“C++程序设计与训练”课程大作业

项目报告

项目名称：众包协作翻译平台

目　录

1 平台功能设计.....3
1.1 总体功能描述.....3
1.2 功能流程描述.....3
2 平台结构设计.....7
2.1 模块设计.....7
2.2 类设计.....8
3 平台详细设计.....10
3.1 类结构设计.....10
3.2 数据库/文件结构设计.....13
3.3 界面结构设计.....14
3.4 关键设计思路.....15
3.5 附加功能实现.....16
4 项目总结.....17
4.1 项目不足.....17
4.2 本次大作业的感想.....17
5 相关问题的说明.....18

1 平台功能设计

1.1 总体功能描述

该翻译平台核心的功能是要实现用户能发布翻译任务，并且这个翻译任务在负责人和翻译者的配合下能够顺利完成，任务中扮演各个角色的用户都能获得相应利益。

对于每个用户该平台都具有发布任务的功能，用户只需要指出自己对于该项任务的要求，具体事项可以由负责人和翻译者负责。发布任务之后，该任务变为“招募负责人中”。

用户类里可以有翻译者和负责人，同一个用户在不同的任务中扮演的角色可以不同，同一个用户也可以同时是多个任务的参与者。负责人报名成功后，可以发布更新该任务以招募译者（此时该任务的状态变为“招募译者中”）指定译者人数、最终确定译者，在译者确定以后，负责人向每个译者分发具体的任务（任务的状态变为“翻译中”），在保证公平以及总酬金允许的范围内可以根据不同译者的任务难易程度和译者的能力指定译者的酬金。在译者翻译的过程中，负责人可以随时对译者的译文进行评价。

译者将负责人分发下来的任务完成，并提交最终的译文，负责人将译文整合之后交给发布者，发布者确认译文之后选择任务完成，至此一项任务宣告完成，此任务的状态变为“已完成”。

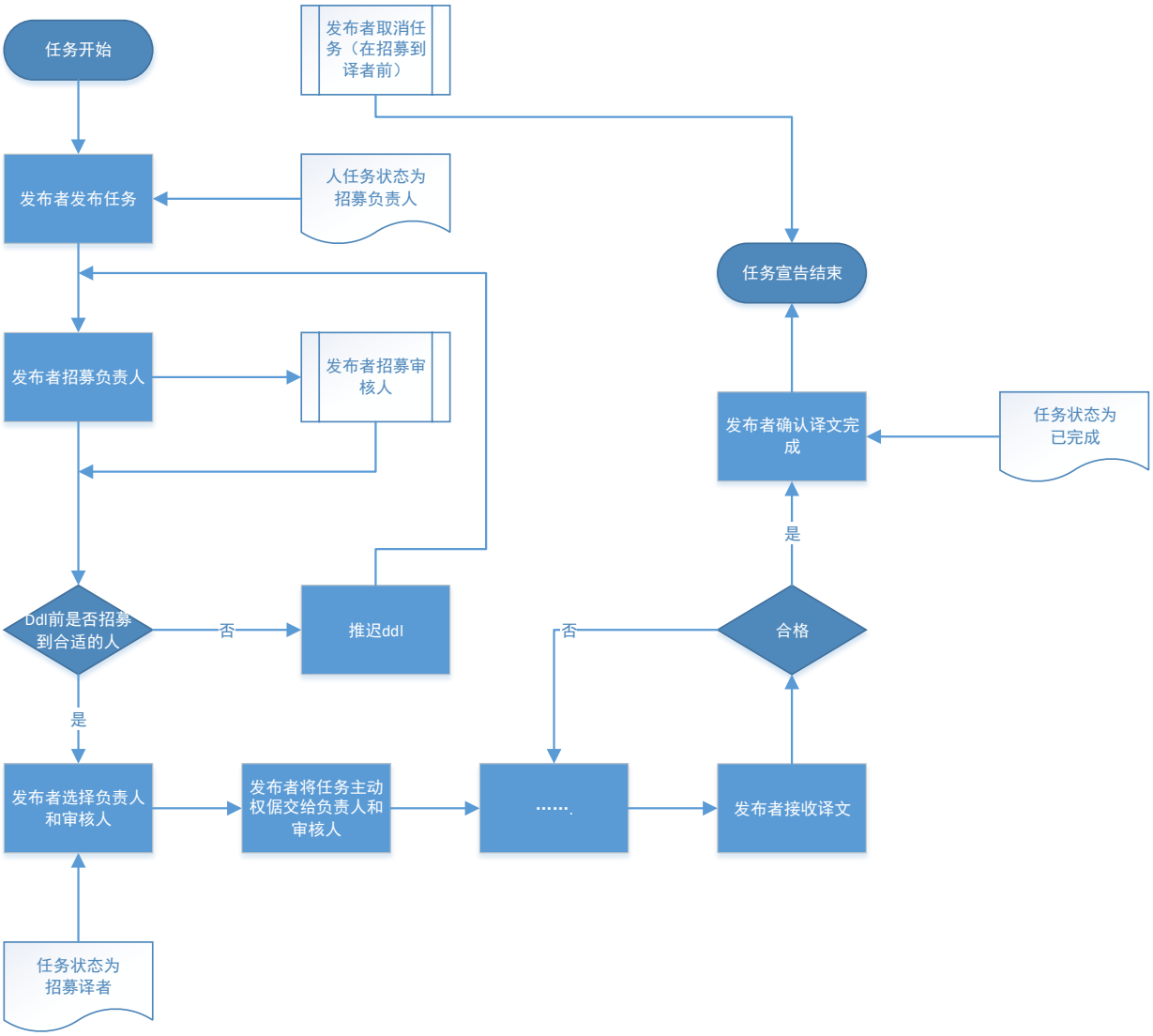
对于任务类而言，它可以储存一项任务的相应属性。任务可以分为发布的任务、负责的任务、翻译者的子任务等，这些任务之间具有组合和继承的关系。

1.2 功能流程描述

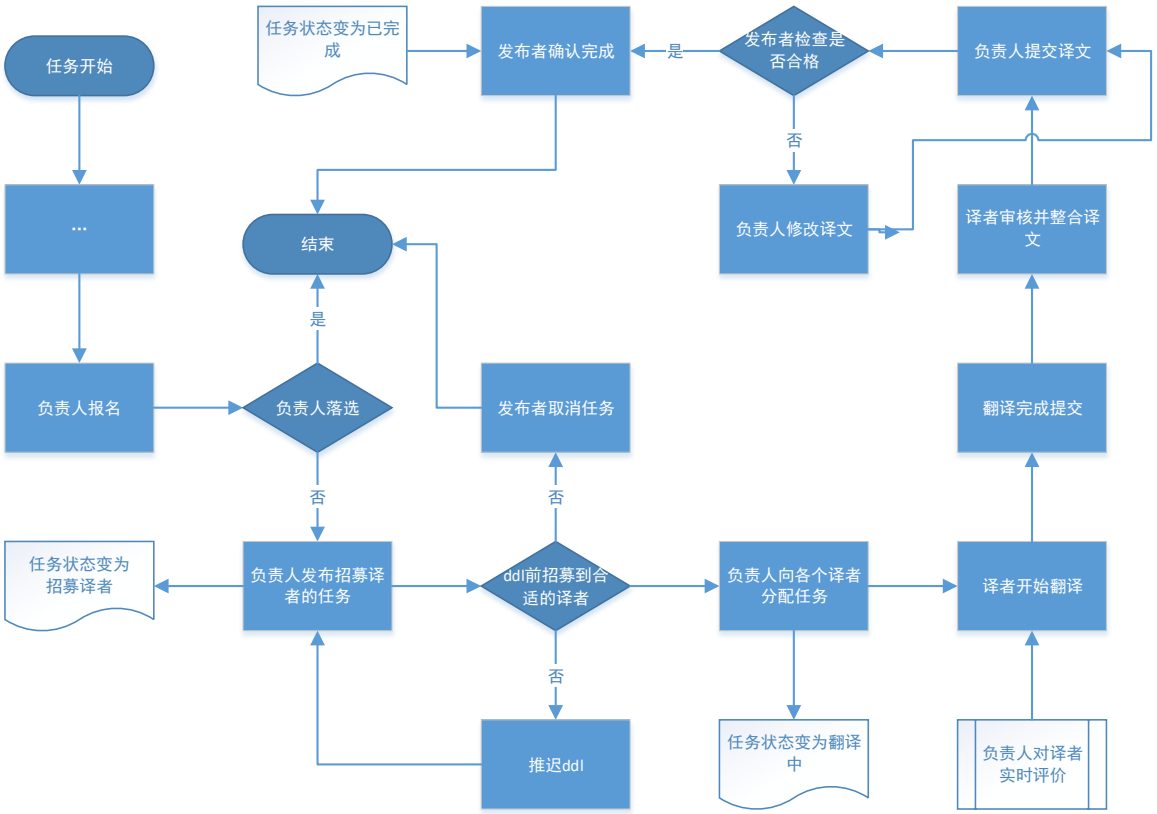
结合上一节的总体功能和翻译任务的进行过程的不同状态描述对应的功能流程，可以用流程图配合文字来说明。

具体的流程图如下：

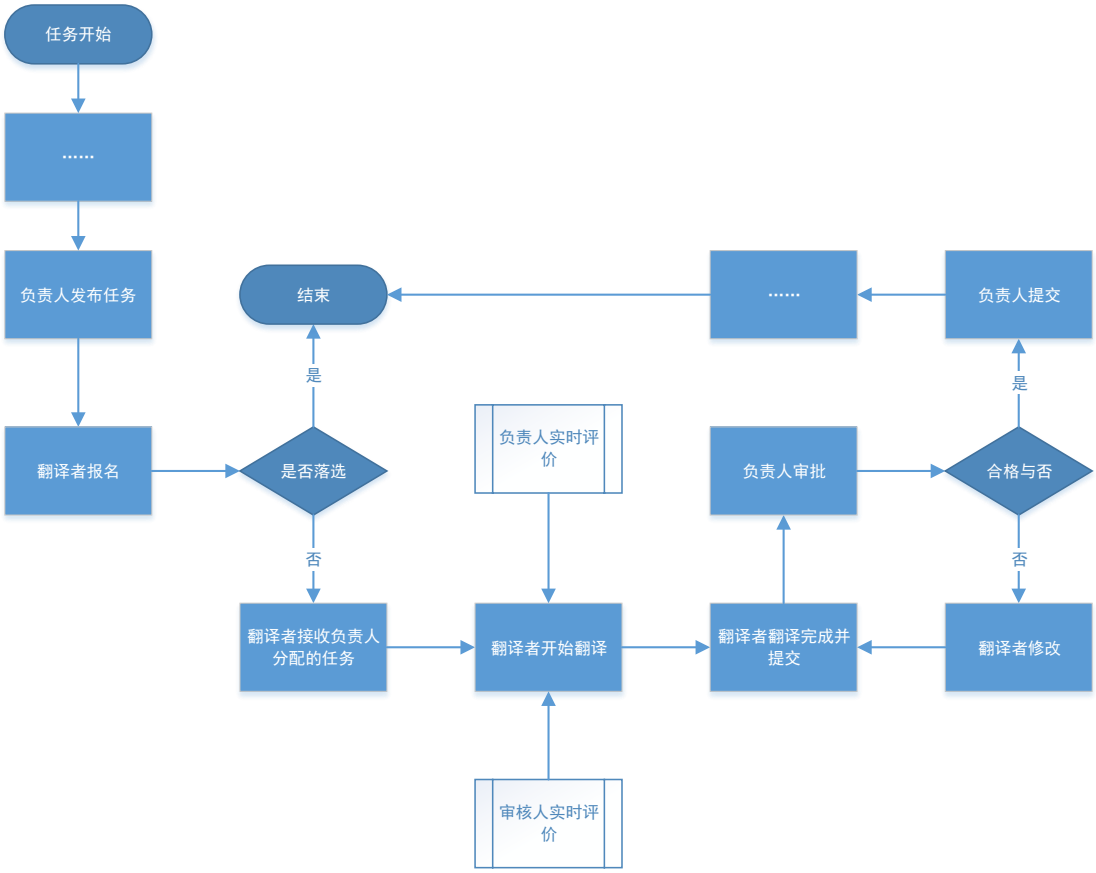
1.2.1 发布者的流程图


任务开始
发布者取消任务（在招募到译者前）
发布者发布任务
人任务状态为招募负责人
任务宣告结束
发布者招募负责人
发布者招募审核人
发布者确认译文完成
任务状态为已完成
是
Ddl前是否招募到合适的人
否
推迟ddl
否
合格
是
发布者选择负责人和审核人
发布者将任务主动权倨交给负责人和审核人
……
发布者接收译文
任务状态为招募译者

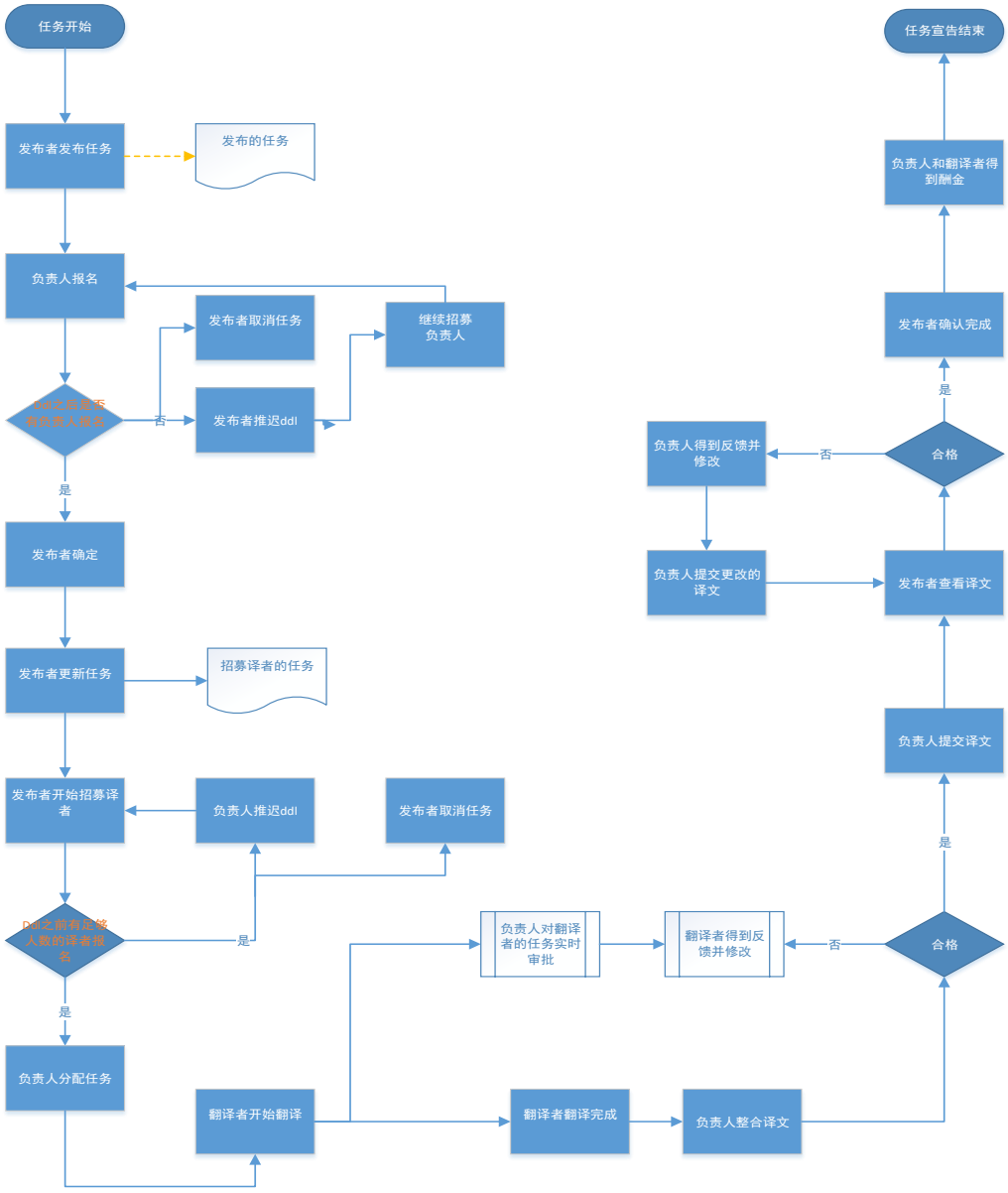
1.2.2 负责人的流程图


任务开始
任务状态变为已完成
发布者确认完成
是
发布者检查是否合格
负责人提交译文
否
...
结束
负责人修改译文
译者审核并整合译文
是
负责人报名
负责人落选
发布者取消任务
翻译完成提交
否
否
任务状态变为招募译者
负责人发布招募译者的任务
ddl前招募到合适的译者
负责人向各个译者分配任务
译者开始翻译
否
推迟ddl
任务状态变为翻译中
负责人对译者实时评价

1.2.3 翻译者的流程图


任务开始
.....
负责人发布任务
结束
.....
负责人提交
是
是
翻译者报名
是否落选
负责人实时评价
负责人审批
合格与否
否
否
翻译者接收负责人分配的任务
翻译者开始翻译
翻译者翻译完成并提交
翻译者修改
审核人实时评价

1.2.4 总流程图


任务开始
发布者发布任务
发布的任务
负责人报名
发布者取消任务
继续招募负责人
ddl之前是否有负责人报名
否
发布者推迟ddl
是
发布者确定
发布者更新任务
招募译者的任务
发布者开始招募译者
负责人推迟ddl
发布者取消任务
ddl之前有足够人数的译者报名
是
否
负责人对翻译者的任务实时审批
翻译者得到反馈并修改
是
负责人分配任务
翻译者开始翻译
翻译者翻译完成
负责人整合译文
合格
否
负责人得到反馈并修改
负责人提交更改的译文
发布者查看译文
负责人提交译文
是
合格
否
发布者确认完成
是
负责人和翻译者得到酬金
任务宣告结束

2 平台结构设计

这是进行复杂软件开发的第二步，即概要设计。此部分需要说清楚整个软件系统包含哪些模块（或功能部件），模块之间的关系和是怎样的；包含哪些主要的类，类之间的关系是怎样的（可以用 UML 类图或对象图表达）。
（此部分的子标题和结构自行拟定。）

2.1 模块设计

整个程序大致分为以下几个模块，分为个人、任务、聊天、系统消息、管理员界面等。

用户模块，该模块包含个人模块、登录模块、注册模块。

个人模块，用户可以查看自己的基本信息，例如用户名、手机号、语言资质证明等，并且可以修改自己的基本信息，有些信息例如语言资质证明需要管理员的同意后才能修改。所有自己参与的任务也会显示在这一模块，用户可以根据自己在每个任务中担任角色的不同来进行不用的操作。

登录模块，在此模块用户可以选择登录普通用户、登录管理员、注册新用户，选择注册新用户会跳转到注册模块。

注册模块，在此模块可以注册新用户，此模块是登录模块的子模块。

任务模块，用户可以查看在平台上发布的所有任务，包含招募译者的任务、招募翻译者的任务、翻译中的任务以及以完成的任务。

消息模块，该模块分为用户之间互发消息的聊天模块以及系统单方向向用户发消息的系统消息模块。

系统消息模块，所有和任务相关的正式消息都会显示在这一模块，在这一模块，用户只能接收系统消息，不能向系统发送消息。系统向用户发送的消息是在用户进行各种对任务的操作时生成的。

聊天模块，用户可以在已注册的用户中请求添加对方为好友，当对方同意后，用户可以和对方互相发送消息。该模块不同于系统消息用户只能接收系统发送的消息，该模块实现了用户之间互通消息的功能，用户之间的交流可以更加方便自如。

管理员模块，此模块代表了该翻译平台，理论上讲，任何一个用户都可以登录这个界面，并对用户向管理员发送的消息进行接收，以及完成后台修改用户积分的操作。用户可以在登录界面输入自己正确的账户名和密码后，选择管理员登录，输入管理员密钥即可登录管理员。

用户对自己参与的任务进行操作的模块，该模块是个人模块的子模块。根据用户的角色的不同，用户可以在发布者、负责人、翻译者、审核人 3 个模块进行切换，并执行相对应的操作。

数据库模块，实现程序从数据库读取和写入数据的操作。

2.2 类设计

主要类的设计如下图的 UML 图所示

考虑到有些类的成员属性有很多，为它们加上接口使用不方便，所以一些类里面声明了友元类，在此不再列出。

3 平台详细设计

3.1 类结构设计

3.1.1 用户模块

User 类

鉴于所有用户都能发布任务，以及成为负责人、翻译者、审核者等对象的功能，该类包含了发布者、负责人、翻译者、审核者的所有属性。根据该类中某些特定属性的不同，用户担任不同的角色。

重要属性：

m_phonenumber，在本程序中，用户的手机号是用户的唯一标识，

m_grade，存储用户的积分，用户的积分是判别用户能否成为翻译者、负责人和审核人的唯一标准。

m_certificate，用户的资质证明是发布者在选择负责人和审核者以及负责人在选择翻译者的重要参考。

m_contacts，储存用户的联系人列表，该列表是根据用户的添加好友和删除好友的操作动态变化的。

该类下辖各个获取以及修改用户属性的函数，提供用户对外的接口。

dlg_register 类

该类实现了注册操作的可视化。

同时，该类还实现了多种容错机制。例如用户注册时的密码匹配问题，检查用户输入的手机号格式是否正确、检查输入是否为空的操作等。

dlglogin 类

该类实现了登录操作的可视化。

user_ui 类

该类实现了用户查看和修改个人信息以及对自己所参与任务的操作的可视化。该类的界面中有一个 tabWidget，包含了个人信息界面和个人任务界面。

在个人信息界面，实现了对用户各种信息进行维护的可视化。此界面有容错机制，例如在用户退出以前自动检测用户的个人信息是否被修改过，如果修改过，会跳出界面提示用户是否还要退出。只有当用户点击“保存并退出”按钮后信息才能保存。

在个人任务界面，用户可以查看自己参与过的所有任务。此界面有 stackedWidget，下辖“我发布的”、“我负责的”、“我翻译的”、“我审核的”三个子界面。如果用户点击“详细信息”，会进入到对该任务操作的界面 user_ui_task 界面。

3.2.2 任务模块

Snopsis 类

该类包含了一个任务简介所需要的信息。是所有任务类中都会包含的类。

重要成员：

m_subtitle，此成员储存发布任务时的任务简介。在负责的任务和发布的任务中是主任务的简介，在翻译任务中是子任务的简介。

m_state，这是一个枚举变量，枚举值有 recruit_c,recruit_t,translating，completed。

Assignment 类

该类依赖于 Snopsis 类，在 Snopsis 的基础上又添加了 m_translation，m_oringin，是所有任务类的基类。

C_assignment 类

负责的任务类，由负责人享有，由 Assignment 公有派生而来。在发布的任务更新之后获得。

R_assignment 类

发布的任务类，由 C_assignment 公有派生而来，这个类包含了主任务的所有属性。

T_assignment 类

翻译的任务类，由 Assignment 派生而来，储存翻译者的子任务所需要的各种属性。

assignment_ui 类

该类实现了平台上所有任务的可视化。该类包含一个 stackedWidget，储存招募负责人、招募译者、正在翻译中、已完成的四种任务。招募负责人招募译者的任务界面用户可以进行报名和查看任务简介的操作，也可以根据任务的各个属性对任务进行排序。在正在翻译中和已完成的界面中用户只能进行任务的查看。

user_ui_task 类

该类实现了用户对自己所负责任务的各种操作，用一个 stackedWidget 集成了“我负责的”、“我翻译的”、“我发布的”、“我审核的”所有任务的操作。

为了满足用户操作任务时的需要，该类依赖于多个子界面的类。

dlg_r_alllco 类

该类是 user_ui_task 类的一个成员。该类实现了发布者查看审核人报名、查看负责人报名、查看审核人报名、负责人查看译者报名以及查看报名者的语言资质证明和报名理由的可视化。在负责人报名的截止期限到了之后，发布者可以从报名的负责人中制定负责人，从报名的审核人中指定审核人。

dlg_update 类

该类实现了负责人更新任务的可视化。在负责人被指定后，负责人应当根据自己翻译的经验，结合发布者的要求向翻译者提出新的要求，更新主任务面向翻译者的简介。在不超出发布者发布的总酬金的基础上，指定翻译者酬金的上限和下限。发布招募翻译者的任务，该任务的状态变为招募翻译者。

dlg_allco_c 类

该类实现了负责人分配任务的可视化。负责人在分配任务时，应该根据子任务的难易程度同时指定翻译者的酬金，每个翻译者的酬金都应该在上限和下限之间，同时所有人的酬金加起来应该等于发布者的总酬金。

dlg_c_view 类

该类实现了审核人和负责人实时评价翻译者译文的可视化。

dlg_delay_ddl 类

该类实现了发布者和负责人推迟任务期限的可视化。发布者只能在发布招募负责人任务阶段修改招募负责人的期限以及在任务未完成时推迟最终的期限，负责人只能在招募译者阶段推迟招募译者的期限。该模块是发布者在原有期限前招不到合适的负责人以及负责人在原有期限前招不到合适的译者时使用的。

以上从 user_ui_task 开始的类全都集成在 user_ui_task 里面，user_ui_task 里面其他的操作直接在原有类里进行实现。

3.2.3 消息模块

message 类

该类是是聊天模块、系统消息模块以及管理员模块的基类。

Validation 类

该类储存所有用户之间申请好友的信息。

message_ui 类

该类实现了聊天功能的可视化。

dlg_system_message 类

该类实现了用户接收系统消息的可视化。系统消息是比较正式的消息，在和用户相关的任务被操作时发出。

3.2.4 管理员模块

该模块比较独立，包含了管理员操作用户的语言资质证明以及修改用户积分的操作。

administrator 类

该类实现了管理员界面的可视化。

admin_view_cer 类

该类实现了用户查看用户的修改语言资质证明的具体信息的可视化。

3.2.5 数据库模块

Database 类

该类实现了对数据库读取和存储数据以及删除消息数据的功能。内置静态变量 s_database，可以满足在不同的文件中用户操作数据库的需求。内置重载 ingressDataBase 函数实现对各种类型数据的导入，以及重载 openDataBase 函数实现对各种数据类型的读取。

3.2.6 集成模块

Widget 类

该类实现了主界面的可视化，主界面是所有界面的父窗口。

3.2 数据库/文件结构设计

描述在数据库（或文件）中存储的结构。

本程序数据库总共有 5 个 table。

Table1：assignments

项目	类型
No	int
date_release	QString
ddl	QString
title	QString
releaser	QString
sub_title	QString
o_language	QString
state	int
salary	float
translation	QString
oringin	QString
contractor	QString
c_salary	float
info	QString
c_ddl	QString
reviewer	QString
contractors	QString
translators	QString
t_ddl	QString
reviewers	QString
translatorss	QString
re_evaluetion	QString
re_salary	float
c_evaluation	QString
min	int
max	int

Table2 messages

项目	类型
date	Qstring
receiver	Qstring
text	Qstring
sender	Qstring
state	int
title	Qstring

Table3 sub_assignments

项目	类型
t_No	int
date_release	QString
ddl	QString
title	QString
sub_title	QString
o_language	QString
language	QString
state	int
No	int
salary	float
releaser	QString
info	QString
t_translation	QString
t_oringin	QString
translator	QString
c_evaluation	QString
r_evaluation	QString

Table4 users

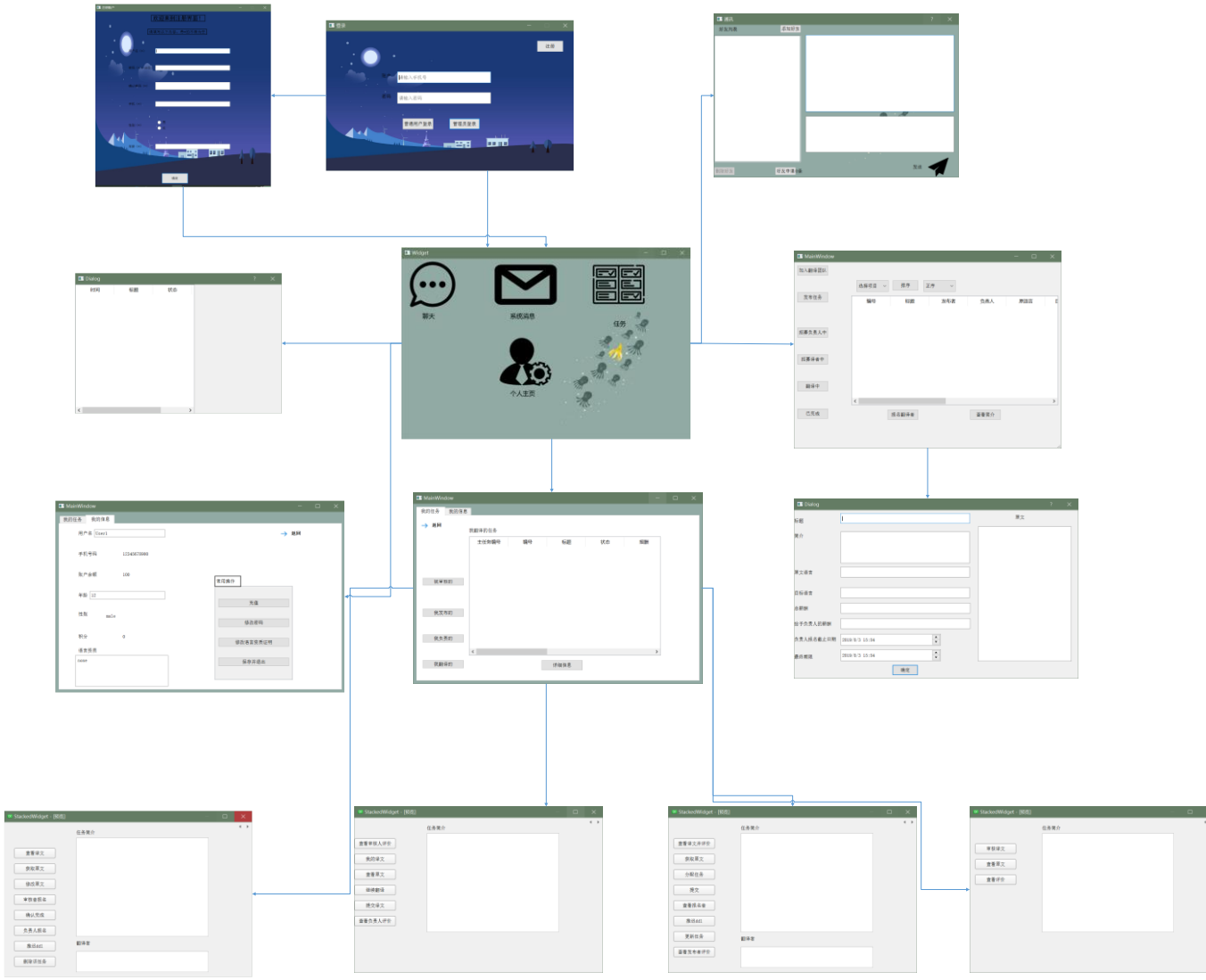
项目	类型
phonenum	QString
username	QString
psd	QString
age	int
balance	float
certificate	QString
sex	QString
grade	int
can_t	int
language	QString
contacts	QString

Table validations

项目	类型
date	QString
sender	QString
receiver	QString
state	QString

3.3 界面结构设计

主要界面跳转关系如下：



3.4 关键设计思路

3.4.1 向数据库中存入数组的操作

由于报名的负责人、报名的审核人、报名的翻译者以及最终参与翻译工作的翻译者的数量都是不确定的，而且一个用户的联系人也是动态变化的，所以不可避免地会需要在数据库中存入动态变化的数组。向数据库中存入数组可以在里面新建一个表格，将数组的元素存入表格中即可。但是考虑到用户和任务的数量不定，使用这种方法会造成数据库的冗余，所以此方法并非最优。经过查阅资料，其实可以利用 QStringList 类的 join 函数和 QString 的 split 函数。该数组在数据库中存储时时 QString 类型，使用特定字符“$”隔开每个数组元素，在使用时先用 QString 的 split 函数转化为 QStringList 类型，对 QStringList 进行动态操作，操作完毕之后使用 QStringList 的 join 函数将“$”重新加入变成 QString 类型，将 QString 类型的变量存入数据库即，从而实现对用的动态操作。

3.5 附加功能实现

3.5.1 用户之间通信

本程序的聊天模块可以实现用户之间互相发送消息的功能，具体如下。

（1）添加好友。

例如，用户 A 想添加用户 B 为好友，用户 A 在获得用户 B 的账号的前提下，向用户 B 提出好友申请。用户 B 登录之后，在自己的聊天模块下有验证消息的按钮，按钮旁边的 label 显示了未读的验证消息条数。用户 B 点击验证消息按钮，在出现的界面中选中用户 A 的申请记录，点击接受可以添加对方为

图 3.5.1 聊天界面

好友，此时好友列表中出现用户 A 的 ID，用户 A 登录后好友列表也出现用户 B 的 ID。

（2）查看历史消息。

用户 A 在好友列表中双击用户 B 的 ID，右上方的界面将会出现用户 A 和用户 B 之间互相发送的消息。

（3）删除好友

用户 A 在列表中选中用户 B 的 ID，此时“删除好友”按钮变为可按，点击之后列表中将会删除用户 B 的 ID，用户 B 的 m_contacts 也会自动将用户 A 的 ID 剔除，同时，数据库中的 messages 也会将用户 A 和用户 B 之间互发的消息删除。

（4）容错机制。

在用户未选中一个好友之前，用户不能发送消息（此时发送按钮 setDisabled 为 true），同样地，验证消息框也有类似的设计。

本模块实现了负责人、翻译者、发布者、审核人之间的信息交流，用户只需要添加对方为好友，就可以和对方互发消息从而进行协商和交流，大大有利于翻译任务的推进。

3.5.2 审核人角色

本程序添加了审核人的角色。

（1）报名审核人。在任务处于招募负责人的状态时，用户可以报名审核人。报名审核人的条件在本程序中设置为积分大于等于 30。用户成功被选为审核人之后，此时该任务的 m_reviewer 的值变为该用户的 ID。

（2）审核译文。审核人审核译文和负责人审核译文的界面是相似的。审核人可以选择翻译者，同时原文框和译文框分别显示该翻译者所翻译的子任务的原文和译文。审核人在评价框对用户的译文可以做出评论，点击“发布评价”按钮之后，审核人的的评价实时更新到用户所负责的子任务的 m_r_evaluation 中去。同时系统向该翻译者发送消息，告知审核人的评价已经更新。

（3）查看发布者的评价。任务完成之后发布者可以对审核人的角色进行评价，审核人在自己的界面点击查看评价。

3.5.3 管理员角色

本程序中的管理员角色亦即平台角色。功能如下：
（1）审核用户提交的语言资质证明。用户提交的语言资质证明有两种，一种是申请加入翻译团队时的初始语言资质证明，另一种是修改语言资质证明时的请求修改的语言资质证明。管理员在审核完毕并同意后，应该可以选择修改用户的积分。系统自动向用户发送 message，提示语言资质证明修改完成。管理员也可以拒绝用户提交的修改语言资质证明请求，此时管理员需要说明自己拒绝的理由，具体拒绝的理由将会封装在 message 类的 m_text 里面发送给用户。、
（2）修改积分。管理员只能修改向自己发送修改资质证明请求的用户的积分，因为只有这种情况下修改积分才是有意义的。管理员一次可以修改的积分范围为 -10 到 100。管理员在未选中对象时，系统会提示管理员选中一个对象。

3.5.4 评价机制

本程序不仅添加了审核人和负责人向翻译者评价的机制，也添加了发布者向审核人和负责人评价的机制。在发布者确认完成时，会出现一个界面可以填写对负责人和审核人的评价。

4 项目总结

4.1 项目不足

1. 本项目的图形界面做的并不美观，除了在某些界面加入了背景图片和图标外，大部分界面都是 Qt 原有的控件。
2. 代码书写不是特别规范。例如，在双目运算符两端一般要加上空格，但由于本人之前用 vs2017 编程，没有类似的习惯，所以很多地方显得有些拥挤。
3.某些设置不是很友好。例如，在负责人分配任务时要指定每个翻译者的酬金，由于项目总酬金一定，以及负责人和审核人的酬金都是确定的（有审核人时审核人的酬金时负责人酬金的 80%），负责人需要根据自己的计算来指定酬金。
4.编码过程中对指针的安全性问题没有做太多预防措施，所以导致有时程序会莫名其妙的崩溃。现在看来，使用指针前检查指针的合法性和安全性是十分关键的一步。
5. 另外在某些部分，由于我对代码的理解不精，部分代码显得比较冗余。

4.2 本次大作业的感想

本次 C++大作业是目前为止我遇到的最硬核的作业之一。个人认为，由于

C++大作业会使用到图形界面编程，以及数据库的操作，而这些内容在课本上并没有，而网络上的资料又浩如烟海，所以特别考验完成者查阅筛选资料和快速应用知识的能力。此外，各种用户、任务等类的设计、界面的设计全都需要自己独立完成，而设计好类和界面是完成大作业最关键的一步。

5 相关问题的说明

本程序使用 Qt Creator 4.9.1（Enterprise）编写，采用 Desktop Qt 5.13.0 MinGW 64-bit 的构建组件编译，数据库使用的是 Qt 自带的 QSqlite。除此之外，对编译环境没有过多要求。