achievement(const QDate &date);
};
```

图 4 MoodDiary 的相关变量与成员函数

2. 心理测试模块

(1) 总体设计说明

该模块 (MoodTest) 中, 共设计了三个界面, 分别为选择问卷界面 (selectionPage), 作答界面（questionPage），结果界面（resultPage）

在选择问卷界面中，设有选择问卷类型的选择框（questionaireComboBox），用户可以选择“心理健康全面评估”和“情绪快速自测”两个选项。同时，在选择框下有一个描述问卷的描述框 (questionaireDescLabel)，以向用户解释其选中问卷的情况（目的，大约所需时间）。在描述框下有一个开始测试按钮（startButton），按下后进入作答界面（questionPage）。

在作答界面中，我们设计了 Question 和 Questionaire 类和一些函数以便捷显示、记录问题题干、选项与具体的答题情况（后文会详细介绍）。同时，我们设计了进度条（progressBar）以显示答题进度。界面下方有“下一题”按钮与“完成测试”按钮以更新答题进度与情况。完成问卷，点击“完成测试”，进入结果界面（resultPage）。

在结果界面中，会显示测试得分，百分比，结果解释与相关建议。这些均记录在 Questionaire 里面，有相应的计算标准（这里不多赘述）。界面下方有“返回测试选择”按钮，按下后返回选择问卷界面。

(2) Question，Questionaire 类及其他函数设计说明

在 Question 类中，text 代表题干，options 为选项文本与分值的 map。

Questionaire 类中，有 name（名字），description（描述），questions（含有问题列表），isQuick（是否为快速问卷），resultInterpretations（分数区间与解释的对应 map）。

开始前，调用 loadquestionaire 函数，从而得到用户选择 questionaire 的相关问题与得分数据。（这些数据直接保存在代码中）

然后，当用户进入问卷/转下一题时，会调用 setupQuestionPage 函数，其会清空现有选项，更新进度显示，同时，其也会根据问卷中的问题设定选项与选项按钮（QRadioButton 类）。每次点击选项按钮后，会更新该道题得分（selectedScore），并最后累加。

答完卷后，会调用 showResult 函数，从而计算得分与结果，并显示在结果界面上。如下图所示。

```
struct Question {
    QString text;
    QMap<QString, int> options; // 选项文本 -> 分值
};

struct Questionnaire {
    QString name;
    QString description;
    bool isQuick; // 是否为快速测试
    QVector<Question> questions;
    QMap<int, QString> resultInterpretations; // 分数区间 -> 解释
};
```

图 5 Question 与 Questionaire 类

```
private:
    Ui::MoodTest *ui;
    QMap<QString, Questionnaire> questionnaires;
    QString currentQuestionnaire;
    int currentQuestionIndex;
    int totalScore;
    //QTimer *progressTimer;
    int progressValue;

    void loadQuestionnaires();
    void setupQuestionnaireSelection();
    void setupQuestionPage(const Question &question);
    void showResult(int score);
};
```

图 6 相关变量与成员函数

3. 心语 AI 聊天系统

该模块中（aichat），我们设计了一个简介美观的界面，显示虚拟助手的图标，用户图标，聊天框与文字框。

我们接入了 deepseek，通过将配置存储在 config.ini 文件，从而调用 api，以实时和 deepseek 聊天，从而创建出一个善解人意的心理 AI 助手。

具体代码原理有点冗长，且没有特别值得一提的设计，故这里不加以详细说明。

4. 沉浸式冥想与休息系统

（1）总体设计说明

该模块中（Rest），用户可以在 musicTypeComboBox_3 中选择白噪音/音乐类型（共 4 种，分别为海浪、雨声、风铃、轻柔音乐）。并且在 countDownSlider 和

breathSpeedSlider（QSlider 类）中可以以拉动进度条的方式设置倒计时时间与呼吸速度。随着倒计时时间流逝，countDownSlider 进度条也会移动。而通过改变呼吸速率，可以改变圆圈动画的缩放速度，从而达到用户跟随圆圈动画呼吸，冥想休息的目的。

我们设置了三个按钮供用户控制音乐/白噪音播放情况，分别是播放按钮（playButton），暂停按钮（pauseButton），停止按钮（stopButton）。按下不同的按钮，可以播放/暂停/停止并重置音乐的播放情况。

同时，用户可以在 bedtimeModeCheckBox（QCheckBox 类）中选择是否为睡前模式。在睡前模式中，倒计时将自动设置为 15 分钟，音乐 15 分钟后会自动停止。

（2）BreathAnimation 类

BreathAnimation 类为我们设计的，用以播放圆圈缩放动画的，派生自 QWidget 的派生类。

动画实现的核心在于重载 paintEvent 类。在 paintEvent 中，我们画出圆形，并且圆形的半径 radius 会随着时间改变。

具体类结构如下图所示。

```
// 呼吸动画组件
class BreathAnimation : public QWidget
{
    Q_OBJECT
    Q_PROPERTY(qreal radius READ radius WRITE setRadius)     △ Q_PROPERTY
    Q_PROPERTY(QColor circleColor READ circleColor WRITE setCircleColor)

public:
    BreathAnimation(QWidget *parent = nullptr);
    qreal radius() const;
    QColor circleColor() const;
    void setRadius(qreal radius);
    void setCircleColor(const QColor &color);

protected:
    void paintEvent(QPaintEvent *event) override;

private:
    qreal m_radius = 50;
    QColor m_circleColor = Qt::lightGray;
};
```

图 7 BreathAnimation 类

5. 情绪可视化统计系统

（1）总体设计说明

该模块中（moodChart），我们根据心情日记中的心情记录。我们将每个心情 emoji 赋予一个分数（最高 5 分代表心情最好，最低 1 分代表心情最差，默认心情为 3 分），并且进行了以下的归总与统计：

- 本周心情指数：近 7 天内每天总体心情（dailyEmoji）分数的平均值。

- 本周出现最多的心情：近 7 天的总体心情与分时段心情中出现最多的心情 emoji。
- 本周写日记次数：近 7 天内 dailyNote 不为空的次数（即写了日记）
- 本月心情指数、本月出现最多的心情、本月写日记次数：统计近 30 天的数据，其余同理。
- 近 30 天表情使用情况：以柱状图的形式显示表情 emoji 的出现频率，从高往低排。
- 近 7 天心情变化：将近 7 天内每天总体心情分数画成折线图
- 近 30 天心情变化：将近 30 天内每天总体心情分数画成折线图

（2）具体函数说明

getMoodValue 函数将某个具体的表情（以表情图片地址的形式存储）转换为分数，具体的转换规则如下图。

```
int MoodChart::getMoodValue(const QString &emojiPath) {
    if (emojiPath.contains("开心") || emojiPath.contains("庆祝")) return 5;
    if (emojiPath.contains("酷帅") || emojiPath.contains("机智") || emojiPath.contains("好吃"))   return 4;
    //if (emojiPath.contains("思考") || emojiPath.contains("平静")) return 3;
    if (emojiPath.contains("伤心") || emojiPath.contains("可怜")) return 2;
    if (emojiPath.contains("生气")) return 1;
    if (emojiPath.trimmed().isEmpty()) return 3; // 无表情计为3
    return 3; // 默认值
}
```

图 8 表情转分数

在 drawWeeklyEmojiUsage 中，通过遍历前七天日期所对应的表情情况，最终统计出本周心情指数，本周出现最多的心情与本周写日记次数。

在 drawMonthlyEmojiUsage 函数中，同理统计出了本月心情指数、本月出现最多的心情与本月写日记次数，同时在此函数中，作出了以 x 轴为表情名称，y 轴为使用次数的近 30 天表情使用情况的柱状图，并加以显示。

在 drawWeeklyChart 和 drawMonthlyChart 函数中，实现了近 7 天心情变化情况与近 30 天心情变化情况折线图表的绘制。

所编写的变量和成员函数如下图所示。

```
Ui::MoodChart *ui;
MoodData moodData;

double weeklyMoodNum;
double monthlyMoodNum;
QString weeklyMood;
int weeklyNum = 0;
QString monthlyMood;
int monthlyNum = 0;
void drawWeeklyChart();
void drawMonthlyChart();
int getMoodValue(const QString &emojiPath); // 根据图片路径转换心情值
void drawWeeklyEmojiUsage();
void drawMonthlyEmojiUsage();
```

图 9 相关变量与成员函数

6. 成就系统

该模块中（achievement），我们根据已有成就状态的存储记录，在界面上显示成就以及成就所对应的完成状态（完成显示绿勾）。我们总共设计了 10 个

在成就系统中，编写了特定的成就类（AchievementItem），在成就类中设计了三个函数（setIcon，setText，setAchieved），以达到快捷设置并显示成就的图标，名称与完成情况。最终通过 addAchItem 函数将成就一一添加到界面里。成就类的设计结构如下图所示。

```
class AchievementItem : public QWidget
{
    Q_OBJECT

public:
    explicit AchievementItem(const QString& path, const QString& text, bool yes = false, QWidget* parent = nullptr);
    ~AchievementItem();
    void setIcon(const QString& path);
    void setText(const QString& text);
    void setAchieved(bool yes);
```

图 10 成就类

成就达成情况的数据存储在两个 Json 文件中（achievement_data.json 与 achievement_status.json）。在 achievement_data.json 文件中，记录一些达成成就所需要的关键变量情况，在 achievement_status.json 文件中，记录着所有成就的完成情况（true/false）。

成就系统的实现核心便在于操作、改写这两个文件的数据以达到记录成就的目的。操作这些成就的代码分别位于以下这些函数中：

- 日记达人与日记达人（进阶）：mooddiary 相关文件中的 judge_achievement 函数
- 心灵分析师与心灵初探秘：moodtest 相关文件中的 showResult 函数
- 心语初相识与心灵树洞王：aichat 相关文件中的 handleResponse 函数
- 图表解读者与周报收藏家：MoodChart 类的构造函数
- 冥想初体验：rest 相关文件中的 startBreathAnimation 函数
- 白噪音收藏夹：rest 相关文件中的 on_playButton_clicked 函数

7. 功能统合

项目的每一个功能都做成了独立的一个页面，通过使用一个页面容器（stackWidget）来容纳所有的页面，并设计了导航栏（navigationBar），用户点击导航栏上对应图标可进入相应功能界面。

同时，我们还设计了一个简洁美观的首页界面，不仅将项目所有功能通过引导栏统合在首页界面上，同时还设有每日一句栏目，内置 18 句哲理励志美句，随机呈现在首页界面上，从而保证了程序的整体性与美观性。下图展示了项目的功能统合结构。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

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

private:
    Ui::MainWindow *ui;
    QStackedWidget* stackWidget;//页面容器
    HomePage* homePage;//主页面
    MoodDiary* diaryPage;//心情日记界面
    MoodChart* chartPage;//统计界面
    MoodTest* testPage;//心理测试界面
    AIChat* chatPage;//AI聊天界面
    Rest* restPage;//冥想休息界面
    Achievement* achievementPage;//成就界面
    void openChatDialog();
    AIChat* chatDialog = nullptr; // 对话框指针
```

图 11 功能统合结构

三、小组成员分工情况

- 组长郁凯文：搭建项目主框架，完成 UI 界面设计，负责实现心情日记与成就系统模块。项目后期也担当了大量的功能模块改进工作。同时也撰写了项目的功能文档与作业报告（大部分），拍摄了程序实际运行视频。
- 耿子喻：完成心理测试、冥想与休息、情绪可视化三大模块的开发工作，协助组长完成部分作业报告工作，协助组长完成部分功能模块改进工作。
- 禹慧俊：专注于虚拟 AI 聊天系统功能的实现与优化，协助组长完成部分功能模块改进工作。

四、项目总结与反思

1. 多人合作的配合问题：因三人电脑系统及 QT 版本存在差异，项目初期面临兼容性挑战。通过统一安装 QT6 版本，并针对不同系统环境进行适配调整，成功解决运行问题。同时借助 GitHub 进行代码托管，实现项目的便捷下载与实时更新，保障协作效率。

2. 部分功能未能很好的突出特点，如虚拟心理助手聊天与普通 AI 聊天无太大差异。

3. 部分功能模块尚未很好的完善，如心理测试问卷太少，题目太少，分析尚不准确。情绪图表统计模块仍可以建立更精确更合理的统计标准。