程序设计实习大作业 作业报告

勇攀高峰组：张远洋 王之略 刘曜玮

1 程序功能介绍

1.1 项目名称

摸鱼小助手

1.2 项目背景

在每天的学习生活中，我们希望自己能保持专注，避免“摸鱼”。但往往事与愿违，有时我们会不自觉地想要“摸鱼”，导致很多工作无法按时完成。我们便希望开发出一个软件，帮助大家减少“摸鱼”的时间，变得更自律。

1.3 项目功能概述

我们软件的主体部分是一个积分系统。每当用户进行了专注学习或其它各种自律的行为，可以在软件中进行记录，获取相应的积分；而当用户进行了“摸鱼”等行为，也可以在软件中进行记录，扣除相应的积分。通过这样的积分系统，用户可以定量化地了解自己最近的专注情况，且由于获取积分的成就感和失去积分的损失厌恶心理，可以变得更加自律，提升学习、工作的效率。

1.4 具体功能设计

注：以下截图中的数据来自我们的演示账号（用户名：演示账号；密码：123456），可登录进行查看。使用该账号也可更方便地测试统计、数据同步等功能。

1.4.1 模版

用户可预先设定若干事项作为模版，添加记录时可通过模板快速添加。模板中支持设定任意多个变量，并可通过设定的公式从变量计算出最终结果。

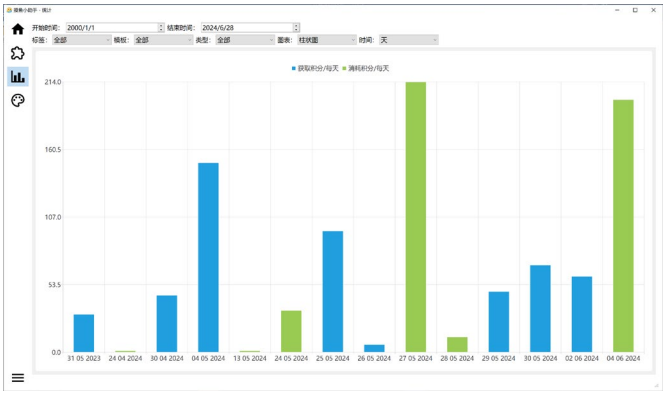
例：模板：使用手机 $\{x\}$ 分钟。扣除 $\max(0, x-180)$ “积分”。

1.4.2 记录

以时间线的形式展示所有记录，可按天/周/月等不同的详细程度展示。可在任意一天添加、删除、修改记录。

1.4.3 数据的统计

可统计特定时间范围内、满足特定条件的记录情况，统计“积分”总的获取与消耗情况。通过漂亮、多样的图表呈现。



1.4.4 账号系统与数据同步

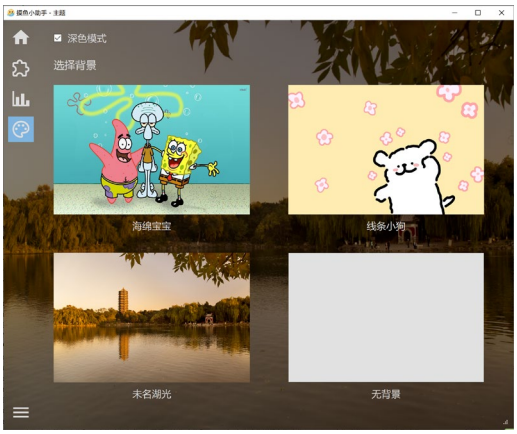
用户可在不同的设备上登录账号，相同账号的模版、记录、设置等内容均可通过云端自动同步。

1.4.5 专注模式

如果用户认为自己用电脑摸鱼了太长时间，可以开启专注模式。专注模式开启期间其窗口将保持置顶，无法通过指定方式以外的方式退出，无法使用任何键盘快捷键。如果切换到其它桌面，还会自动提示返回专注模式。专注模式中还支持开启深色模式、或者切换背景图片，以提供更好的体验。专注模式分为普通模式和倒计时模式，倒计时模式下用户可提前设定期望的专注时间，未到指定时间将无法退出。

1.4.6 主题

软件为用户提供了不同的主题。用户可根据自身喜好选择背景图片、开启深色模式等。



2 项目各模块与类设计细节

2.1 数据结构与类的设计

2.1.1 模版功能的类设计

所有模版均为类 Mod 的对象，该类的字段定义如下：

```
class Mod {
    int id;
    QUuid uuid; // 唯一标识一个模板，在网络同步时发挥作用
    enum RECORD_TYPE type; // 模板类型：OBTAIN 或 CONSUME;
    bool deleted; // 标记为 deleted 的模板仍存在数据库中，但不在模板界面显示
    QString name; // 记录显示出来的信息
    QString short_name; // 模版的简称，用于添加记录时选择模版、数据统计等功能
    Formula *formula; // 用于计算积分的公式
    int input_num; // 记录模板中的变量数量
    std::vector<QString> variable; // 记录一个模板中的所有变量名
    std::vector<QString> labels; // 记录模板的标签
}
```

2.1.2 记录功能的类设计

记录由两个不同的类定义：RecordDirect 和 RecordByMod，分别代表直接添加的记录和通过模板添加的记录。它们继承自共同的基类：Record. 这三个类的字段定义如下：

```
class Record {
    int id; // 用于标识本地数据库中的 id
    QUuid uuid; // 用于云端同步标识的 id
    long long created_time; // 记录创建的时间戳
    QDate date; // 该条记录所在的日期
}
class RecordDirect: public Record {
    QString name; // 该条记录的名称
    enum RECORD_TYPE type; // 该条记录的类型
    int point; // 该条记录的积分
}
class RecordByMod: public Record {
    Mod *mod; // 该条记录所使用的模板
    double *inputs; // 该条记录的变量
}
```

其中，Record 类定义了下面这些虚函数接口：

```
virtual enum RECORD_CLASS get_class() const = 0;
virtual enum RECORD_TYPE get_type() const = 0;
virtual int get_point() const = 0;
virtual QString get_display_name() const = 0;
// 以下两个函数用于序列化、反序列化每种类型的 record 中存储的信息，以存入数据库中或从数据库中取出
virtual QString to_string() const = 0;
virtual void from_string(std::unordered_map<QString, Mod *> uuid_map, QString s) = 0;
```

此外，我们定义了继承自 std::deque<Record *>的 MultipleRecord 类来存储多个 Record 的对象。

2.1.3 内存中存储数据的类设计

为了实现不同模块之间的数据同步与通信，我们定义了 Data 类：

```
class Data : public QObject{
    std::map<QDate, MultipleRecord *, std::greater<QDate>> records; // 记录
    std::vector<Mod*> mods; // 模板
    std::vector<QString> totallabels; // 存放所有标签
    int total_points = 0;
    int last_week_points = 0;
};
```

其中 records 是按天组织的记录数据，方便在主界面将记录按天展示。

2.2 各模块的实现细节

2.2.1 不同功能模块之间的跳转与通信

我们三个组员每个人负责若干个模块，即编写一个 QMainWindow 的子类来实现相应的功能。为了整合不同的功能模块并实现跳转，我们创建了一个 QMainWindow 的子类 MainWindow，其中包含一个 QStackedWidget 来存放不同的 window. MainWindow 的左侧有一个功能栏，用户点击对应功能的按钮，便可改变 QStackedWidget 所展示的 widget，从而切换到不同的页面。

为了实现不同模块之间的数据通信，保证数据的一致性，我们定义了一个类：Data. 该类中存放了所有模版和记录的数据，由 MainWindow 创建并初始化，然后在每个页面的构造函数中传入 MainWindow 创建的 Data 的实例。这样一个功能模块修改了 Data 中的数据，其它功能模块中 Data 的数据也会被同步修改。同时，我们让 Data 继承 QObject，并定义了若干个信号，如：模版被更改、记录被更改等。这样当某个功能模块对 Data 中的数据产生了修改时，可以触发相关的信号，使其它模块可以做出反应，修改对应的 UI.

2.2.2 数据的本地存储

登录信息、模板与记录的数据、各种设置数据均需本地存储。因此，我们使用了 SQLite 数据库来存储各种数据。每个用户的数据存储在独立的数据库中，模版和记录分别对应 `mod` 和 `record` 这两个表。未来，我们计划将数据库换为加密的数据库，并由服务器分发密钥（类似微信电脑版），以保护用户的隐私（非登录状态下将无法查看本地数据库中的数据）。

为了存储一些轻量化的信息，我们创建了一个全局数据库，并在其中创建了 `kv_table` 这个表来存储 key-value 对。我们实现了统一的接口 `save_value` 和 `get_value` 来方便快捷地对这些 key-value 对进行修改与读取。同时我们设置了缓存来减少与数据库的交互，提升运行效率。

2.2.3 数据的云端同步

我们租借了阿里云的服务器，并用 PHP 实现了各种接口，这样应用程序便可以和服务器进行通信。考虑到我们的软件应该允许用户离线使用，且应保证数据能在联网时被同步，因此需要实现类似 Git 的功能。为此，我们需要记录用户对模版、记录的操作历史。

客户端在本地记录了用户与服务器同步了的最后一条操作历史（用 UUID 标识），每当用户进行了新的操作时，客户端将这一信息和所有未同步的操作历史发送给服务器。服务器端将这些未被同步的操作历史记录下来，然后根据客户端发送的已同步的最后一条操作历史，返回给客户端其它地方上传的客户端尚未同步的操作历史。客户端在收到服务器端的返回信息后，根据新的操作历史更新本地存储的数据。对于同一条数据，我们默认时间靠后的操作会覆盖时间靠前的操作，这样便并不会出现冲突，也不存在需要手动合并冲突的问题。

2.2.4 公式与变量

我们创建了一个 `Formula` 类用来存放用户在模板中定义的公式，其中包含一个 `QString` 用于存放公式内容和一个 `std::vector<QString>`用于存放所有变量。用户在想要设置为变量名的名称两侧添加大括号即可被识别为变量，计算公式时先将公式中的中所有变量通过字符串替换的方法替换为数字，转化成表达式求值问题来实现。

2.2.5 模版与标签

所有的模版数据被显示在一个 `QScrollArea` 中，每个模版在 UI 界面中展示其类型、别名以及详细信息，并附有三个功能按钮，分别用于修改模版、删除模版和修改模版的标签。其中标签按钮使用了我们自定义的继承自 `QPushButton` 的 `TagButton` 类。`TagButton` 中有一个 `QTimer` 对象用于计时和一个对话框 `tags_dialog` 对象用于显示标签，计时器计时结束时 `tags_dialog` 将会被自动隐藏。之后我们重载了 `eventFilter`，当鼠标停留按钮或者对话框上时我们停止计时器的计时，离开时则重启计时。由此我们实现了当鼠标不在标签对话框上一段时间后自动隐藏对话框。

当用户删除模版时，为了保证相关的记录不被删除，我们并不会实际地在数据库中删除这一模版，而是将其 `is_deleted` 字段标记为 `true`. 此时，用户无法通过这个被删除的模版添加新的记录，但原来通过这个模板添加的记录仍能被正常地显示、修改。如果用户在修改模版的时候选择不更新之前的记录，则也会将原来的模版保留，但设置 `is_deleted` 字段为 `true`.

2.2.6 记录

主界面中使用一个 `QScrollArea` 来展示所有记录。当用户切换记录展示的时间粒度时，更新 `QScrollArea` 中的内容。当用户点击界面右侧的日历时，计算当前日期的记录所

在的位置，然后使 QScrollArea 跳转到对应位置，实现记录的快速定位。

2.2.7 数据的统计

我们从 Data 的对象中读取标签、模板、积分等数据，之后创建了若干个 QComboBox 使用户可以选择想要统计的数据的标签、模板、类型、时间以及图表样式。我们通过构建一个标签与模板相对应的 map 来实现 QComboBox 中选项的填充，然后根据所选的选项建立一个日期与分数相对应的 map，最后依据此 map 来获取图表所需的数据。在创建 QChart 类的实例后，将数据填充至图表中，实现数据的统计与可视化。

2.2.8 专注模式

专注模式包括普通模式和倒计时模式，在进入专注模式前创建一个 FocusDialog 供用户进行选择。专注模式的 UI 注重简洁，只显示当前时间、名人名言、以及已专注时长（若为倒计时模式则为还需专注时长）。右上角有四个按钮，分别用于刷新名人名言、切换背景图片、切换深色模式、退出专注模式。

为了实现专注模式的功能，我们重载了 closeEvent，使得用户只能通过退出按钮来退出专注模式。同时我们使用键盘钩子捕获并禁用所有的快捷键。对于触摸板切换桌面的问题，我们暂时无法阻止，因此设置一个 QTimer 每 2 秒检测一下专注模式窗口是否在当前桌面的最上方，若不在则弹出始终保持全屏置顶且显示“请返回专注模式”字样的窗口。

2.2.9 主题

为了避免背景图片上的颜色使界面上的部分文字看不清的问题，我们重载了主窗口的 paintEvent 来实现背景图片的设置，并为背景图片设置了透明度。配合 setPalette 函数，我们实现了背景图片在深色模式与浅色模式之间的切换。

由于在深色模式下各种地方显示的颜色都有区别，我们定义了 ColorProvider 类来提供不同的颜色。该类会根据当前是否为深色模式提供不同的颜色，这样应用程序的其它地方便不用考虑当前是否为深色模式，可以使用统一的代码获取符合当前主题的颜色。

对于各种图标，由于需要将其显示为不同的颜色，我们将它们都放入了一款字体中。这样，只需要将 QLabel 或 QPushButton 的字体设置为这一字体便能显示出对应的图标，且可通过设置文本的颜色来改变图标的颜色。

3 小组成员分工情况

选题与创意：张远洋、王之略、刘曜玮
功能设计：张远洋、王之略、刘曜玮
主要功能编写：

模版：王之略
记录：张远洋
统计：刘曜玮
主题：刘曜玮
数据存储与同步：张远洋
专注模式：王之略

测试与优化：张远洋、王之略、刘曜玮
报告编写、视频制作：张远洋、王之略、刘曜玮

4 项目总结与反思

4.1 项目的成果与收获

我们按计划完成了大部分的功能，包括模版、记录、统计、主题、专注模式等等，并通过了路演筛选。通过完成这次大作业，我们锻炼了项目规划能力、代码架构能力、团队协作能力，掌握了 Qt 及 Git 等工具的使用，获益匪浅。

4.2 项目规划的问题及反思

我们在确定项目选题后很快就确定了大部分功能的设计，虽然最后完成了大部分设计中的功能，但项目规划与推进仍存在部分问题。五一前我们计划在五一假期结束后完成模版、记录、统计这三项基本的功能，然而由于大家对 Qt 都不熟悉，进度都很缓慢，均没能按期完成规划。之后直到路演前大部分功能才写完，这导致我们没有充分的时间对成品进行打磨、优化，部分细节处理不到位。也是因此，部分原定完成的小功能，例如：成就与经验系统、分享功能等没有完成。这次大作业的经验告诉我们在完成项目时要制定更为合理、具体的项目规划，并及时相互敦促、落实，以按照预期完成所有功能。