**武汉大学计算机学院**

**本科生课程设计报告**

HomeworkChecker总体设计与实现

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：高级语言程序设计

指 导 教 师 ：常军老师

团 队 成 员 一：林思行（2020302111055）

团 队 成 员 二：杨锦荣（2020302111311）

团 队 成 员 三：伍思烨（2020302111043）

二○二一年六月

**郑 重 声 明**

本团队呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

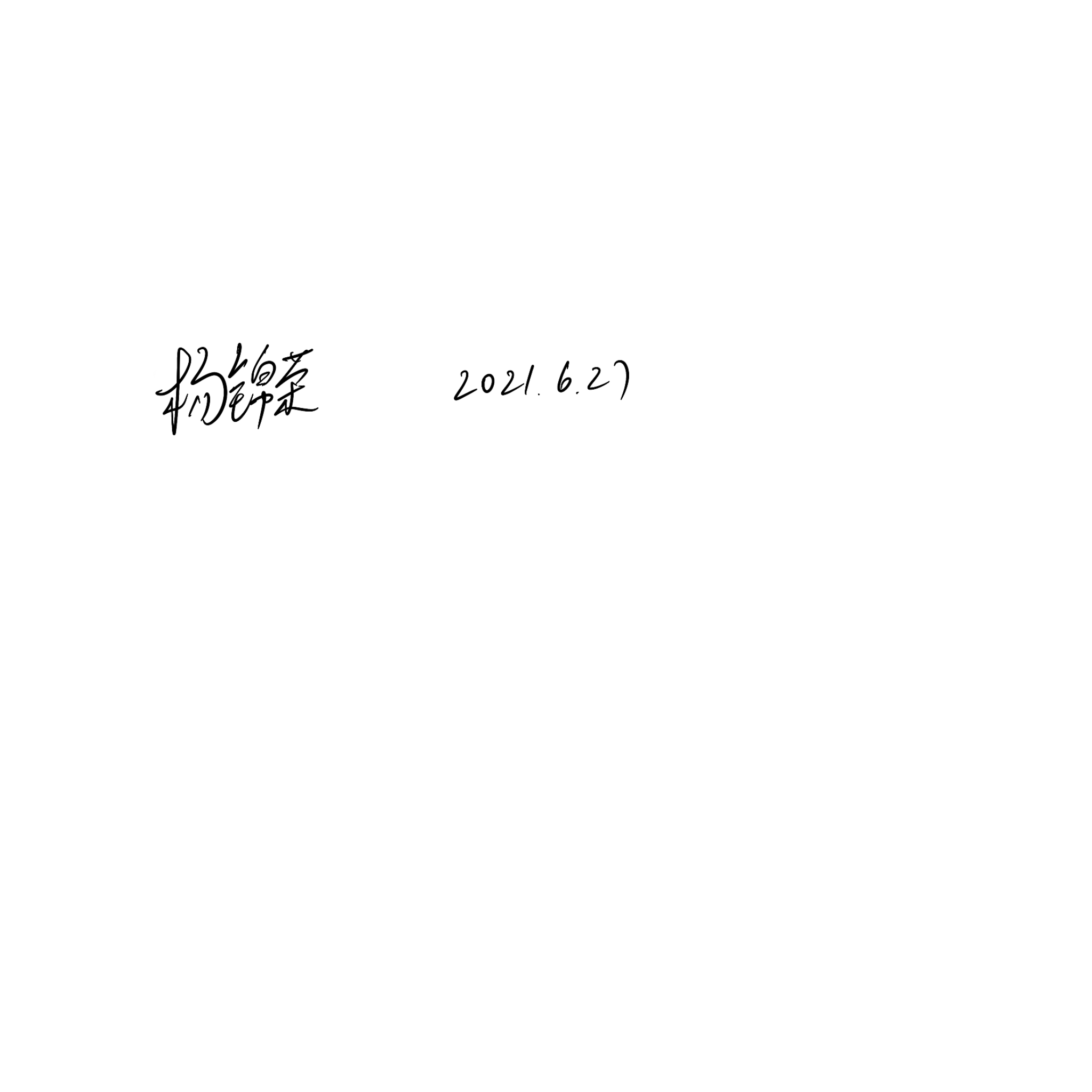
形状

低可信度描述已自动生成

形状

低可信度描述已自动生成

团队成员签名： 日期：



摘 要

本产品为一款服务于大学教师于学生的教学辅助系统，目标人群主要针对大学教师、学生（其是程序设计课的教师和学生），由HomeworkCheckerServer（服务端）和HomeworkChecker（教师端）构成。根据产品定位，教师可在此系统上布置作业、批改作业、记录成绩；学生可以通过发送消息到指定QQ号提交作业；教师批改完毕后会将评语和成绩发送至学生的QQ。期末时教师可以通过此系统一键导出平时成绩。

根据上述关于产品的描述，我们可以归纳出本产品的使用场景：学生查询作业、提交作业、查看批改结果，教师布置作业、批改作业、记录成绩等。

**关键词：**教学辅助;作业;成绩

目 录

[1 项目目的与意义 1](#_Toc75645326)

[1.1 相比传统的人工方式为在校师生提供便利的教学辅助系统 1](#_Toc75645327)

[1.2 在线教育规模扩大的社会背景下提供作业、成绩分析支持 1](#_Toc75645328)

[2 总体设计 3](#_Toc75645329)

[2.1 产品功能结构 3](#_Toc75645330)

[2.1.1 HomeworkCheckerServer服务端（学生使用QQ进行交互） 3](#_Toc75645331)

[2.1.2 HomeworkChecker教师端 3](#_Toc75645332)

[2.1 产品特色 3](#_Toc75645333)

[2.2 软件界面 4](#_Toc75645334)

[2.3 分工 4](#_Toc75645335)

[2.4 协作开发 5](#_Toc75645340)

[3 DataManager模块 6](#_Toc75645341)

[3.1 整体功能 6](#_Toc75645342)

[3.2 项目结构 6](#_Toc75645343)

[3.2.1 引用 6](#_Toc75645344)

[3.2.2 整体构成 6](#_Toc75645345)

[3.2.3 层次关系 7](#_Toc75645346)

[3.3 关键技术 7](#_Toc75645347)

[3.3.1 数据库 7](#_Toc75645348)

[3.3.2 注册登录中对密码的处理 11](#_Toc75645349)

[3.4 总结 12](#_Toc75645350)

[4 QQMessage模块 13](#_Toc75645351)

[4.1 整体功能 13](#_Toc75645352)

[4.2 项目结构 13](#_Toc75645353)

[4.2.1 引用 13](#_Toc75645354)

[4.2.2 整体构成 13](#_Toc75645355)

[4.2.3 组件依赖 14](#_Toc75645356)

[4.3 关键技术 14](#_Toc75645357)

[4.3.1 WebSocket技术 15](#_Toc75645358)

[4.3.2 Analyst分析模块 17](#_Toc75645359)

[4.3.3 跨平台适配 19](#_Toc75645360)

[4.4 总结 20](#_Toc75645361)

[5 HomeworkCheckerServer服务端程序 22](#_Toc75645362)

[5.1 整体功能 22](#_Toc75645363)

[5.2 项目结构 22](#_Toc75645364)

[5.2.1 引用 22](#_Toc75645365)

[5.2.2 整体构成 22](#_Toc75645366)

[5.2.3 组件依赖 22](#_Toc75645367)

[5.3 运行截图 23](#_Toc75645368)

[5.3.1 软件运行 23](#_Toc75645369)

[5.3.2 QQ端用户交互 23](#_Toc75645370)

[6 HomeworkChecker教师端程序 28](#_Toc75645371)

[6.1 整体功能 28](#_Toc75645372)

[6.2 项目结构 28](#_Toc75645373)

[6.2.1 引用 28](#_Toc75645374)

[6.2.2 整体构成 28](#_Toc75645375)

[6.3 关键技术 29](#_Toc75645376)

[6.3.1 编写Qml可调用的自定义C++类 29](#_Toc75645377)

[6.3.2 注册自定义C++类 30](#_Toc75645378)

[6.3.3 在Qml中实例化自定义C++类 30](#_Toc75645379)

[6.3.4 在Qml中调用C++函数 30](#_Toc75645380)

[6.3.5 将C++ Vector容器的内容呈现在Qml的列表上 31](#_Toc75645381)

[6.4 运行截图 32](#_Toc75645382)

[6.5 总结 37](#_Toc75645383)

[结 论 38](#_Toc75645384)

[参考文献 39](#_Toc75645385)

[附 录 40](#_Toc75645386)

[A 服务端QQ用户交互界面详细截图 40](#_Toc75645387)

[B 教师端用户交互界面详细截图 46](#_Toc75645388)

[C 项目文件树状图 49](#_Toc75645389)

[D 内存使用 53](#_Toc75645390)

# 项目目的与意义

## 相比传统的人工方式为在校师生提供便利的教学辅助系统

当今时代是飞速发展的信息时代，在各行各业中离不开信息处理，这正是计算机被广泛应用于信息管理系统的环境。计算机的最大好处在于利用它能够进行信息管理.使用计算机进行信息控制，不仅提高了工作效率，而且大大地提高了其安全性.尤其对于复杂的信息管理,计算机能够充分发挥它的优越性。

随着经济的飞速发展，互联网科技日新月异，社会知识经济在其中的地位越来越高,与此同时，国家综合国力和国际竞争能力越来越取决于教育发展，科学技术和知识创新的水平，在经济和社会发展过程中，教育将呈现出越来越突出的作用。在这个背景下，教师与学生的教学辅助系统是一个教育单位不可缺少的部分，它的内容不仅对于广大学生来说意义非凡，对于学校的决策者和管理者来说都至关重要,所以教学辅助系统应该能够为用户提供充足的便利，如简单易用、界面友好的图形界面等。然而，有些学校还是使用传统人工的方式管理学生成绩，填写各种表格，这种管理方式存在着许多缺点，如:效率低，保密性差，另外所用其时间长，产生大量的文件和数据，这对于查找，更新和维护都带来了不少的困难。

## 在线教育规模扩大的社会背景下提供作业、成绩分析支持

2020年，在线教育再次站在聚光灯下，资本鼎力支持，头部玩家跑马圈地。但随行业发展/科技进步，行业业态在不断进化，020、OMO、双师课堂等新名词不断出现，传统的线下教培机构也在不断加入科技元素，把一部分招生、管理、教辅甚至教学等环节线上化，而线上教培机构也在通过线下地推、开设线下体验店等方式获客。实际上，目前已不存在纯粹的线下教育和线上教育之分，二者在相互改造和渗透。狭义的在线教育需同时具备线上交付，不局限在固定学习场景、随处可学的特征;而广义的在线教育只需满足线上交付即可。[1]

日历

描述已自动生成

图**1.1** 在线教育发展迅速

在在线教育行业市场规模不断增长的背景下，一款能提供作业、成绩分析支持的教学辅助系统便成为迫切需求。作业的布置与批改，成绩的评定与反馈等环节，均为在线教育的重要环节。在此次大作业中，我们综合C++编程来实现一个教学辅助系统。教师可在此系统上布置作业、批改作业、记录成绩；学生可以通过发送邮件到特定QQ号提交作业；教师批改完毕后会将评语和成绩发送至学生的邮箱。期末时教师可以通过此系统一键导出平时成绩。

# 总体设计

本产品为一款服务于大学教师于学生的教学辅助系统，目标人群主要针对大学教师、学生（其是程序设计课的教师和学生）。根据产品定位，教师可在此系统上布置作业、批改作业、记录成绩；学生可以通过发送邮件到特定QQ号提交作业；教师批改完毕后会将评语和成绩发送至学生的邮箱。期末时教师可以通过此系统一键导出平时成绩。

根据上述关于产品的描述，我们可以归纳出本产品的使用场景：学生提交作业、查看批改结果，教师布置作业、批改作业、记录成绩等。

## 产品功能结构

### HomeworkCheckerServer服务端（学生使用QQ进行交互）

* + 查看教师布置的作业
  + 提交作业
    - 文本
    - 附件
  + 查看批改结果
    - 成绩
    - 评语

### HomeworkChecker教师端

* 查看学生提交的作业
* 作业打分、写评语
* 查看、导出平时成绩
* 使用数据库储存学生名单、作业、成绩

## 产品特色

* 有简单易用、用户友好的图形界面
* 软件体积小，内存占用低（见附录D）
* 跨平台（教师端支持Windows、macOS，服务器端支持Windows、Linux）

## 软件界面



图**2.1** HomeworkChecker教师端总览

文本

描述已自动生成

图**2.2** HomeworkCheckerServer服务端运行界面

## 分工

### **共同完成**

* 完善创意
* 书写报告

### **杨锦荣**

* QQMessage模块
* HomeworkCheckerServer服务端
* 服务端客户端网络通信
* 教师端“作业”、“设置”页面及其子页面的功能
* Windows、Linux端开发与调试

### **林思行**

* 撰写需求文档
* 设计UI（见附录B图B.7）
* DataManager模块
* 教师端Qt Qml完善（使用响应式布局，提供窗口缩放支持）
* 教师端“总览”、“班级”、“成绩”页面及其子页面的功能
* Mac端开发与调试

### **伍思烨**

* Qt Qml框架搭建

## 协作开发

本次大作业采用Git进行版本控制和协作开发。项目仓库地址：<https://github.com/NagisaCo/HomeworkChecker.git>

# DataManager模块

## 整体功能

DataManager模块作为沟通教师端、学生端和后端的类库，负责对数据库操作进行抽象，并在此基础上管理整个项目需要用到的数据，包括帐号、班级、学生、教师、作业、成绩等。

## 项目结构

### 引用

* MySQL[6] (第三方库)
* OpenSSL[8] (第三方库)

### 整体构成

注：DataManager既是子项目名也是其中一个子模块的名称

DataManager 数据管理项目 负责人：林思行

├─ DMTest DataManager测试

│ └─ main.cpp

└─ DataManager

├─ CMakeLists.txt

├─ DBManager.cpp

├─ DBManager.hpp 数据库操作函数

├─ DMError.cpp

├─ DMError.hpp DataManager操作异常类

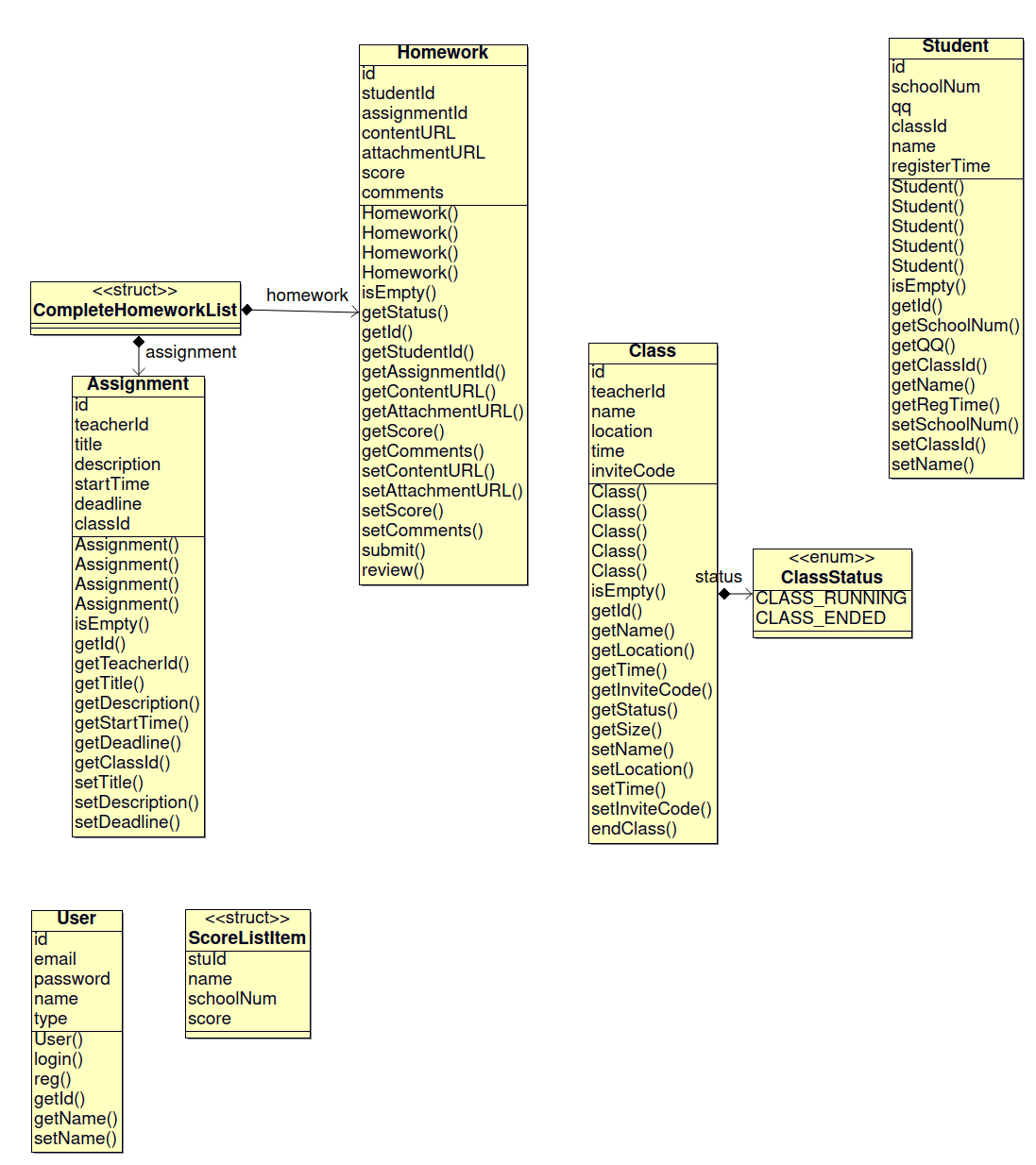
├─ DMUtils.cpp

├─ DMUtils.hpp DataManager实用工具

├─ DataManager.cpp

└─ DataManager.hpp DataManager数据管理类库

### 层次关系



图**3.1** DataManager模块组件依赖

## 关键技术

### 数据库

为了更高效、稳定地储存数据和查询数据，本次大作业使用了在业界广泛使用的MySQL数据库。[7]

#### 数据库表设计

##### 帐号表（users）

表**3.1** 数据库账号表

| **字段** | **id** | **username** | **password** | **type** | **avatar\_url** | **name** |
| --- | --- | --- | --- | --- | --- | --- |
| 数据类型 | INT | VARCHAR(50) | VARCHAR(50) | TINYINT(1) | VARCHAR(100) | VARCHAR(15) |
| 描述 |  | 用户名 | 密码 | 用户类型（0=教师，1=学生） | 头像URL（由于时间关系已去除此功能） | 教师姓名 |

##### 布置的作业表（assignments）

表**3.2** 数据库布置作业表

| **字段** | **id** | **teacher\_id** | **title** | **description** | **start\_date** | **deadline** | **class\_id** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据类型 | BIGINT | INT | VARCHAR(80) | TEXT | TIMESTAMP | TIMESTAMP | INT |
| 描述 |  | 布置的教师ID | 标题 | 描述 | 布置时间 | 截止时间 | 目标班级ID |

##### 提交的作业表（homework）

表**3.3** 数据库提交作业表

| **字段** | **id** | **student\_id** | **assignment\_id** | **content\_url** | **attachment\_url** | **score** | **comments** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据类型 | BIGINT | INT | BIGINT | VARCHAR(600) | VARCHAR(600) | SMALLINT | TEXT |
| 描述 |  | 学生ID | 布置的作业ID | 作答文件URL | 附件URL | 成绩 | 评语 |

##### 班级表（classes）

表**3.4** 数据库班级表

| **字段** | **id** | **teacher\_id** | **name** | **location** | **time** | **code** | **status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据类型 | BIGINT | INT | VARCHAR(25) | VARCHAR(20) | VARCHAR(10) | CHAR(4) | TINYINT(1) |
| 描述 |  | 教师ID | 名称 | 上课地点 | 上课时间 | 邀请码 | 0=未结课，1=已结课 |

##### 学生表（students）

表**3.5** 数据库学生表

| **字段** | **id** | **school\_num** | **qq** | **class\_id** | **name** | **register\_time** |
| --- | --- | --- | --- | --- | --- | --- |
| 数据类型 | INT | CHAR(11) | VARCHAR(15) | INT | VARCHAR(20) | TIMESTAMP |
| 描述 |  | 学号 | QQ号 | 班级ID | 姓名 | 注册时间 |

#### DBManager子模块

对MySQL官方提供的MySQL库和MySQL Connector/C++库进行二次抽象，将连接数据库、查询数据库的操作进行封装，并加入了调试工具，方便对错误进行处理。

提供的函数如下：

/// 连接数据库

/// @param account 数据库帐号

bool connectDatabase(DBAccount account);

/// 关闭数据库连接并释放查询结果使用的内存

void closeConnection();

/// 检查数据库连接是否仍然可用，如果可用，返回0

int checkConnection();

/// 查询数据库

/// @param queryString SQL语句

/// @returns code 错误代码（0=成功）

int query(std::string queryString);

/// 获取数据

/// @param table 表名

/// @param columnNames 列名（SQL格式）

int select(std::string table, std::string columnNames);

/// 获取数据

/// @param table 表名

/// @param columnNames 列名（SQL格式）

/// @param conditions 条件（SQL格式）

int select(std::string table, std::string columnNames, std::string conditions);

/// 获取数据

/// @param table 表名

/// @param columnNames 列名（SQL格式）

/// @param conditions 条件（SQL格式）

/// @param order 排序方式（SQL格式）

int select(std::string table, std::string columnNames, std::string conditions, std::string order);

/// SELECT操作结果的行数

unsigned long numRows();

/// 获取结果的下一行，等于mysql\_fetch\_row

MYSQL\_ROW fetchRow();

/// 插入数据

/// @param table 表名

/// @param columnNames 列名（以英文逗号分隔）

/// @param values 数据字符串（SQL格式）

int insert(std::string table, std::string columnNames, std::string values);

/// 更新数据

/// @param table 表名

/// @param columnAndValue 列名-值（SQL格式）

int update(std::string table, std::string columnAndValue);

/// 更新数据

/// @param table 表名

/// @param columnAndValue 列名-值（SQL格式）

/// @param conditions 条件（SQL格式）

int update(std::string table, std::string columnAndValue, std::string conditions);

/// 删除数据

/// @param table 表名

/// @param conditions 匹配条件（SQL WHERE语句格式）

int remove(std::string table, std::string conditions);

/// INSERT、UPDATE、DELETE操作影响的行数

unsigned long affectedRowCount();

/// SQL注入检查

/// @param str 要检查的字符串

std::string sqlInjectionCheck(std::string str);

#### DataManager子模块

该模块在DBManager的基础上对数据库进行再次抽象，数据库几乎不可见。数据操作转化为User、Class、Student、Homework、Assignment五个C++类的函数以及DataManager命名空间内的类外函数。抽象之后的类关系参见“层级关系”节。

### 注册登录中对密码的处理

尽管数据库连接使用了SSL，但为了避免在网络中传输原始密码和在数据库中储存原始密码，还是使用了hash函数对密码进行处理，网络中传输的和数据库内存放的仅是密码的哈希值。

生成哈希值的函数：（BKDR算法）

unsigned int BKDRHash(const std::string str) {

char \*strPtr = const\_cast<char \*>(str.c\_str());

unsigned int seed = 131;

unsigned int hash = 0;

while (\*strPtr) {

hash = hash \* seed + (\*strPtr++);

}

return (hash & 0x7FFFFFFF);

}

## 总结

通过负责该模块的设计与编写，了解了SQL数据库的操作，学习了C++的异常机制，并且对较复杂的过程分析的能力有一定提升。

C++下第三方库的使用，由于缺少较成熟的包管理，部分库文件需要下载源码本地编译（本模块中自己编译了OpenSSL库），并且自己添加include path和library path，管理库依赖。经历了非常多的链接错误，结合本学期“计算机系统基础”课程中对链接的介绍，更全面深刻地了解了头文件、静态链接、动态链接。

本模块的优点：

* 抽象程度较高，数据库对上层应用基本不可见
* 类层次简单，耦合度低
* 提供了丰富的调试工具，通过定义DEBUG、VERBOSE宏可以调整输出调试信息的等级

由于时间和能力限制，本模块还有许多不足之处：

* 未实现异步通信，上层应用也未对此进行处理，与服务器通信的时候会阻塞主线程，导致网络差的时候教师端有明显的卡顿
* 异常处理并未完善，DBManager和DataManager的部分函数对异常的处理仅是返回某个特定的非0整数（由MySQL库决定）
* 未实现本地缓存功能，查询请求冗余度较高
* 对SQL注入有一定的防护功能，但仍有办法攻破（限于时间关系仅做了非常简单的处理）

# QQMessage模块

## 整体功能

接收来自go-cqhttp发送的websocket消息，对qq收到的私聊消息进行命令分析，返回响应文本，接收作业提交内容，存储到服务器中并更新数据库。

## 项目结构

### 引用

* DataManager
* Nlohmann::json[3] (third-party library)
* websocketpp[4]  (third-party library)
* asio[5]  (third-party library)
* mysql[6] (third-party library)

### 整体构成

QQMessage QQ消息处理程序 负责人：杨锦荣

├─ Analyst 文本处理及分析

├─ Exception 自定义异常类

├─ File 本地文件管理类

├─ FileInfo 文件信息类

├─ PrivateMessageGetter 接收私聊消息类

├─ PrivateMessageSender 发送私聊消息类

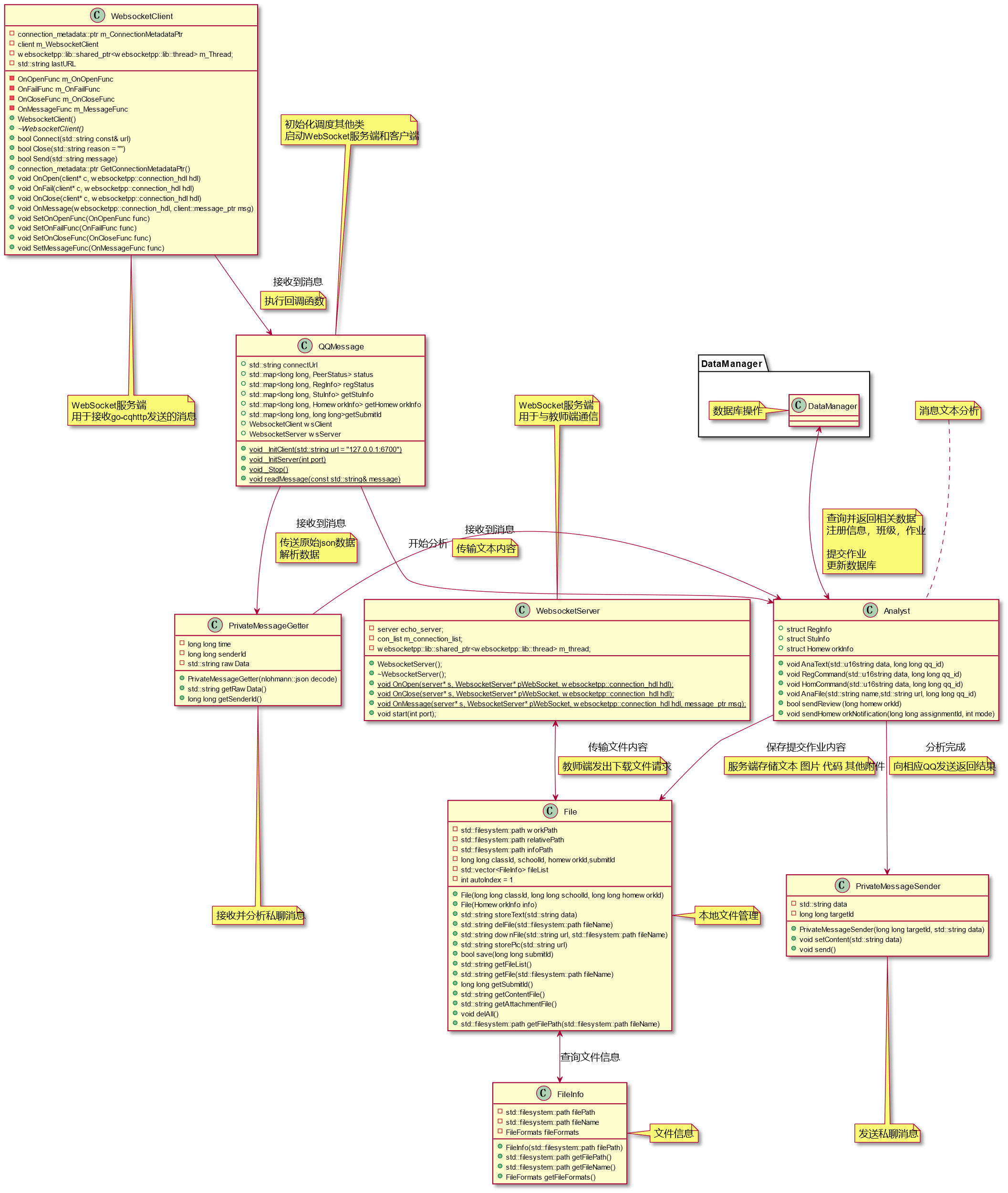
├─ QQMessage QQ消息处理主程序

├─ Tools 工具包

├─ WebsocketClient WebSocket客户端

└─ WebsocketServer WebSocket服务端

### 组件依赖



图**4.1** QQMessage模块组件依赖

## 关键技术

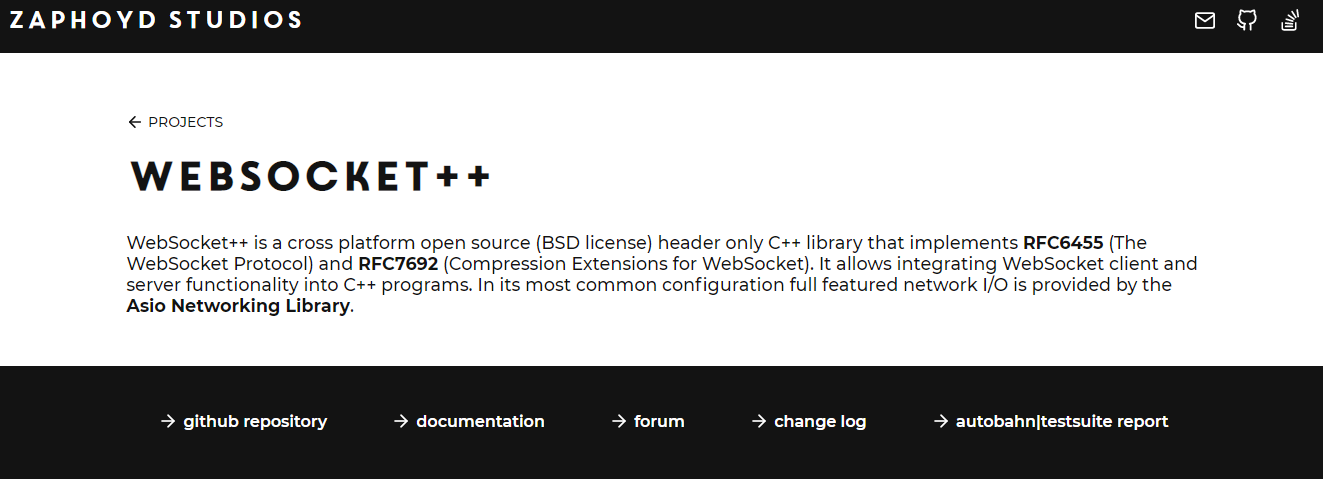
QQMessage模块作为服务端的核心构成，需要对接收到的消息进行处理分析，并与教师端App进行通讯。

### WebSocket技术

WebSocket作为在单个TCP连接上进行的全双工通信，有效的克服了HTTP协议通讯发起的单向性，避免使用轮询，提高通信效率。[2]

#### 第三方库选择

由于目前C++17标准库中并没有网络库，需要使用Asio网络库，作为boost库中优秀的一部分，现以可以在C++17下单独使用，不再依赖于boost。基本网络库之上选择了[websocketpp](https://github.com/zaphoyd/websocketpp)[4]，根据使用文档和相关例程简化websocket服务端与客户端，同时引入多线程避免网络通信时阻塞主线程。



图**4.2** WEBSOCKET++项目官网

#### 多线程

在编写网络通信部分时，会发现由于网络传输的速度远低于访存操作，当接受或发送大量数据时无法同时解析qq消息文本，会造成服务的中断，于是需引入异步编程与多线程技术。

利用asio和c++标准库中的多线程技术，实现了websocket服务端和客户端独立于主线程运行，并在程序退出时终止其他线程。

m\_Thread = websocketpp::lib::make\_shared<websocketpp::lib::thread>(&client::run, &m\_WebsocketClient);

m\_Thread->join();

#### 定义json传输规范

服务端与客户端通信内容实用json编码，便于解析数据。设计时分析双端通信需求，简要列出关键数据及key值。

[action] send\_review get\_file send\_file\_info send\_file heartbeat send\_notification

send\_review:

send: [homeword\_id]

return: [homeword\_id] [status]"success"/"fail"

get\_file:

[homework\_id]

[file\_name]

send\_file

[transfer\_id]

[part]

[content]

{

"action": "send\_file",

"transfer\_id": "10254",

"homework\_id": "19",

"name": "1.png",

"totol\_part": "2",

"part\_size": "1048576",

"size": "1128402",

"status ": "start "

}

heartbeat

send\_notification:

[type] new\_homework

#### 文件传输分块

依据websocket的特性，在websocket之上可以一次性发送大量数据，但是实际操作中服务端发送大量数据，客户端在接受过程中由于长时间未响应会造成连接异常中断，导致数据传输不完整。所有要在应用层对所发送文件进行分块，根据分块顺序上传至客户端。

##### 接收到传输文件指令，获取文件本地路径及文件信息

File file(st.getClassId(), st.getId(), as.getId());

std::string fileName = decode.at("file\_name");

##### 准备缓冲区，将文件分段存入内存

std::ifstream ifs(file.getFilePath(fileName), std::ios::binary);

ifs.seekg(0, ifs.end);

int length = (int)ifs.tellg();

if (ifs) {

ifs.seekg(0, ifs.beg);

std::string buffer(BUFFER\_SIZE, '\0');

int offset = 0;

int readSize = std::min(BUFFER\_SIZE, length - offset);

while (readSize > 0 && ifs.read(&buffer[0], readSize)) {

fileContent.push\_back(buffer);

offset += readSize;

readSize = std::min(BUFFER\_SIZE, length - offset);

}

}

ifs.close();

##### 发送文件分块传输初始化信息

##### 依次发送分段数据

##### 传输结束后发送结束消息

### Analyst分析模块

该模块是整个QQMessage模块内逻辑关系最为复杂的一个部分，整体采用面向过程的方法进行编写，对QQ接收到的消息进行切分，识别相应指令并按照需求进行回应。

#### 分阶段处理消息

设计时对学生输入命令的全过程进行详细的分析，确定关键的命令列表，将学生操作模式分为：未注册、空闲、注册中、提交作业中，四种状态。本地保存学生的各种状态信息，确保在一个消息结束后可以正常执行操作流程。

extern std::map<long long, PeerStatus> status;

extern std::map<long long, RegInfo> regStatus;

#### 本地缓存学生信息

由于测试时网络连接不稳定，访问服务器耗时较大，同时对服务器造成巨大开销。为减少服务器访问次数，对学生的关键信息进行本地保存，兼顾数据时效性的前提下尽可能缓存频繁使用的数据。

/// <summary>

/// 学生信息

/// </summary>

struct StuInfo

{

/// <summary>

/// 学生id

/// </summary>

long long studentId;

/// <summary>

/// 学号

/// </summary>

long long studentNum;

/// <summary>

/// 班级id

/// </summary>

long long classId;

};

/// <summary>

/// 作业提交详情

/// </summary>

struct HomeworkInfo

{

/// <summary>

/// 学生id

/// </summary>

long long studentId;

/// <summary>

/// 学号

/// </summary>

long long studentNum;

/// <summary>

/// 班级id

/// </summary>

long long classId;

/// <summary>

/// 作业id

/// </summary>

long long homeworkId;

/// <summary>

/// 提交id

/// </summary>

long long submitId;

};

### 跨平台适配

#### 文件编码统一UTF-8

在跨平台开发时(Win Mac Linux)，Win下的GBK文件编码导致发送中文消息时产生乱码。于是统一文件编码为UTF-8，并且在处理中文字符串时注明u8标识符。

#### 中文字符串采用u16string与u16char类型

由于Windows与Linux下c++宽字符(wchar)的长度不统一，导致对字符串split操作时需要考虑wchar的两种大小。于是选用c++较新支持的u16string类型，减少不同平台的适配。

#### 宏定义区分不同平台

软件需要使用部分系统函数，需要根据win和linux具体的情况进行适配，了解不同系统的库差异。

std::string File::downFile(std::string url, std::filesystem::path fileName)

{

if (std::filesystem::exists(workPath / fileName))

{

fileName = fileName.stem().string() + std::to\_string(Tools::getTimestamp()) + "." + fileName.string().substr(fileName.string().find\_last\_of(".") + 1);

}

try

{

#if defined(WIN32) || defined(\_WIN32) || defined(\_\_WIN32) && !defined(\_\_CYGWIN\_\_)

#include <Windows.h>

#pragma comment(lib, "urlmon.lib")

URLDownloadToFile(NULL, stringToLPCWSTR(url), stringToLPCWSTR((workPath / fileName).string()), 0, NULL);

#else

std::string file=(workPath / fileName).string();

std::string comm="curl \""+url+"\" --create-dirs -o \""+file+"\"";

std::cout << comm<<std::endl;

system(comm.c\_str());

#endif

return fileName.string();

}

catch (...)

{

return "";

}

}

#### 处理服务端与客户端中文不同编码方式

由于不同系统采用GBK和UTF-8不同编码方式，对本地文件进行操控时需要适配客户端与服务端分别采用GBK和UTF-8多种情况。程序内部统一UTF-8编码，在进行本地文件读写时进行编码转换。

#### 编写CMake文件，实现跨平台编译

cmake\_minimum\_required(VERSION 3.8)

project(QQMessage)

set(CMAKE\_BUILD\_TYPE "Debug")

add\_definitions(-std=c++17)

aux\_source\_directory(. DIR\_SRCS)

link\_directories(

/usr/lib/x86\_64-linux-gnu

../DataManager/DataManager/build

)

include\_directories(

/usr/local/lib

/usr/include/mysql

../packages/asio/include

../packages/json

../packages/websocketpp/include

../DataManager/DataManager

)

add\_library(QQMessage STATIC ${DIR\_SRCS})

target\_link\_libraries(QQMessage libmysqlclient.so libDataManager.a)

## 总结

通过负责该模块的设计与编写，了解了网络编程与数据库的操作，学习了C++的异常机制，实现了多线程编程，并且对较复杂的过程分析的能力有一定提升。

C++下第三方库的使用，由于缺少较成熟的包管理，大多数库文件需要下载源码本地编译，并且自己添加include path和library path，管理库依赖。经历许多的链接错误，全面深刻的了解了头文件的作用，以及静态链接、符号重定位的过程。

由于时间和能力限制，本模块还有许多不足之处

* 设计时类间过于复杂，导致编写程序时过于繁琐
* 缓存设计不规范，后续学习中了解到规范的多路组相联缓存设计和LRU、LFU算法，同时缓存实现可以放入DataManager模块，减少模块间耦合程度
* 异常处理并未完善，没有对不同异常进行分情况处理。很多时候捕获异常只是为了防止程序中断
* 服务端配置信息未进行离线化本地保存，修改ws地址和端口需对程序源码修改
* 分块文件上传模块只有基本功能，没有实现文件校验、乱序传输、断点续传等实用功能

# HomeworkCheckerServer服务端程序

## 整体功能

服务端启动程序，主要调用QQMessage模块

## 项目结构

### 引用

* QQMessage

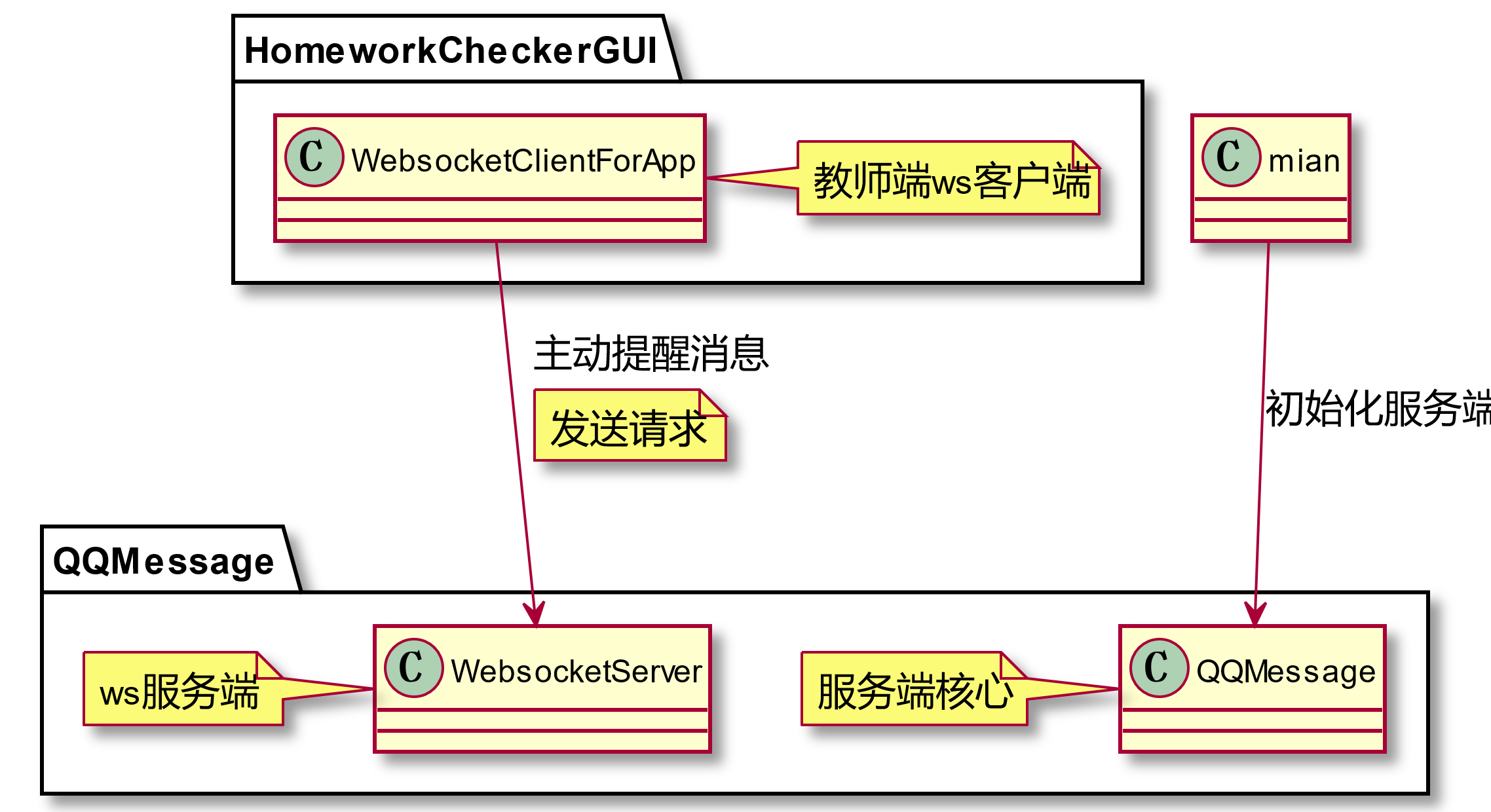
### 整体构成

HomeworkCheckerServer 服务端项目 负责人：杨锦荣

├─ CMakeLists.txt CMake编译文件

└─ HomeworkCheckerServer.cpp 服务端程序

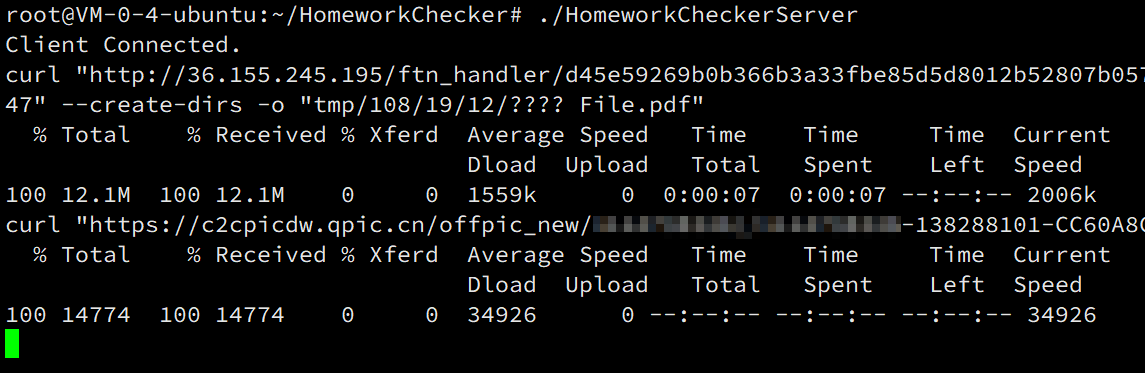
### 组件依赖



图**5.1** HomeworkCheckerServer程序组件依赖

## 运行截图

### 软件运行



图**5.2** 服务端命令行界面

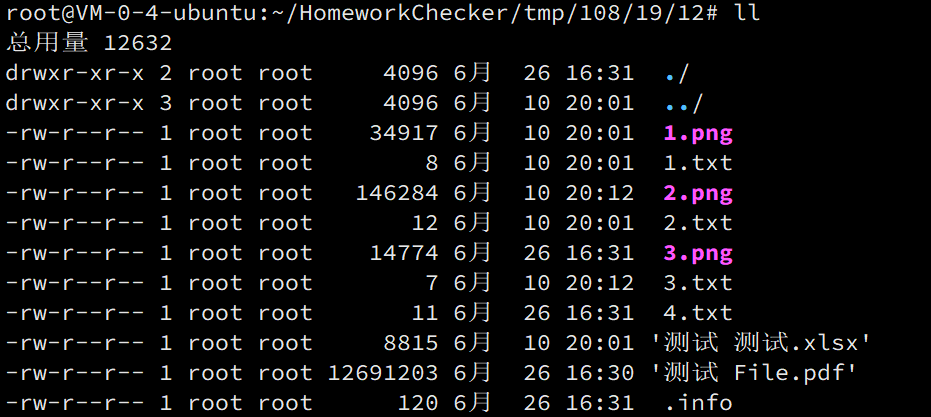
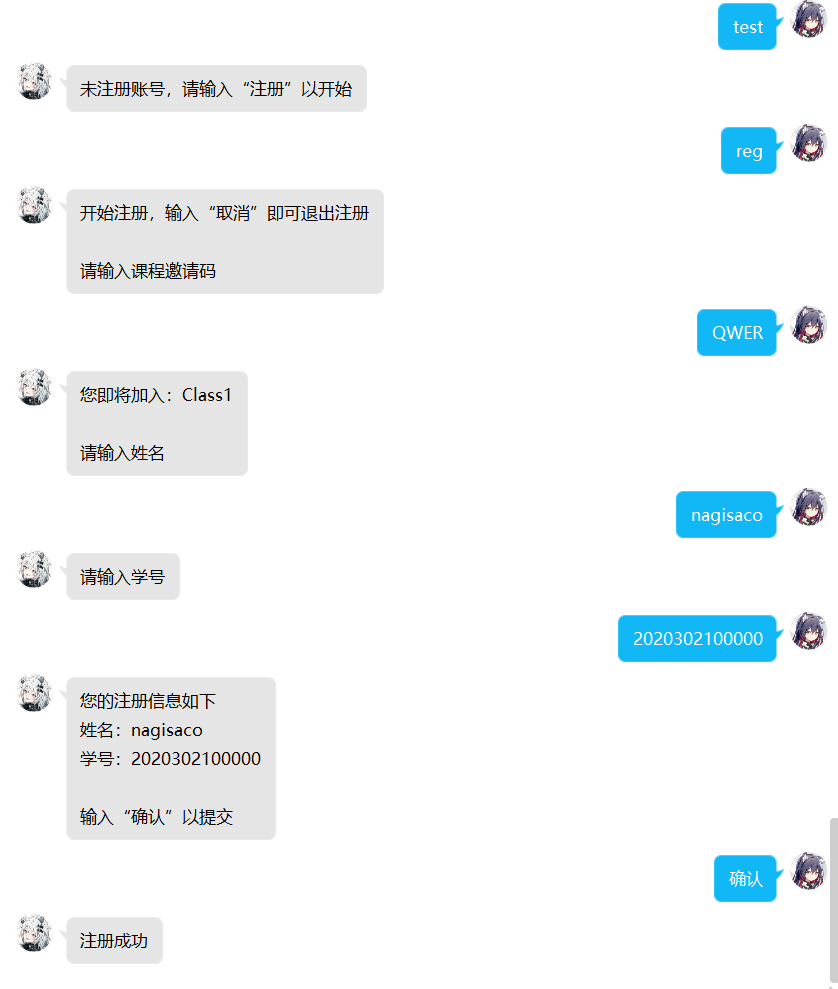


图 **5.3** 服务端单次作业提交文件保存

### QQ端用户交互

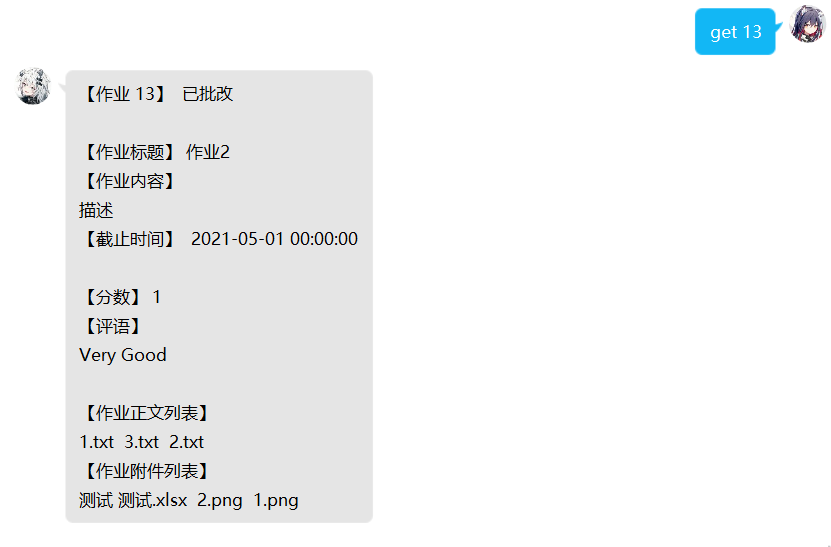
详细交互截图请见：附录A 服务端QQ用户交互界面详细截图。

#### 注册



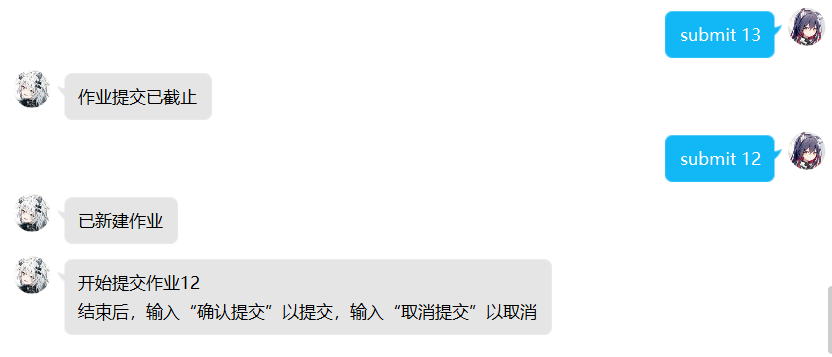
图**5.4** 注册阶段截图

#### 查询作业

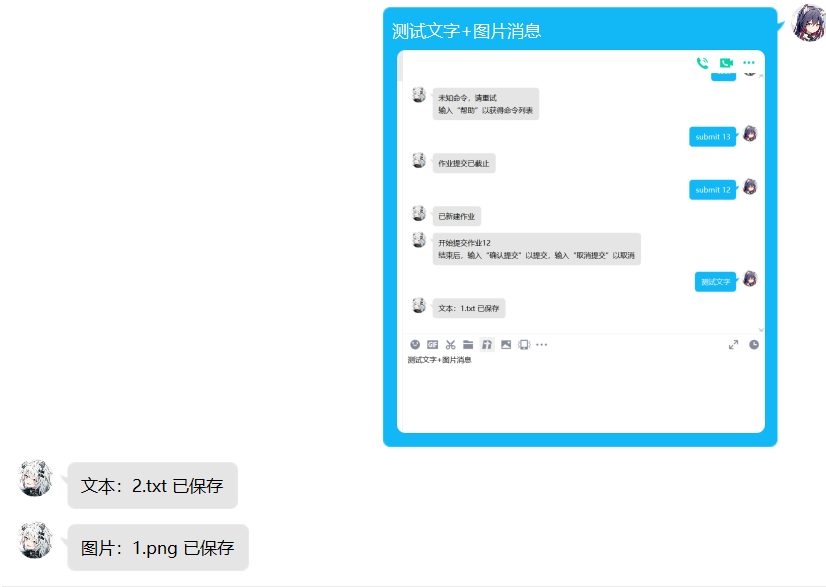


图**5.5** 查询作业详情

#### 提交作业



图**5.6** 开始提交

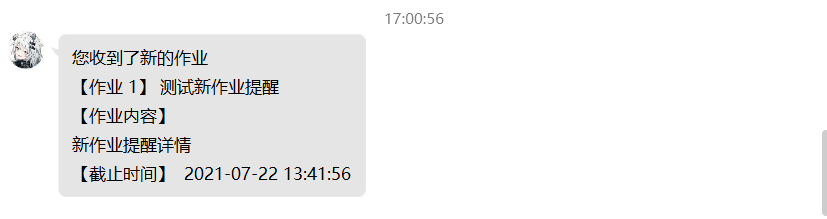


图**5.7** 提交作业内容

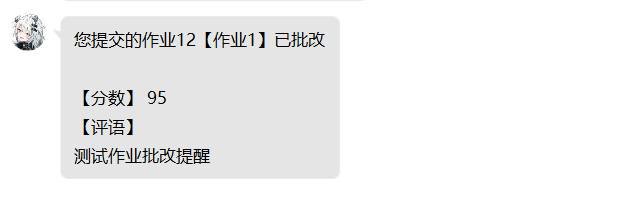


图**5.8** 提交作业

#### 新消息主动提醒



图**5.9** 新作业提醒



图**5.10** 批改完成提醒

# HomeworkChecker教师端程序

## 整体功能

为了提升软件的易用性，我们为教师端提供了图形用户界面。教师端所有操作均可在图形化界面中完成。

## 项目结构

### 引用

* Qt （第三方库）

### 整体构成

HomeworkCheckerGUI  
├─ ClassPage.qml “班级”页面  
├─ DateRectangle.qml 日期选择器  
├─ GeneralPage.qml “总览”页面  
├─ InfoOfClass.qml 班级信息  
├─ InfoOfTask.qml 作业信息  
├─ LaunchScreen.qml 启动页  
├─ LoginPage.qml 登录页  
├─ MarkPage.qml “成绩”页面  
├─ NewAClass.qml 新建班级  
├─ NewAssignment.qml 新建作业  
├─ RegPage.qml 注册页  
├─ SettingPage.qml “设置”页面  
├─ TaskPage.qml “作业”页面  
├─ WebsocketClientForApp.cpp  
├─ WebsocketClientForApp.h  
├─ account.cpp  
├─ account.h 全局帐号管理  
├─ classscoreviewcontroller.cpp  
├─ classscoreviewcontroller.h 班级分数页面控制器  
├─ classviewcontroller.cpp  
├─ classviewcontroller.h ”班级“页面控制器  
├─ correctHomework.qml 批改作业  
├─ correcthomework.cpp  
├─ correcthomework.h 批改页面控制器  
├─ generalviewcontroller.cpp  
├─ generalviewcontroller.h ”总览“页面控制器  
├─ infooftask.cpp  
├─ infooftask.h 作业详情页面控制器  
├─ main.cpp 启动函数  
├─ main.qml 主窗口  
├─ qml.qrc qml资源配置文件  
├─ scoreWithClass.qml 班级成绩  
├─ scoreWithStudent.qml 学生成绩  
├─ setting.config  
├─ settingpage.cpp  
├─ settingpage.h ”设置“页面控制器  
├─ studentscoreviewcontroller.cpp  
├─ studentscoreviewcontroller.h 学生成绩页面控制器  
├─ taskpage.cpp  
├─ taskpage.h ”作业“页面控制器

## 关键技术

为了配合C++来制作图形界面，我们选用了Qt Quick框架。每个页面对应一个Qml文件。

Qml负责呈现页面，而页面的功能需要通过qml内嵌的js调用C++的函数来实现。

### 编写Qml可调用的自定义C++类

以Account类为例：

#include <QObject>  
#include <QString>  
#include <QDebug>  
#include "qqml.h"  
#include "DataManager.hpp"  
​  
class Account: public QObject {  
   Q\_OBJECT  
   Q\_PROPERTY(int id READ getId CONSTANT)  
   Q\_PROPERTY(QString userName READ getName WRITE setName)  
   QML\_ELEMENT  
public:  
   explicit Account(QObject \*parent = nullptr);  
   Q\_INVOKABLE int reg(QString userName, QString password);  
   Q\_INVOKABLE int login(QString userName, QString password);  
   Q\_INVOKABLE bool connectDB();  
   static int getId();  
   static QString getName();  
   static void setName(const QString &name);  
     
private:  
   static DataManager::User user;  
};

### 注册自定义C++类

在Qt App引擎启动之前使用以下的方法注册自定义C++类（视图控制器）：

//自定义CPP模块  
qmlRegisterType<GeneralViewController>("GeneralVC",1,0,"GeneralVC");  
qmlRegisterType<Account>("Account",1,0,"Account");  
qmlRegisterType<SettingPage>("MySettingPage", 1, 0, "MySettingPage");  
qmlRegisterType<ClassViewController>("ClassVC", 1, 0, "ClassVC");  
qmlRegisterType<TaskPage>("AssignmentVC", 1, 0, "AssignmentVC");  
qmlRegisterType<InfoOfTask>("HomeworkVC", 1, 0, "HomeworkVC");  
qmlRegisterType<ClassScoreViewController>("ClassScoreVC", 1, 0, "ClassScoreVC");  
qmlRegisterType<StudentScoreViewController>("StudentScoreVC", 1, 0, "StudentScoreVC");  
qmlRegisterType<CorrectHomework>("HomeworkInfoVC", 1, 0, "HomeworkInfoVC");

### 在Qml中实例化自定义C++类

import ClassVC 1.0  
...  
ClassVC {  
  id: classVC  
}

### 在Qml中调用C++函数

function refresh() { //js函数  
   classVC.refresh() //C++函数  
   classListModel.clear()  
   endedClassListModel.clear()  
   classVC.classList.forEach(ele => {  
     if (ele.status === 0) {  
         classListModel.append(ele)  
    } else {  
         endedClassListModel.append(ele)  
}  
})  
}

### 将C++ Vector容器的内容呈现在Qml的列表上

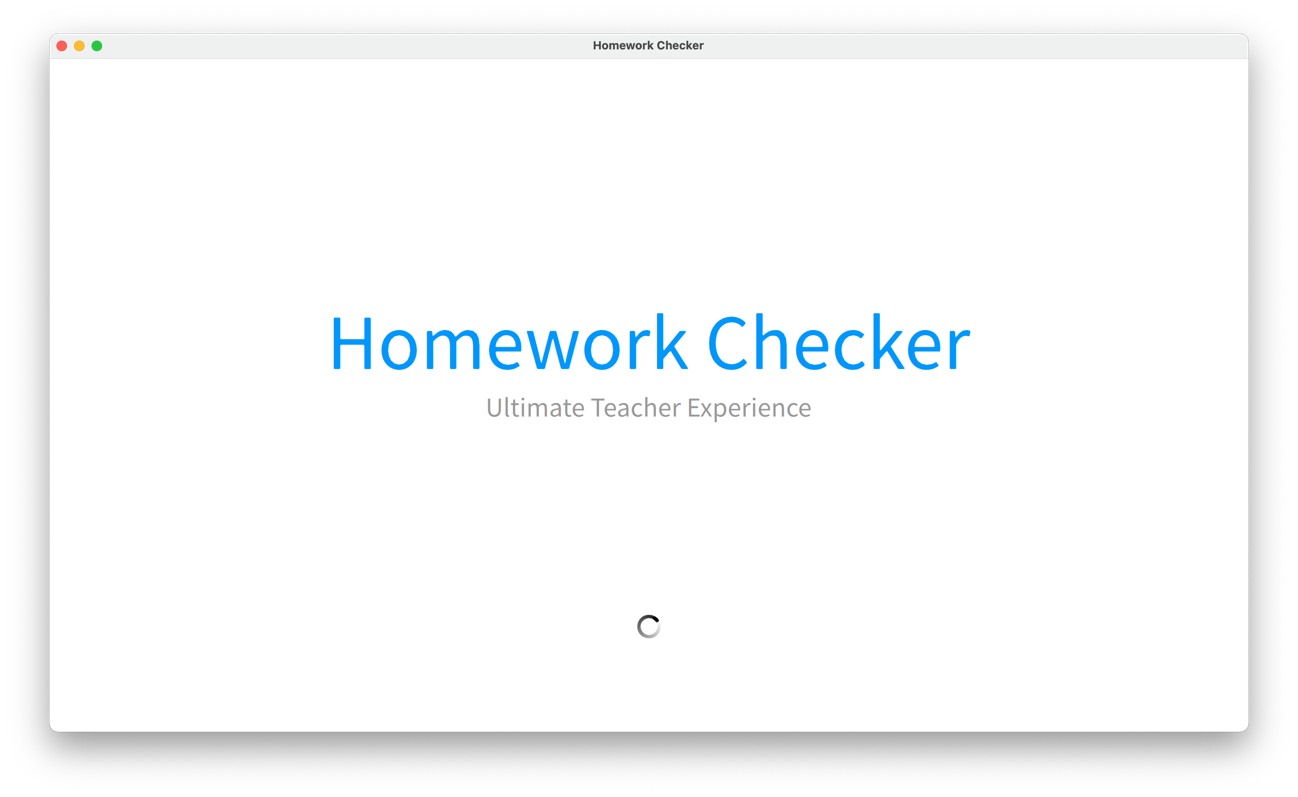
//classscoreviewcontroller.h 截取部分  
Q\_PROPERTY(QJsonArray scoreList READ getScoreList CONSTANT)  
public:  
Q\_INVOKABLE void getData(long classId);  
   QJsonArray getScoreList();

//classscoreviewcontroller.cpp 截取部分  
void ClassScoreViewController::getData(long classId) {  
//清空原有列表  
   while (scoreList.count()) {  
       scoreList.pop\_back();  
  }  
   try {  
    //获取数据  
       std::vector<DataManager::ScoreListItem> list = DataManager::getScoreList(classId);  
       if (list.size() == 0)  
           return;  
       double min = (std::numeric\_limits<double>::max)(), max = 0, sum = 0;  
    //遍历vector  
       for (auto item : list) {  
           if (item.score > max) {  
               max = item.score;  
          }  
           if (item.score < min) {  
               min = item.score;  
          }  
           sum += item.score;  
        //新建JSON对象  
           QJsonObject obj;  
           obj.insert("stuId", QString::fromStdString(std::to\_string(item.stuId)));  
           obj.insert("name", QString::fromStdString(item.name));  
           obj.insert("schoolNum", QString::fromStdString(item.schoolNum));  
           obj.insert("score", QString::fromStdString(DMUtils::double2FixedStr(item.score, 2)));  
        //将JSON对象添加至JSON数组  
           scoreList.append(obj);  
      }  
       highestScore = max;  
       lowestScore = min;  
       avgScore = sum / list.size();  
  } catch (DataManager::DMError error) {  
       qDebug() << "[ERROR] [ClassScoreViewController] " << error.what() << Qt::endl;  
  }  
}

//scoreWithClass.qml  
//数据模型  
ListModel {  
  id: scoreListModel  
}  
//加载数据  
function loadData() {  
  classScoreVC.getData(classId)  
  scoreListModel.clear()  
  highest.text = classScoreVC.highestScore.toFixed(2)  
  average.text = classScoreVC.avgScore.toFixed(2)  
  lowest.text = classScoreVC.lowestScore.toFixed(2)  
  //遍历JSON数组  
  classScoreVC.scoreList.forEach(ele => {  
      scoreListModel.append(ele)  
  })  
}

## 运行截图

此处仅展示macOS上的运行截图，Windows上的运行截图与此几乎相同，故不重复展示。详细截图请见：附录B 教师端用户交互界面详细截图。

 图**6.1** 软件启动加载页面

 图**6.2** 登录页面

图**6.3** 总览页面

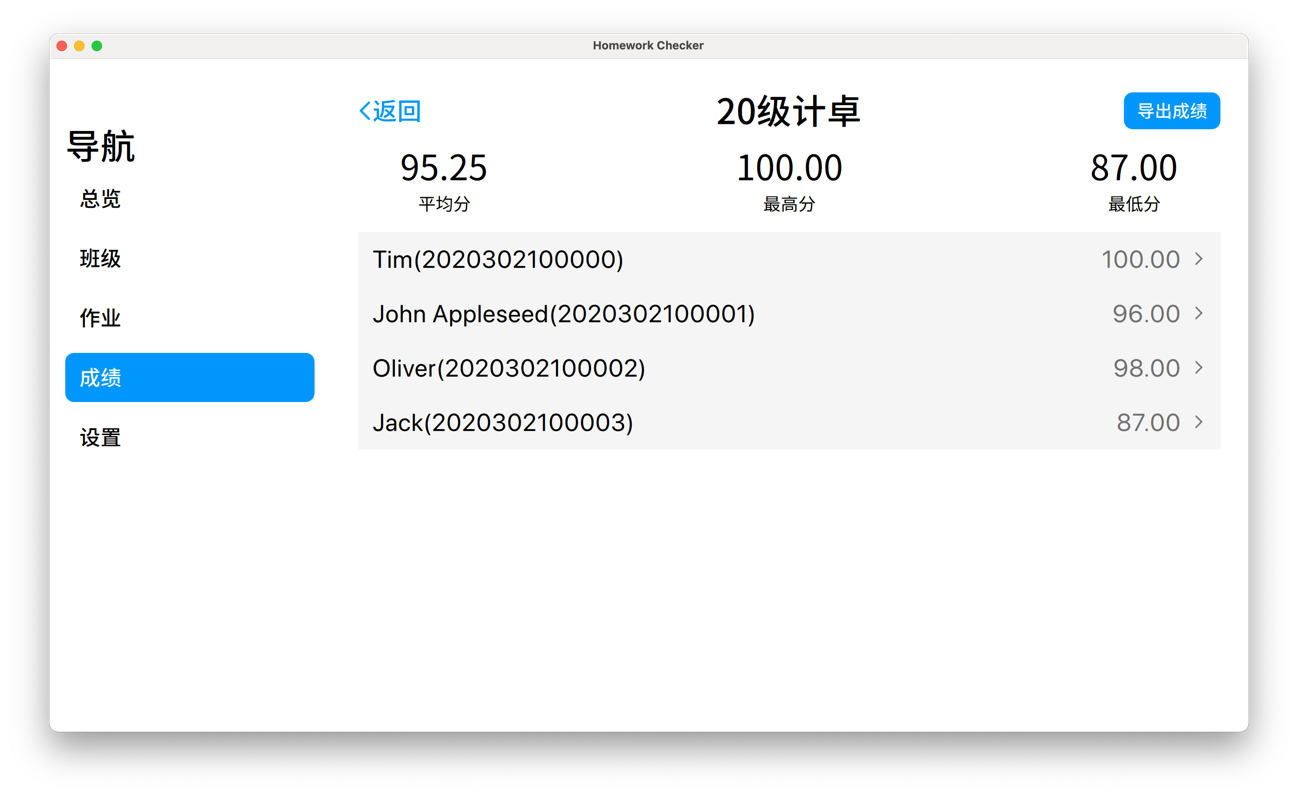
图**6.4** 班级页面和班级信息详情弹窗

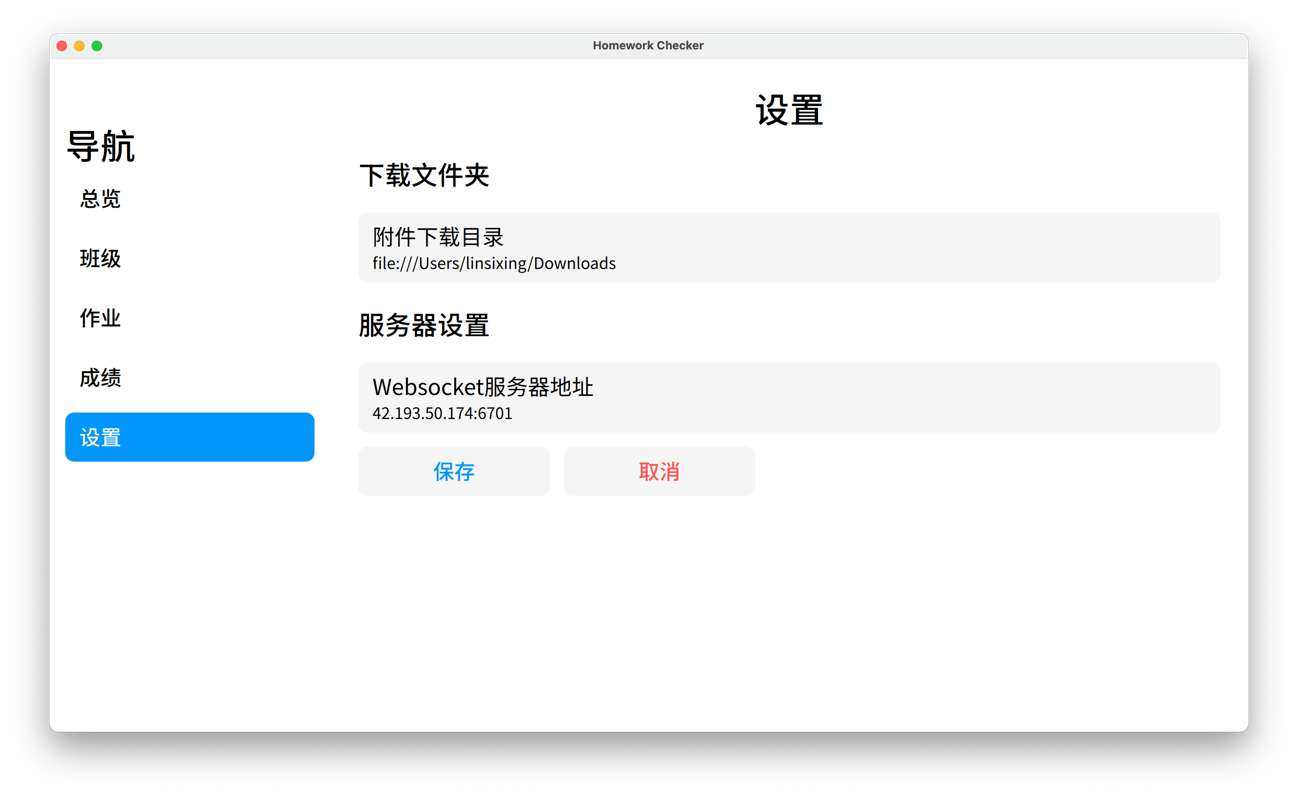
图**6.5** 作业页面

图**6.6** 作业详情页面



图**6.7** 批改页面

图**6.8** 班级成绩页面

图**6.9** 设置页面

## 总结

通过编写该模块，学习了Qt Quick框架的使用方法。

本模块的优点：

* 提供了简单易用的图形界面
* 占用内存小

由于时间和能力限制，本模块还有许多不足之处：

* 未实现阴影效果
* 未实现过渡动画
* 错误提示没有单独的弹窗

# 结 论

经过从设计到开发最后文档编写完成，三位成员一共耗时近两个月，最终相对圆满完成了此项目。项目开发有效代码量超过9,194行（含共计2000余行的qml与markdown文件），注释量超过1,251行，GitHub上commits数达到近180个。

项目功能基本完成，能做到正常而流畅的使用，功能性完整，稳定性较好。

通过此次项目合作，不仅对C++基本语法知识有了更深刻的理解与掌握，还对网络编程、多线程编程、Qt使用、JavaScript语法、跨平台开发以及编译链接过程有了初步的了解。更重要的是培养了良好的发现问题、解决问题、反思问题的能力。大家从一个相对比较弱的基础条件下，逐步共同学习进步。从最开始的遇到问题只会百度翻阅博客，到现在能灵活运用搜索引擎查阅StackOverflow等专业论坛，甚至遇到连网上都很难找到的错误后可以通过仔细阅读官方文档与相关代码自行解决。

但同时由于时间与能力限制，还有很对实用且必要的功能我们尚未开发。在开发过程中也发现前期设计方案具有一定缺陷，没有一定的项目开发经验缺少了准确分析实际问题的能力。

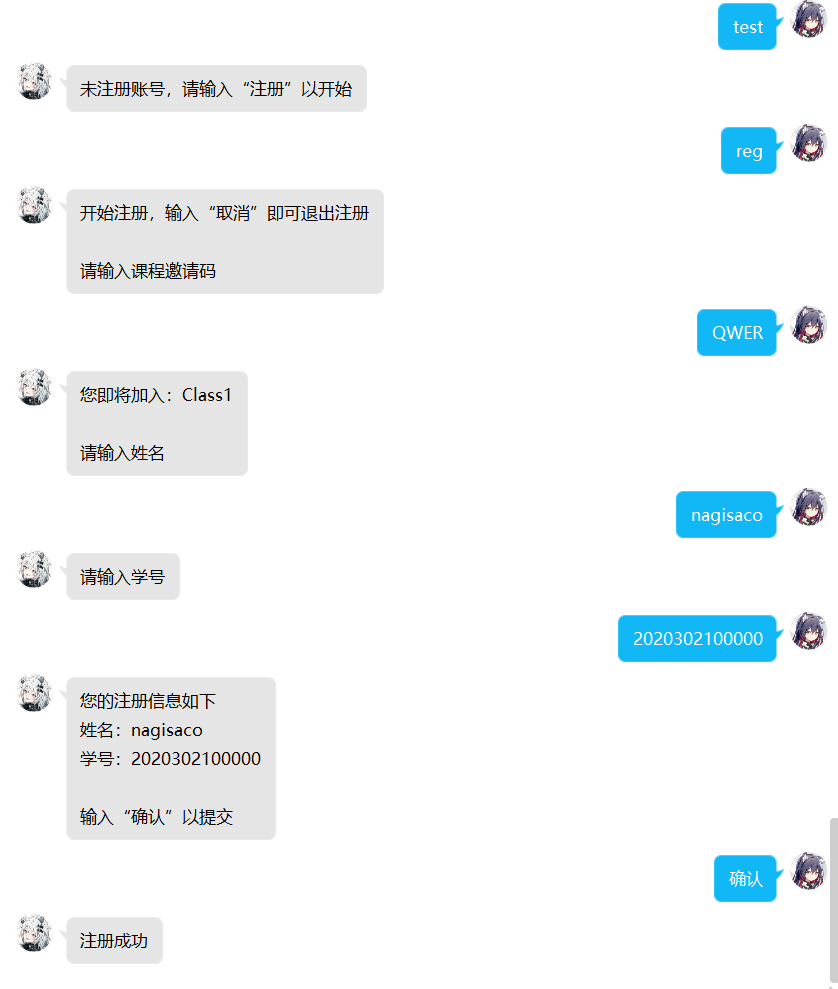
# 参考文献

1. 艾瑞咨询.2020年中国在线教育行业研究报告[EB/OL].http://report.iresearch.cn/report\_pdf.aspx?id=3724,2021-3-22.
2. Peter Lubbers, Frank Greco, Kaazing Corporation.HTML5 WebSocket: A Quantum Leap in Scalability for the Web[EB/OL].http://www.websocket.org/quantum.html,2018-9-28.
3. nlohmann.JSON for Modern C++[EB/OL].https://json.nlohmann.me/,2021-6-26.
4. zaphoyd.WebSocket++[EB/OL].https://www.zaphoyd.com/projects/websocketpp/,2020-7-7.
5. think-async.Asio C++ Library[EB/OL].https://think-async.com/Asio/,2021-5-17.
6. Oracle Corporation.MySQL Community Server[EB/OL].https://dev.mysql.com/downloads/mysql/,2021-5-16.
7. MySQL :: MySQL 5.7 C API Developer Guide :: 4 C API Function Reference [EB/OL]. https://dev.mysql.com/doc/c-api/5.7/en/c-api-function-reference.html,2021-6-1.
8. OpenSSL Software Foundation.OpenSSL[EB/OL].https://www.openssl.org/,2021-6-17.

# 附 录

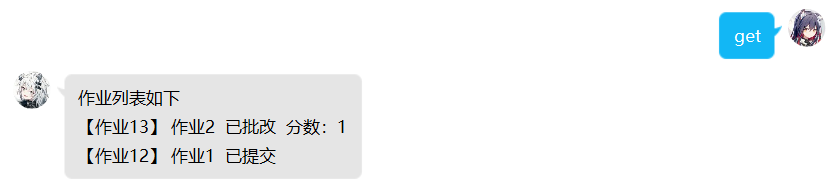
## A 服务端QQ用户交互界面详细截图

##### 注册

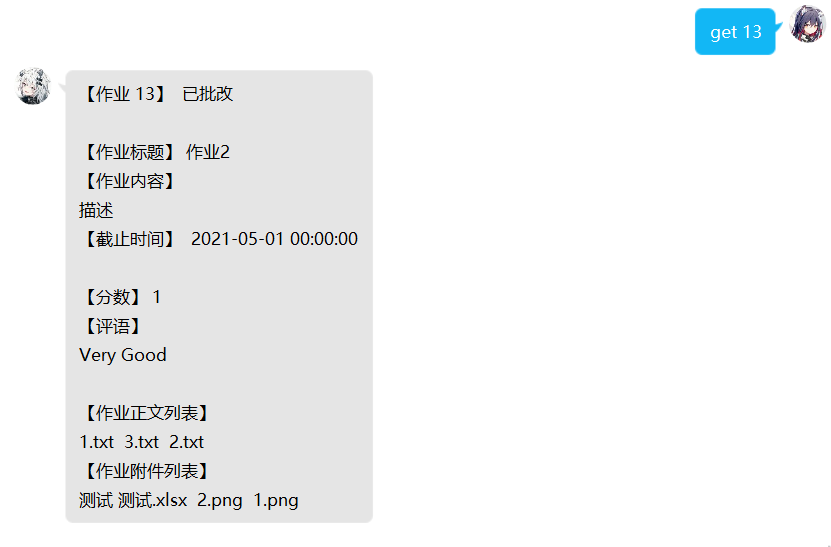


图**A.1** 注册阶段截图

##### 查询作业

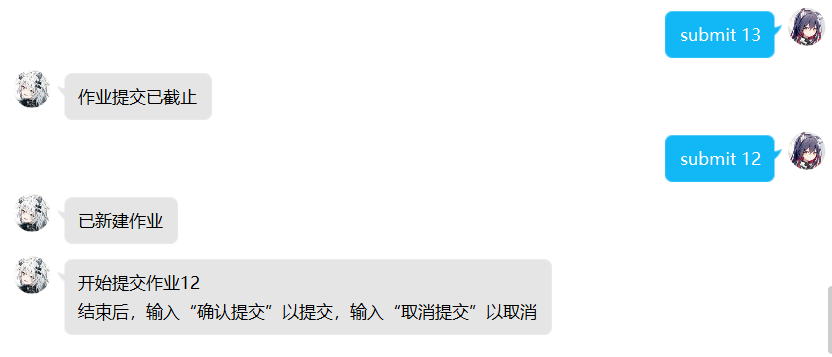


图**A.2** 查询作业列表



图**A.3** 查询作业详情

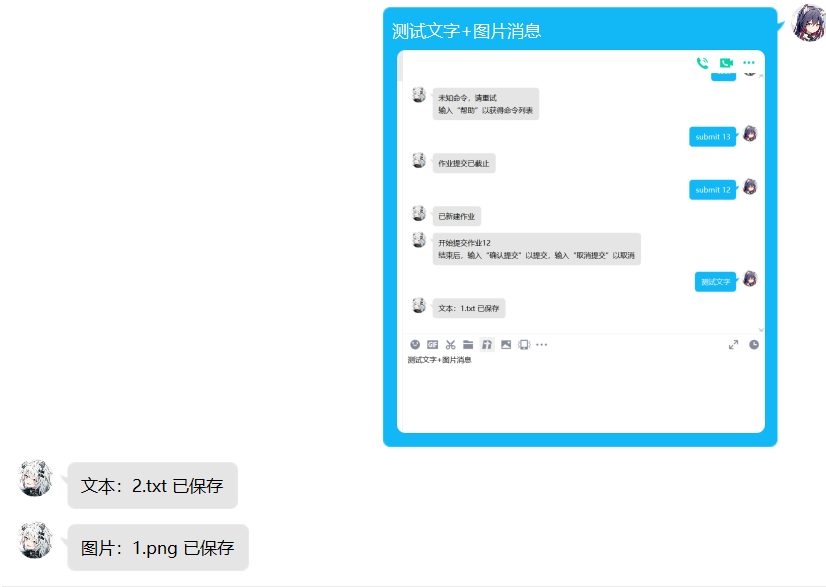
##### 提交作业



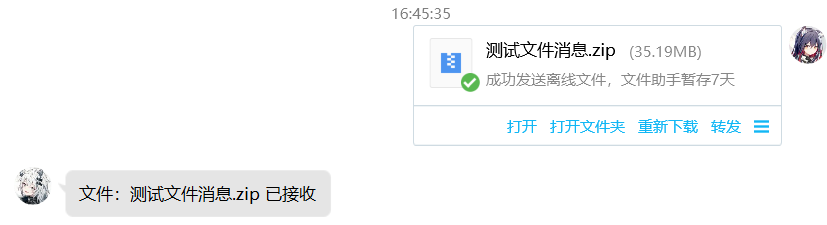
图**A.4** 开始提交



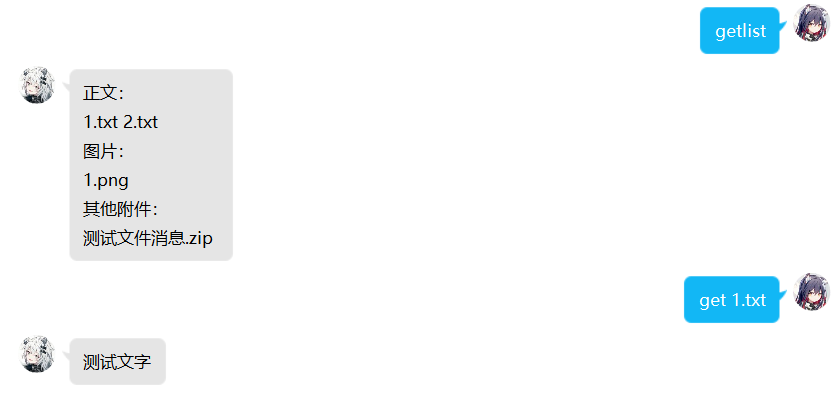
图**A.5** 提交文字消息



图**A.6** 提交图片消息



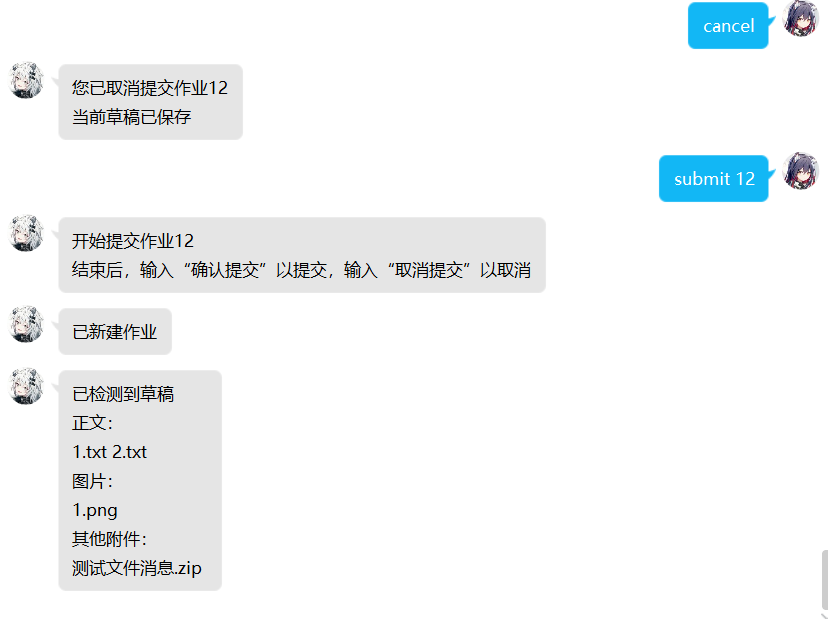
图**A.7** 提交文件消息



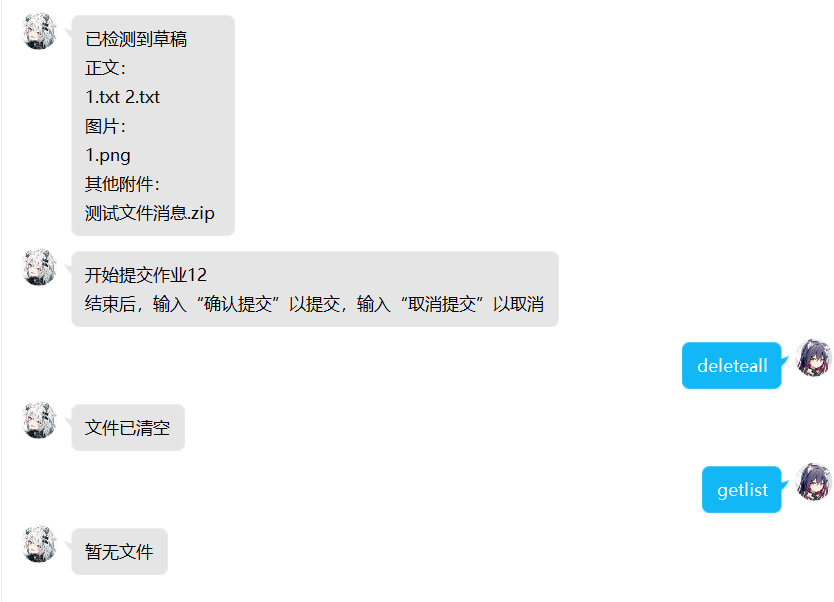
图**A.8** 获取已提交内容



图**A.9** 删除已提交内容



图**A.10** 取消提交并保存草稿

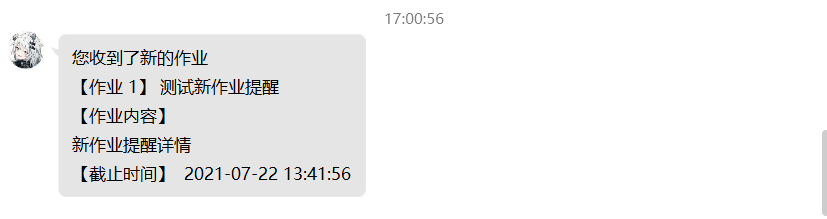


图**A.11** 清空提交内容

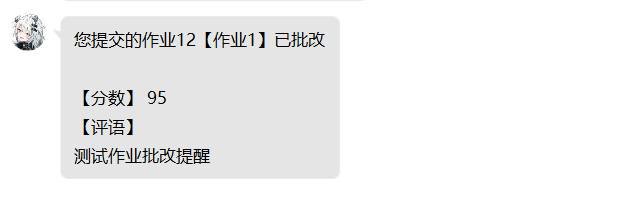


图**A.12** 提交作业

##### 新消息主动提醒



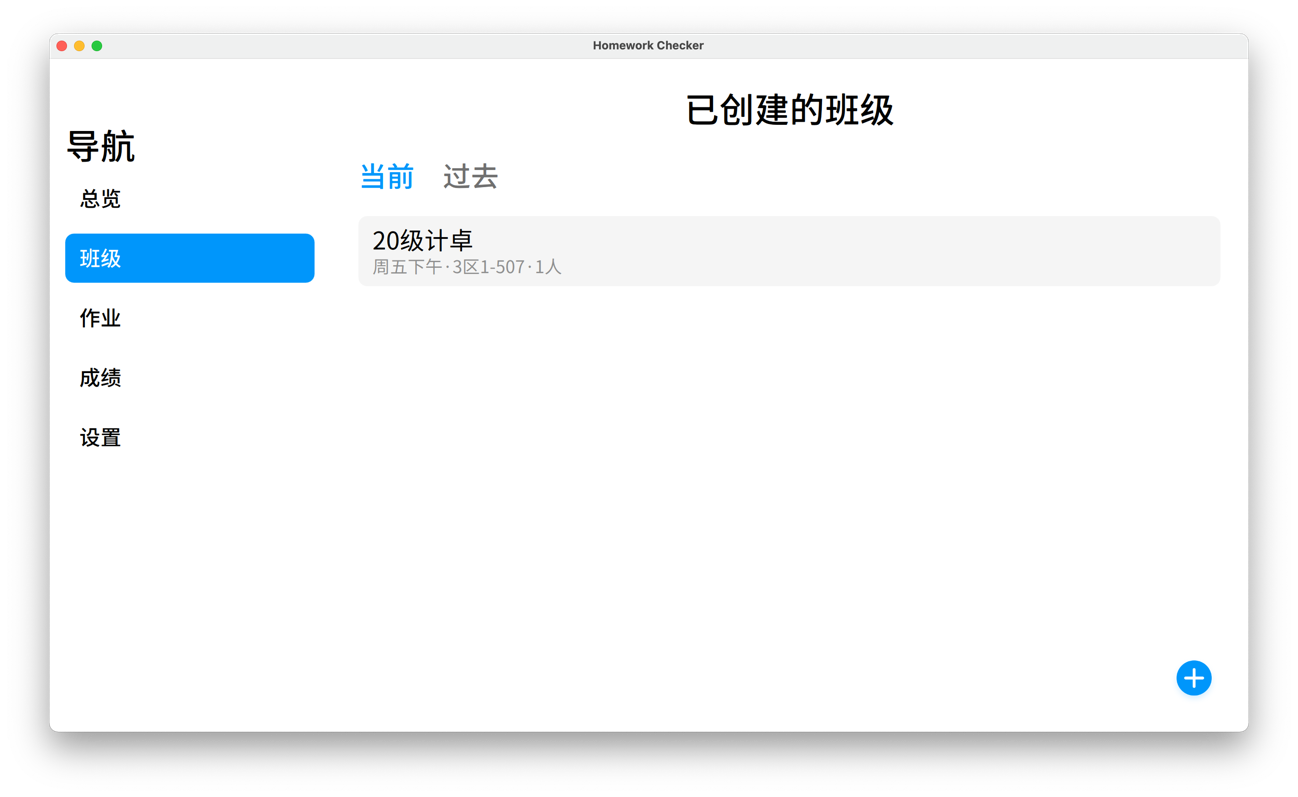
图**A.14** 新作业提醒



图**A.15** 批改完成提醒

## B 教师端用户交互界面详细截图

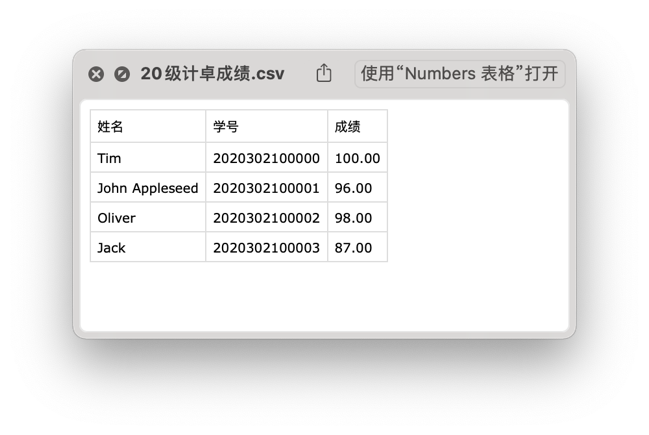
图**B.1** 注册页面

图**B.2** 班级列表

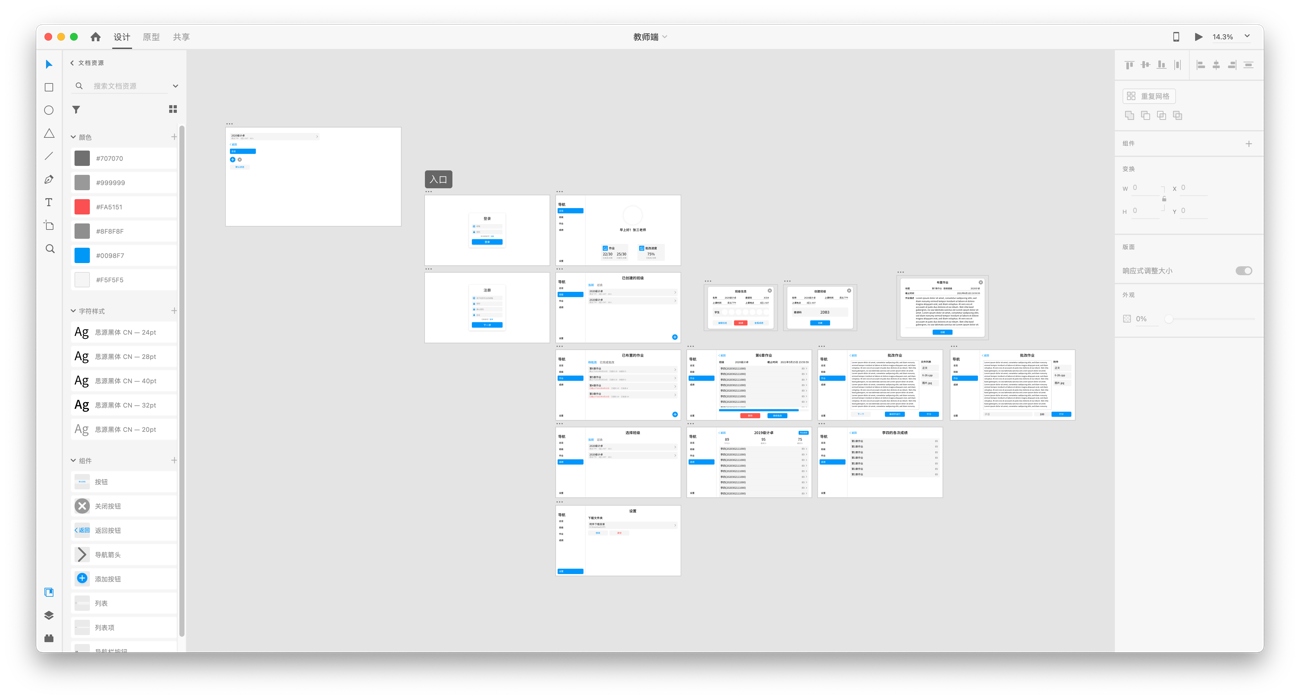
图**B.3** 布置作业

图**B.4** 全部作业已批改完成

图**B.5** 个人成绩



图**B.6** 导出的成绩表格（预览视图）

图**B.7** Adobe XD设计图截图

## C 项目文件树状图

HomeworkChecker

├─ .gitattributes

├─ .gitignore

├─ DataManager 数据管理项目 负责人：林思行

│ ├─ DMTest DataManager测试

│ │ └─ main.cpp

│ ├─ DataManager

│ │ ├─ CMakeLists.txt

│ │ ├─ DBManager.cpp

│ │ ├─ DBManager.hpp 数据库操作函数

│ │ ├─ DMError.cpp

│ │ ├─ DMError.hpp DataManager操作异常类

│ │ ├─ DMUtils.cpp

│ │ ├─ DMUtils.hpp DataManager实用工具

│ │ ├─ DataManager.cpp

│ │ └─ DataManager.hpp DataManager数据管理类库（包含User、Class、Student、Homework、Assignment类）

│ ├─ DataManager.vcxproj

│ ├─ DataManager.vcxproj.filters

│ ├─ DataManager.vcxproj.user

│ └─ DataManager.xcodeproj Xcode项目

├─ Documents 文档文件 负责人：林思行 杨锦荣 伍思烨

│ ├─ DataManager.md

│ ├─ General.md

│ ├─ GUI.md

│ ├─ HomeworkChecker.txt

│ ├─ HomeworkCheckerServer.md

│ ├─ QQMessage.md

│ ├─ tree.md 树状图

│ ├─Analysis 项目trace 负责人：林思行

│ │ └─MemoryTrace.trace 内存泄漏检查结果

│ └─pic 文档图片

│ ├─DataManager

│ ├─General

│ ├─HomeworkCheckerServer

│ └─QQMessage

├─ HomeworkChecker.sln Visual Studio解决方案

├─ HomeworkChecker.xcworkspace Xcode工作区

├─ HomeworkCheckerServer 服务端项目 负责人：杨锦荣

│ ├─ CMakeLists.txt

│ ├─ HomeworkCheckerServer.cpp 服务端程序

│ ├─ HomeworkCheckerServer.vcxproj

│ ├─ HomeworkCheckerServer.vcxproj.filters

│ └─ HomeworkCheckerServer.vcxproj.user

├─ LICENSE GPL-2.0许可

├─ QQMessage QQ消息处理程序 负责人：杨锦荣

│ ├─ Analyst.cpp

│ ├─ Analyst.h 文本处理及分析

│ ├─ CMakeLists.txt

│ ├─ Exception.cpp

│ ├─ Exception.h 自定义异常类

│ ├─ File.cpp

│ ├─ File.h 本地文件管理类

│ ├─ FileInfo.cpp

│ ├─ FileInfo.h 文件信息类

│ ├─ PrivateMessageGetter.h 接收私聊消息类

│ ├─ PrivateMessageSender.cpp

│ ├─ PrivateMessageSender.h 发送私聊消息类

│ ├─ QQMessage.cpp

│ ├─ QQMessage.h QQ消息处理主程序

│ ├─ QQMessage.vcxproj 项目文件

│ ├─ QQMessage.vcxproj.filters

│ ├─ QQMessage.vcxproj.user

│ ├─ Tools.cpp

│ ├─ Tools.h 工具类

│ ├─ WebsocketClient.cpp

│ ├─ WebsocketClient.h WebSocket客户端

│ ├─ WebsocketServer.cpp

│ ├─ WebsocketServer.h WebSocket服务端

├─ README.md 项目简介 负责人：林思行

├─ Setup.sql Mysql配置文件 负责人：林思行

├─ UserInterface 学生端图形化界面程序 负责人：林思行 杨锦荣 伍思烨

│ └─ HomeworkCheckerGUI

│ ├─ ClassPage.qml “班级”页面

│ ├─ DateRectangle.qml 日期选择器

│ ├─ GeneralPage.qml “总览”页面

│ ├─ HomeworkCheckerGUI.pro Qt项目配置文件

│ ├─ HomeworkCheckerGUI.pro.user

│ ├─ HomeworkCheckerGUI.vcxproj

│ ├─ HomeworkCheckerGUI.vcxproj.filters

│ ├─ HomeworkCheckerGUI.vcxproj.user

│ ├─ HomeworkCheckerGUI.xcodeproj

│ ├─ Info.plist

│ ├─ InfoOfClass.qml 班级信息

│ ├─ InfoOfTask.qml 作业信息

│ ├─ LaunchScreen.qml 启动页

│ ├─ LoginPage.qml 登录页

│ ├─ MarkPage.qml “成绩”页面

│ ├─ NewAClass.qml 新建班级

│ ├─ NewAssignment.qml 新建作业

│ ├─ RegPage.qml 注册页

│ ├─ SettingPage.qml “设置”页面

│ ├─ TaskPage.qml “作业”页面

│ ├─ WebsocketClientForApp.cpp

│ ├─ WebsocketClientForApp.h

│ ├─ account.cpp

│ ├─ account.h 全局帐号管理

│ ├─ classscoreviewcontroller.cpp

│ ├─ classscoreviewcontroller.h 班级分数页面控制器

│ ├─ classviewcontroller.cpp

│ ├─ classviewcontroller.h “班级”页面控制器

│ ├─ correctHomework.qml 批改作业

│ ├─ correcthomework.cpp

│ ├─ correcthomework.h 批改页面控制器

│ ├─ fonts 字体

│ ├─ generalviewcontroller.cpp

│ ├─ generalviewcontroller.h “总览”页面控制器

│ ├─ icon.ico Windows应用图标

│ ├─ icon.rc

│ ├─ images 图片

│ ├─ infooftask.cpp

│ ├─ infooftask.h 作业详情页面控制器

│ ├─ main.cpp 启动函数

│ ├─ main.qml 主窗口

│ ├─ qml.qrc qml资源配置文件

│ ├─ scoreWithClass.qml 班级成绩

│ ├─ scoreWithStudent.qml 学生成绩

│ ├─ setting.config

│ ├─ settingpage.cpp

│ ├─ settingpage.h “设置”页面控制器

│ ├─ studentscoreviewcontroller.cpp

│ ├─ studentscoreviewcontroller.h 学生成绩页面控制器

│ ├─ taskpage.cpp

│ ├─ taskpage.h ”作业“页面控制器

│ ├─ untitled.vcxproj

│ └─ untitled.vcxproj.filters

├─ WebsocketClientForApp 教师端WebSocket库 负责人：杨锦荣

│ ├─ WebsocketClientForApp.cpp

│ ├─ WebsocketClientForApp.h 教师端WebSocket类

│ ├─ WebsocketClientForApp.vcxproj

│ ├─ WebsocketClientForApp.vcxproj.filters

│ ├─ WebsocketClientForApp.vcxproj.user

│ └─ main.cpp 测试用主函数

├─ go-cqhttp QQ机器人(Win端) https://github.com/Mrs4s/go-cqhttp AGPL-3.0许可

│ ├─ config.hjson QQ登录配置文件json

│ ├─ config.yml QQ登录配置文件xml

│ ├─ go-cqhttp.exe QQ机器人程序

│ ├─ start.cmd 快速启动命令

│ └─ start.lnk 快速启动快捷方式

├─ lib 临时生成静态链接库

├─ packages 引用第三方库文件

│ ├─ asio Asio网络库 (Header Only Library)

│ ├─ json json库 (Header Only Library)

│ ├─ mysql mysql使用(Win、Mac端)

│ └─ websocketpp websocket库(Header Only Library)

└─ src 资源文件

├─ icon 图标

│ ├─ email.png

│ ├─ homework.png

│ ├─ name.png

│ ├─ password.png

│ ├─ progress.png

│ └─ server.png

├─ ui UI设计图

│ ├─ 学生端.xd

│ └─ 教师端.xd

└─ 作业管理系统产品需求文档.docx 需求分析文档

## D 内存使用

教师端在macOS下使用Xcode的Instrument工具进行了内存Profile，未发现内存泄漏，瞬时最大内存使用量约141MB，平均使用量约106MB。详情可见：

Documents/Analysis/MemoryTrace.trace

图形用户界面, 应用程序

描述已自动生成

图**D.1** 内存泄露测试结果