

需求分析说明书

第一章 引言

1.1 编写目的

建立一个在线英语中文交互式学习系统，该系统基于视频、音频及文本为用户提供沟通途径。用户通过该系统实现英语和中文的学习以及结交更多的朋友。中美两国的用户通过使用该软件可以方便地实现语言学习和交友目标。

1.1.1 申请人

无论身处国内国外都可以申请账号，申请人不限国籍。申请人在登录之后，可在网上寻找与自己需求相匹配的人进行交流学习。比如，一位中国大学生想去斯坦福大学求学，那么他通过输入条件后，寻找对斯坦福大学比较了解的美国人，然后利用该软件开始沟通。在这种情况下，这位中国大学生既学习了英语，还开阔了眼界。

1.1.2 赚虚拟币的人

用户在经过注册、登录等一系列步骤后，即可在网上寻找自己的教学水平可以满足要求的任务，找到对自己有学习请求的申请人，然后申请人向该用户提供相应数目的虚拟币，二者即可达成交易。这样一来，此用户不仅可以在自己闲暇之余加深自己对知识的熟练程度，还可以把自己的学识传授给更多的人，赚取金币，而且可以结交更多的朋友。

1.1.3 运营商

运营商需要开发一个在线英语中文交互式学习系统，以满足中美双方用户的学习需求，促进两国英语中文的学习交流。另外，运营商提供虚拟金币购买服务，用户可以通过多种方式购买金币，为运营商带来收入。

1.2 项目背景

随着计算机网络的蓬勃发展，网络视频系统已被广泛应用于各行各业，利用网络方便世界各地人们的沟通交流已成为一种趋势。科技的进步也使得人们的生活发生了巨大的变化，人们在家就可以观看千里之外的东西。因此，开发一个在线中英文交互式学习系统，顺应着时代的发展现状和时代的需求。

1.2.1 国外现状

随着互联网的广泛普及，人们普遍具有了基本的操作电脑的能力，同时多媒

体传播系统也不再局限于在网络上传输声音信息：视频会议、可视电话等多种网上信息传播途径蓬勃发展。

在国外，在线实时直播系统作为现代化传播信息的一种主要应用手段，已经处于成熟时期。另外，网络直播系统正向智能化和虚拟现实等多方向发展。

在美国，现场教学体系快速发展，但它的发展依然存在一些问题：例如，在教学过程中，透明度、便于使用和交流是一些常见的需要改进的方面。许多实时系统多个课程提供一个单一的源接口，允许方便地访问、使用简便。由于当地的技术资源，开发工具和适应当地条件的性质，必须考虑到传输平台是这些技术问题，因为这些直接影响教学活动的行为。

1.2.2 国内现状

目前，在国内，已经有很多企业在做网络直播产品的研发。这些视频直播类产品具有较好的应用前景，然而在功能方面却略显单一，尤其缺乏对移动设备的支持。另外，在系统的兼容性方面还有待加强。

第二章 项目介绍

2.1 项目目标

服务目标：该项目主要致力于跨国交友互助学习，为用户提供与国际友人的交流机会，使各国用户能够通过该平台找到相互学习的伙伴，在交流中学习另一门语言，相互解答，相互纠正，通过对话提高彼此的语言水平，因此该平台应能够提供广泛的国际交友资源和语音视频聊天服务，从而满足用户的学习和交友需要，同时应提供友好的界面，为用户带来良好的使用体验。

功能目标：由于该项目致力于使用户通过聊天交友达到学习目的，因此该项目应该具有实时文本对话、语音通话、视频通话功能；为了为用户找到最合适的学习交友资源，该平台应具有良好的匹配功能，根据用户需求推荐最合适的学习交友资源。

使用目标：用户在使用该平台的过程中，可以通过线上付款购买虚拟币或者包月服务，从而获取学习资源和实时文本对话、语音通话和视频通话服务。

2.2 项目适用用户

该项目适用于想通过交友进行语言学习的用户，用户可将该平台用于外语启蒙、口语听力提高等各类语言学习中，该项目适用于各年龄、各行业想进行语言

学习的用户，适用范围广泛。

2.3 可行性分析

首先，从技术层面来说，目前开发该平台项目的计算机硬件已经非常普及，因此该平台开发的硬件要求是可以满足的，如今社交平台众多，视频通话技术应用广泛，该类平台的开发技术也是可以掌握的。

其次，从该项目适用范围来讲，如今国际间的交流越来越广泛，对外语的掌握越来越重要，因此人们对外语学习的需求也越来越多。对于语言的学习，交流是最直接，也是最有效的学习方式，通过与国际友人的直接交流，可以规避外语学习中的误区，也可以学习到最纯正的外语语法，同时也能够最大限度提高口语和听力水平。然而，受地域限制，想要随时随地找到国际友人彼此进行语言学习并不是容易的，出国进行语言学习的成本又是巨大的。故而该项目正是能够为广大用户提供一个广阔的国际交友学习平台，用户可以轻松找到同样想进行语言学习的国际友人进行线上交流，这极大的迎合了广大用户的需求，因此该项目拥有广泛的潜在用户，从这一点上来说是可行的。

最后，从该项目的盈利方面来看，用户需要通过购买虚拟币来获取该平台的学习资源和一对一实时视频通话服务，也可以购买会员获取特权服务，该平台可以主要通过售卖虚拟币达到盈利目的，因此从盈利方面来看该项目也是可行的。

综合以上三方面，我们可以得出该项目可行的结论。

第三章 模块功能

本系统是一个集在线教育与社交于一体的平台，用户可在本平台基于文本、音视频的一对一交流过程中寻求所需帮助或者为需要帮助用户提供帮助，从而达到相互学习的目的。

本平台具体的功能模块主要有：注册登录模块、用户模块、充值模块、后台管理模块（用户信息记录）。

用户使用本平台按照一定的流程进行：用户注册登录，浏览平台推荐的信息，匹配用户寻求帮助，通过一对一连线帮助需要帮助用户，通过搜索查找需要的需求和服务信息等。

用户按照信息的完善程度分为：游客、注册用户、付费用户和解答人。游客注册登录成为注册用户，注册用户具有查看请求、查看和修改个人信息的功能，

付费用户可以发布请求、进行历史记录查询，付费用户进行个人能力认证后可称为解答人。

3.1 注册、登录模块

3.1.1 用户注册

用户进入社区主页面后，对于第一次登录的用户来说，首先需要注册，单击“注册”按钮即可进入注册界面，注册完成后返回登录界面。游客可以浏览网站内容，无法使用/被使用相关服务。

注册界面用户输入相关信息注册信息，系统后台验证输入信息的合法性，如出现违规信息返回注册界面再次输入相关信息，若信息审核通过，提示用户注册成功，向数据库用户信息表中插入一条记录，并跳转至登录界面。

3.1.2 用户登录

用户注册成功后或用户访问网站首页，点击登录，跳转到登录页面。登录页面要求用户输入用户名，密码，以及验证码，用户提交信息后，系统索引数据库查询是否存在此用户。若否，则提示用户不存在；若存在则查询密码是否一致。若否，提示密码错误，并跳转至登录界面；若正确则提示登录成功，跳转至首页。

3.2 用户模块

3.2.1 查看用户信息

用户进入个人信息界面后可以调用信息查询功能。系统根据用户的查询条件向数据库用户信息表提取相关信息如：头像、昵称、年龄等信息按照要求进行显示。

3.2.2 修改用户信息

本模块用于用户信息如：基本信息、密码、绑定邮箱、手机等信息修改。用户输入相关信息实时验证合法性，合法提交信息，更新用户信息表。否则，修改无效。

3.2.3 一对一聊天

用户可以在系统中向其他用户发起一对一文字或者视频聊天。

3.3 后台管理模块

3.3.1 内容管理模块

本模块主要对平台页面显示的内容管理，排序等以及审核用户日常操作等行为，对于不良或者不合法的信息进行管理。

3.3.2 聊天室管理模块

本模块搜集热门话题以聊天室的形式展现,以及审核用户发起的创建聊天室请求信息合法性。

3.4 充值模块

当用户使用需要充值才能进行的功能时,跳转至充值界面,并展示充值说明。

用户选择具体的月费/年费后提交,待确认用户是否了解充值规则以及对充值金额进行确认后,如充值成功,向用户会员数据库插入一条数据,跳转至个人账户页面,显示当前剩余时间,否则返回充值页面。

第四章 非功能需求

4.1 运行环境

本系统应可运行于: Window 系统中。

本系统成型后为一款在线视频学习交友社交平台,由于 Windows 平台在各类用户群体中普及度最广,且使用障碍较少,同时在设计初期,开发者更容易对系统的各方面进行分析和测试,以改进设计及需求,所以在本在线视频学习交友社交平台的设计及开发初期,我们先着手于 Windows 平台的系统设计及开发,同时对该系统的各方面需求及设计漏洞进行完善。

待该系统在 Windows 系统中设计完善且开发可行后,我们再着手其他平台的后续工作。

4.2 性能需求

使用 PC 端浏览器对网站进行访问时,忽略本地机器性能影响的情况下,各页面的打开速度小于 10s。

包含查看、修改用户信息,登录注册等功能在内的,需要对数据库进行增删改查的情况,使用视频通话时建立连接无障碍。

4.3 安全性要求

本系统的安全性要求主要包括用户的隐私安全性,用户在注册、实名制等过程当中会提交很多隐私信息,包括用户名、密码、性别、年龄、身份证信息、真实照片等内容,如果这些信息泄露,会对用户造成很大的影响,所以我们在设计系统时,应该把用户信息保护系统作为安全性要求的核心加入到系统中。

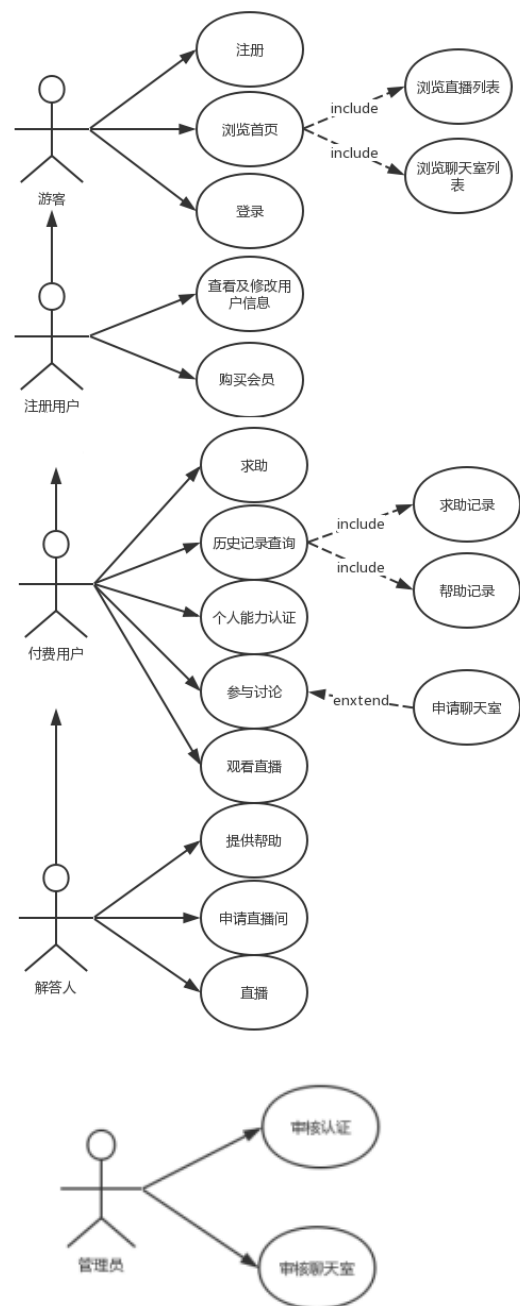
在存储用户隐私信息时,我们可以对相应信息进行加密或者消息摘要处理,

使得存储在数据库中的信息为处理后的密文，而不存储相对应的明文信息，这样可以在一定程度上保护用户的隐私。

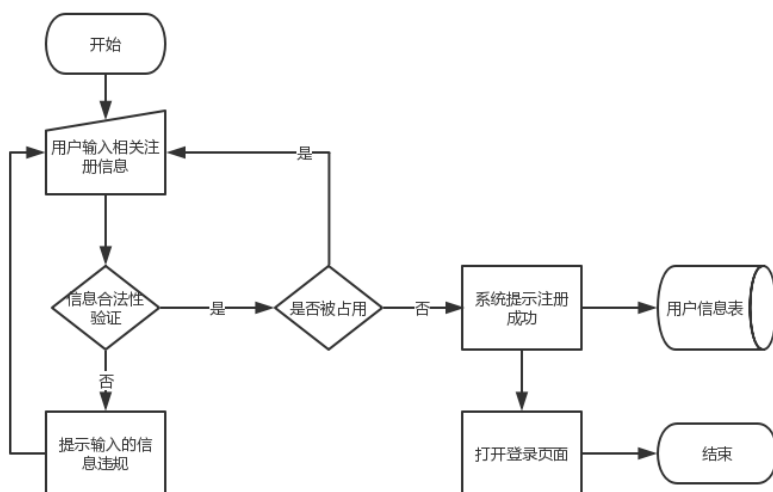
同时，我们也应该在系统设置中加入用户隐私选项，让用户自行设置自己的哪几项信息可以对外可见，这样就可以满足不同用户在交友社交时的不同需求。

第五章 用例图及流程图

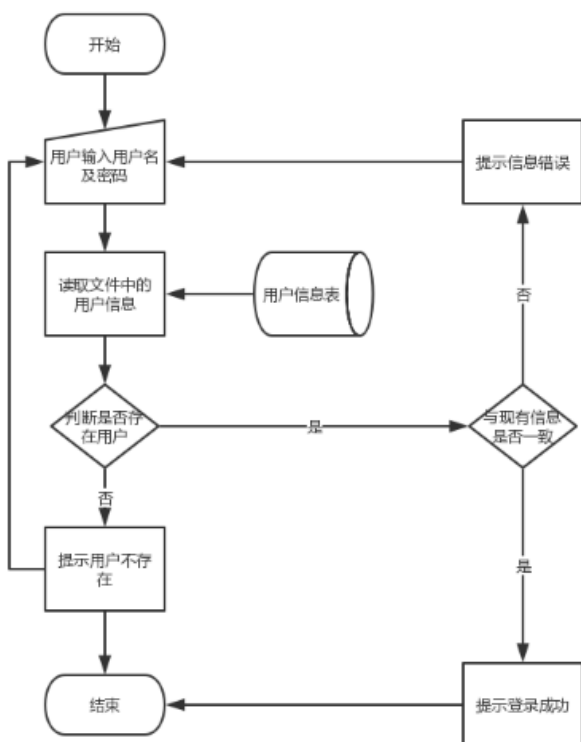
5.1 用例图



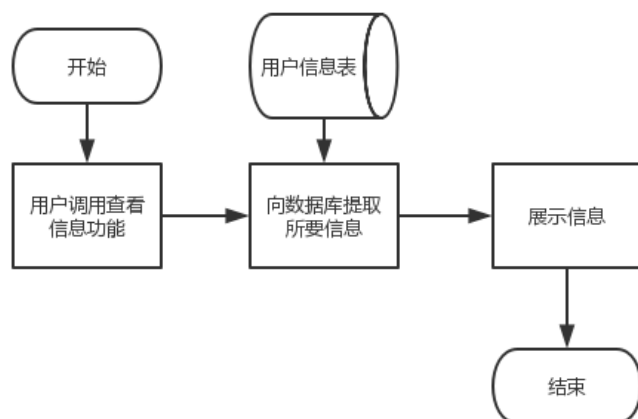
5.2 注册



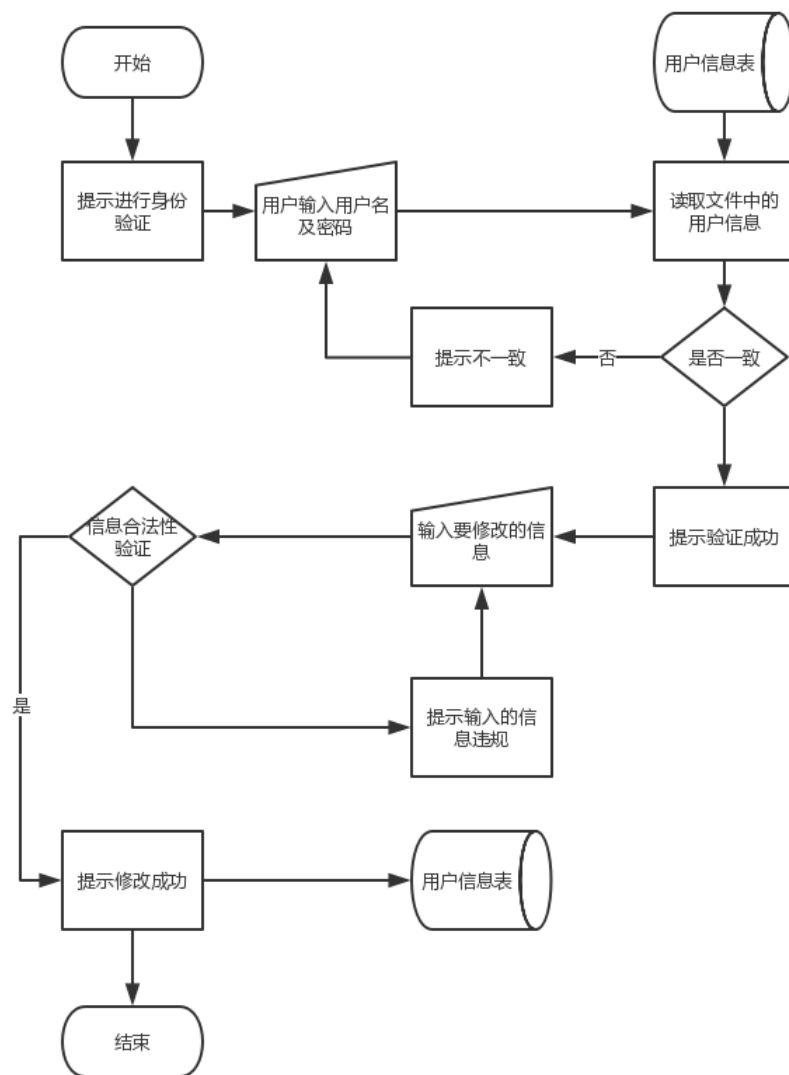
5.3 登录



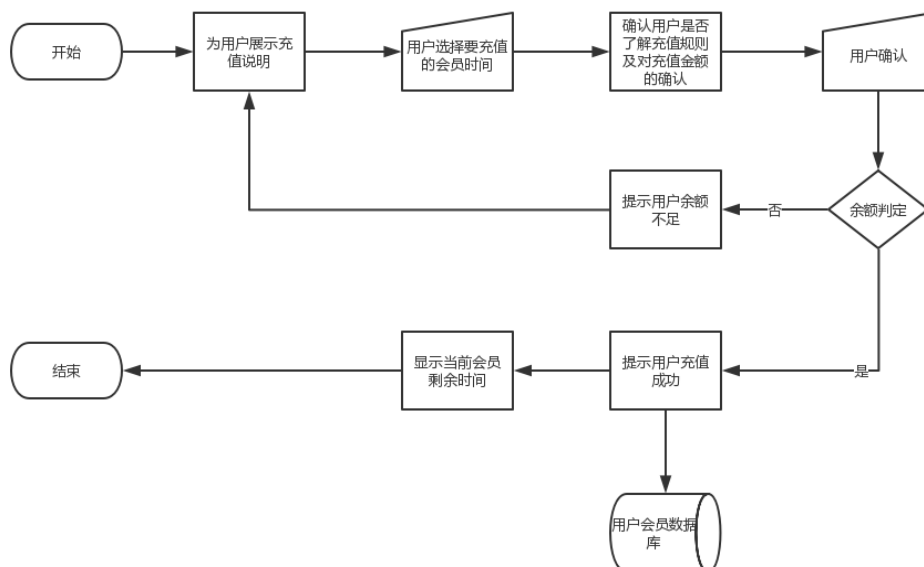
5.4 查看用户信息



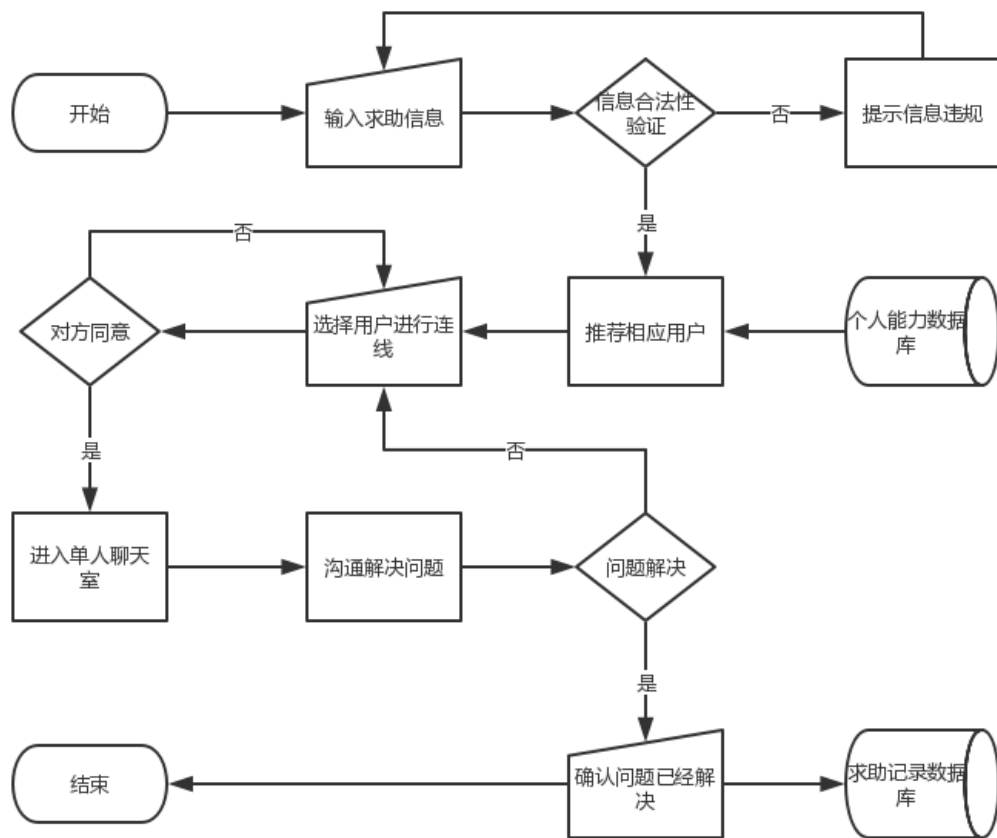
5.5 修改用户信息



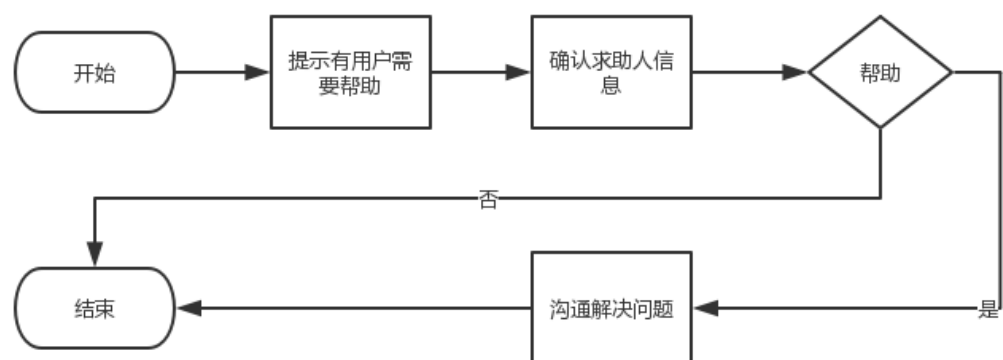
5.6 购买会员



5.7 求助



5.8 提供帮助



第六章 用户使用说明

6.1 用户类型

6.1.1 游客

所有未注册或已经注册但未登录的用户访问本系统时，身份都是游客。

游客可以进入网站首页浏览用户列表、浏览聊天室列表等，也可进行注册、登录功能。

游客仅能浏览首页推送列表，但不能观看具体内容，观看具体内容需要进行注册或登录。

6.1.2 注册用户

未注册用户需要提交相应的注册信息如账号、密码等来完成账号注册，注册成功后可以登录本系统。完成注册的用户即成为注册用户。

注册用户可以查看、完善或者修改用户基本信息、账户信息等。

注册用户可以进行会员充值变为会员及付费用户。

6.1.3 会员（付费用户）

注册用户选择充值期限完成充值后即可成为会员（付费用户），会员可按月充值或者按年充值。会员享有若干非会员所不具备的权利。

会员可以在个人中心查看自己的会员期限、支付记录等。

会员可以和系统中的其他用户进行匹配，用户完善个人信息，并输入求助的检索条件，系统会对系统中其他用户进行匹配，该用户可以向匹配用户发起一对一连线。

会员可以查询自己在本系统中的各项活动记录，包括向他人求助记录和帮助别人解答的记录等。

会员可以在个人中心填写相应的个人能力信息并且提供相应的证明材料来完成个人能力认证，完成个人能力认证之后即可帮助他人解答问题。

会员可以申请与其他人进行在线交流和讨论。

6.1.4 解答人

会员完成个人能力认证，经系统确认后，可成为解答人。

解答人可以向他人提供一对一帮助。

6.2 用户使用流程

6.2.1 游客

用户进入系统后，在未登录状态下可显示热门聊天室和已发布的直播信息。

游客进入首页后，点击系统页面“注册”按钮，即可进入注册页面。

注册页面要求用户输入注册用户名、密码并确认密码，用户输入相应信息后，

点击注册按钮，系统将确认用户名是否已经被注册过，如果用户名未被注册过并且密码符合填写要求，那么就可以完成注册。

注册完成后系统会自动跳转至登录页面，用户进入系统主页面时点击右上角登录按钮，也可进入登录页面。

在登录页面，用户需要输入用户名及密码，点击登录按钮，系统将确认该用户名是否存在，以及用户名和输入密码是否匹配，如果系统确认信息无误，那么用户登录成功。

6.2.2 注册用户

未登录用户首页有“登录”按钮，点击后进入登录页面，在页面输入用户名和密码进行登录，系统确认用户名和密码无误后将成功登录至该用户的个人主页，个人主页中有个人中心界面，用户可在个人中心界面进行其他操作。用户可通过页面“退出登录”按钮退出当前登录状态，退出登录后可以以其他账号重新登录。

用户登录后，可以在个人中心查看个人信息，也可以在用户中心中可以进一步完善或者修改相应用户信息选项，包括性别、年龄、出生日期、语言种类、个人简介、用户头像等。

用户可以在个人中心查看个人信息，已填写的个人信息会显示在界面的相应位置上。

注册用户进入会员中心，选择会员充值即可成为会员。会员充值有两种，月费会员和年费会员，用户可以根据自己的选择购买会员。只有成为会员才能进行求助、进入聊天室等。

6.2.3 会员（付费用户）

当用户成为会员后，就可以使用进入聊天室交流等功能。

6.2.3.1 聊天室

每个聊天室均有主题描述，登录后可以选择感兴趣的聊天室，若并未充值会员或会员到期，则系统提示“您现在不是会员，不能进入聊天室”，同时页面将跳转至充值页面；若用户当前仍是有效会员用户，则弹出聊天室对话页面进入群聊。

会员除加入已有聊天室以外，还可以自己申请创建聊天室，系统通过申请之后，聊天室会显示在首页列表，其他用户可以选择进入。

6.2.3.2 匹配用户

用户可以在首页点击“匹配用户”按钮，可进入匹配信息填写页面。

匹配信息页面有：希望匹配的用户的语言、年龄、职业、自身水平等，用户可以在每一项中选择合适的范围，除语言以外，其他可选择“不限”，选择完毕后，系统将根据填写的内容对系统中其他用户进行检索。

系统根据用户填写的信息检索出符合条件的其他用户，并按照一定的顺序进行排序，将在线用户排在前面，离线用户排在后面。

6.2.3.3 一对一连线

用户可以在系统匹配到的相关用户中进行选择，可以查看用户资料发起与该用户的会话。

点击“聊天”按钮向该用户发起连线，若并未购买会员或会员已到期，则系统将会提示“您现在不是会员，不能发起连线”，页面跳转进入充值页面；若当前仍是有效会员，则可以发出对话请求。若一定时间内连线请求被同意，则进入对话页面；否则连线失败，用户可以继续向其他用户发出对话请求。

在对话页面中，可以发送文本信息，语音信息或者视频通话，语音通话和视频通话发起后，在对方同意的情况下方可接通。

完成连线后，用户可对解答方进行评价，本次连线将会被记录在求助记录中。

6.2.3.4 个人中心

用户在首页点击“个人中心”进入个人中心页面。

个人中心页面包括基本信息、个人能力认证、会员中心、历史记录、账号管理、退出登录菜单。选择不同的菜单，页面将会显示相应内容。

基本信息：用户基本信息分别有用户名、语种、年龄、性别、职业、感兴趣的语种、爱好等，点击右上角编辑按钮可以对信息进行编辑，编辑后可对本次修改进行保存。

个人能力认证：会员填写相应语言能力信息进行认证。如果用户未进行认证，则页面将显示“未认证”，可以点击下方“填写认证信息”按钮，输入语种、已获得的能力水平等信息来进行个人能力认证，如果已提交认证信息，页面将会显示“系统正在进行认证”如果认证成功，该页面将会显示认证信息以及“认证成功”。用户认证完成后即可帮助他人。

会员中心：会员中心显示会员的有效期，同时可以在会员中心进行会员充值。

历史记录：历史记录分为求助记录和帮助记录。求助记录可显示该用户向他

人求助的历史记录，包括连线的对象用户、时间、聊天记录等。帮助记录显示本人帮助过他人的记录，包括时间、聊天记录等。

账号管理：可以修改密码，页面提示用户输入旧密码和修改的新密码，系统对用户输入的密码进行验证，验证密码修改有效后，完成修改并提示用户修改成功，修改成功后，需要用户使用新密码重新登录。

退出登录：点击后用户回到首页，可以使用其他账号登录。

备注：

（1）除首页以外，其他每个页面都在左上角有“返回”按钮，点击即可上一级页面。

（2）对于非会员注册用户，其会员中心一项中的会员特权为“非会员”，其个人能力认证以及历史记录显示为灰色，不能点开。

6.2.4 解答人

会员完成个人能力认证后可以使用一对一聊天交流等功能。

当被其他用户连接时，页面将弹出该用户的求助信息，同时也可以查看求助用户的其他基本信息等。

在提供帮助时，可以使用文本、视频的形式进行交流，视频交流用户可以选择接收或者拒绝。

总体设计

第一章 系统说明

1.1 系统功能

本系统是一个在线英语中文交互式学习系统，该系统基于视频、音频及文本为用户提供沟通途径。用户可以通过该系统进行英语和中文的学习，或者在中英文学习等方面帮助他人，同时结交更多的朋友。作为求助者的用户，可以在线制定个性化的搜索条件，搜索满足特定条件的其他用户，向其发送求助请求，然后与其展开交流。作为“教师”一方的用户，可以答复求助者的请求，并收取虚拟币。

1.2 运营方式

本系统提供充值服务，用户可以购买虚拟币用于系统内的交易。用户在答复其他用户的求助时，可以要求收取虚拟币；用户在向其他用户求助时，需要支付对方要求数目的虚拟币。通过这种方式，用户满足了自己的学习需求以及社交需求，而系统运营方也从虚拟币充值服务中赚取了利润。

1.3 运行环境

本系统采用 Python 语言编写，运行于 Windows 操作系统，与以下操作系统皆可兼容：

- Windows 10 (8u51 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 R2 SP1 (64-bit)
- Windows Server 2012 and 2012 R2 (64-bit)

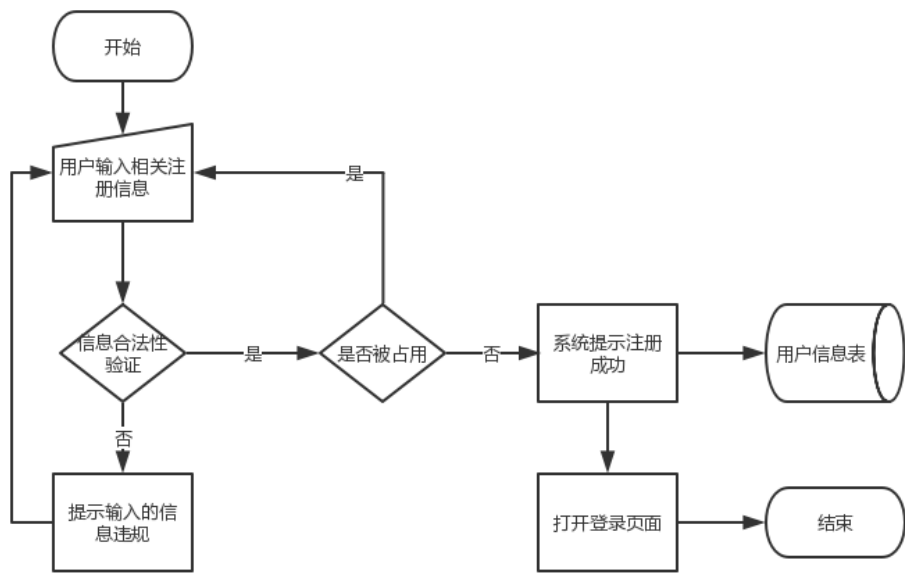
1.4 模块划分

本系统主要由四大模块组成：注册登录模块、用户模块、内容管理模块、充值模块以及后台管理模块。注册登录模块提供注册新用户、已有用户登录两大功能；用户模块的功能有：查看用户信息，修改用户信息，开始或结束用户之间的视频/音频/文字聊天，以及查看用户历史记录；内容管理模块主要对系统页面显示的内容进行管理，排序等操作；充值模块负责接收用户充值信息，并在用户充

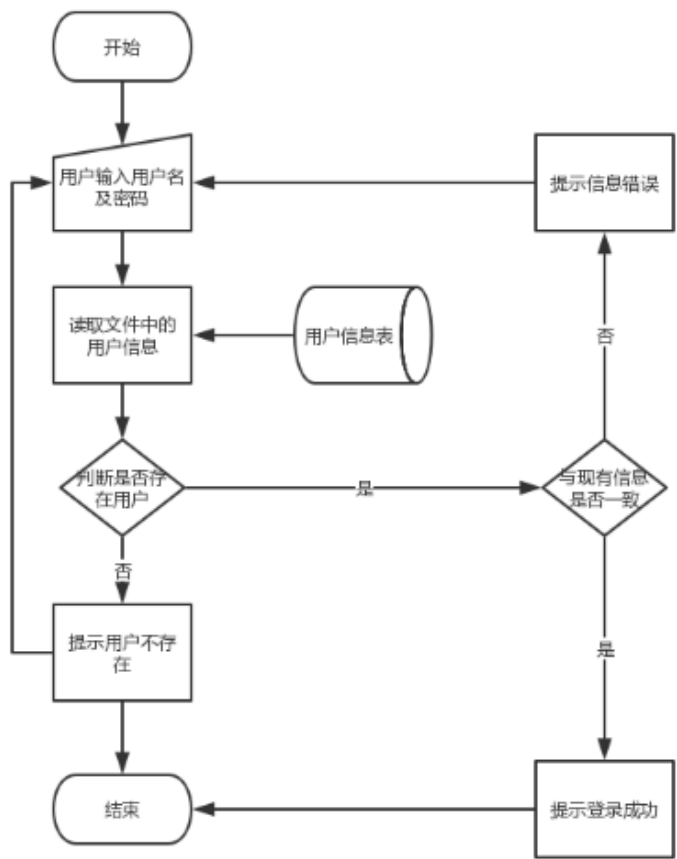
值之后修改用户身份（会员/非会员），或者修改用户剩余虚拟币数量；后台管理模块是底层模块，用于接收上述模块的各种请求，以及根据请求操作数据库。

1.5 系统流程图

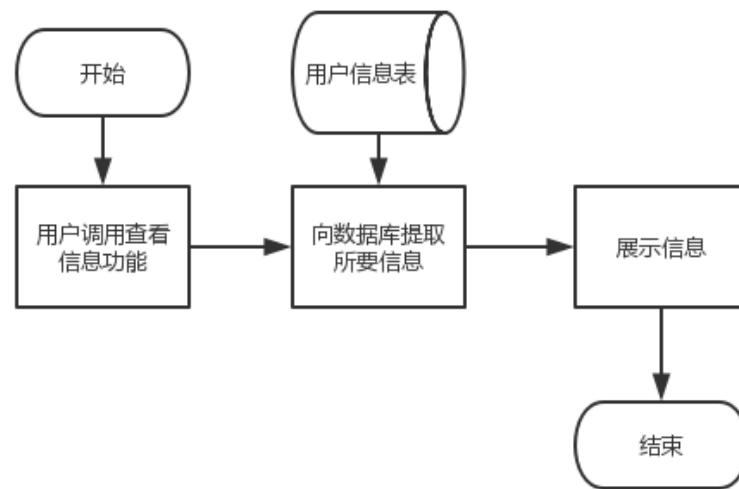
1.5.1 注册



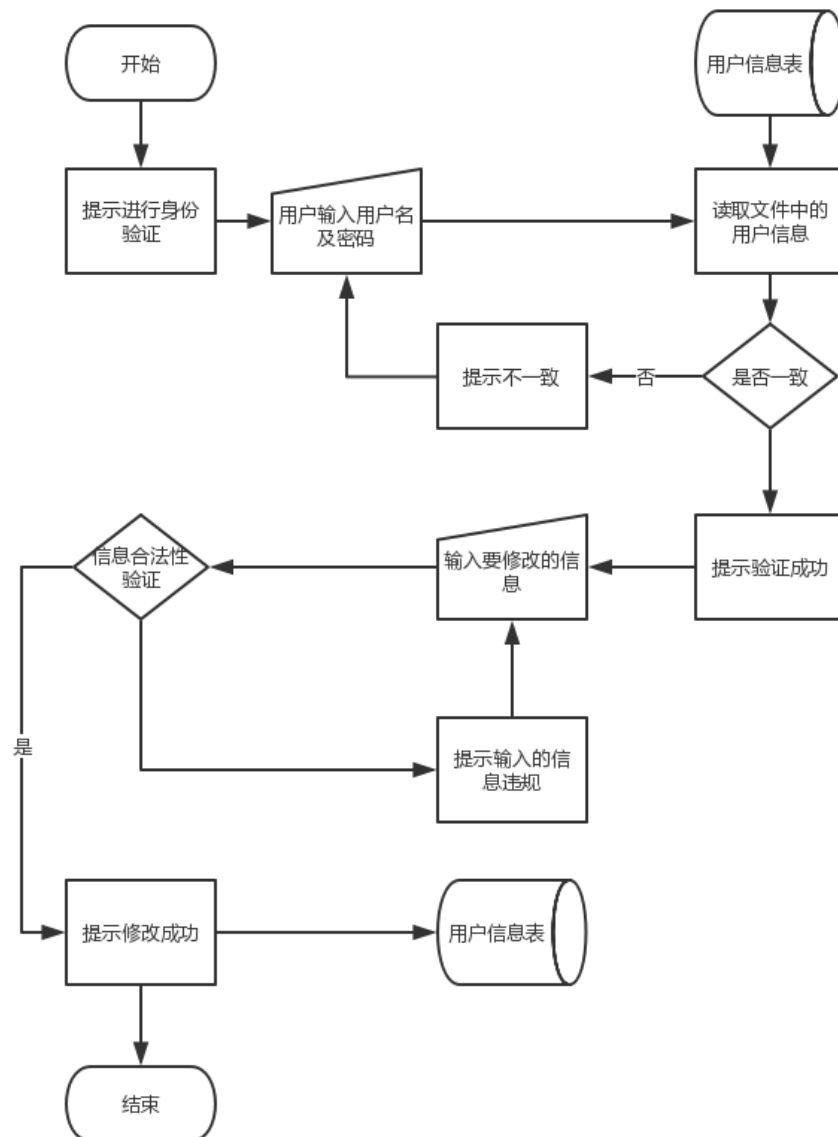
1.5.2 登录



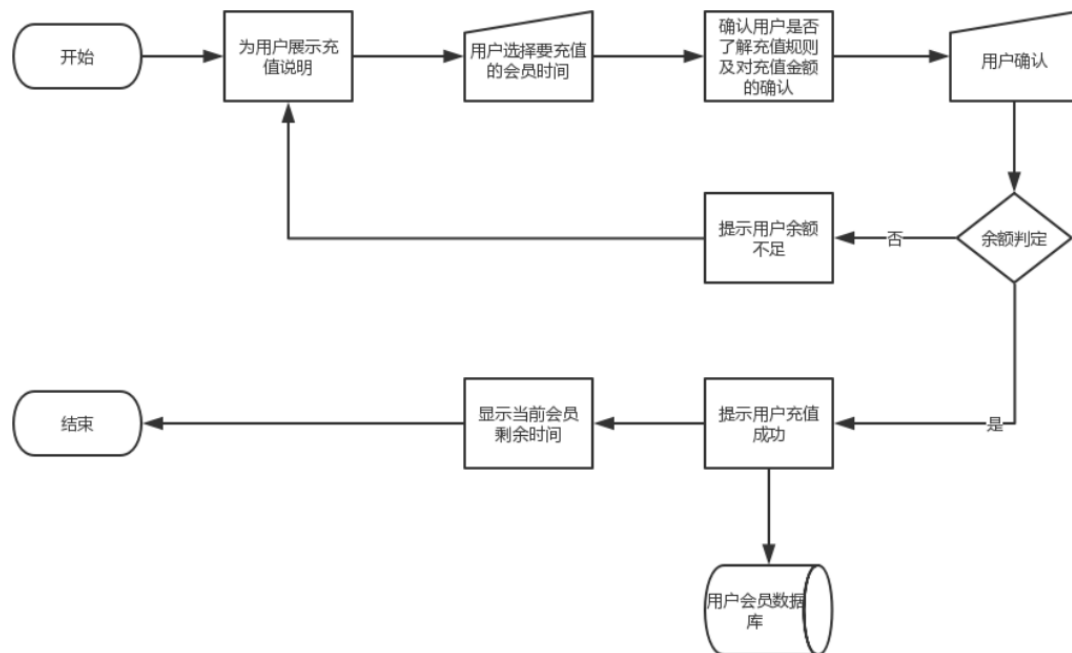
1.5.3 查看用户信息



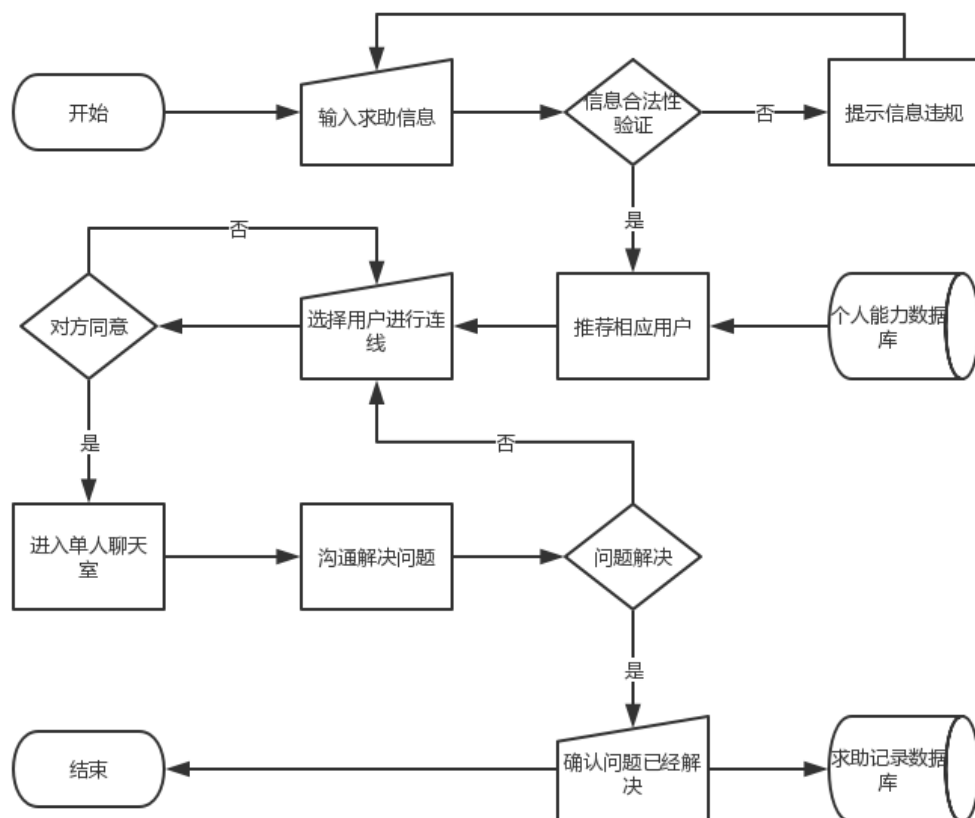
1.5.4 修改用户信息



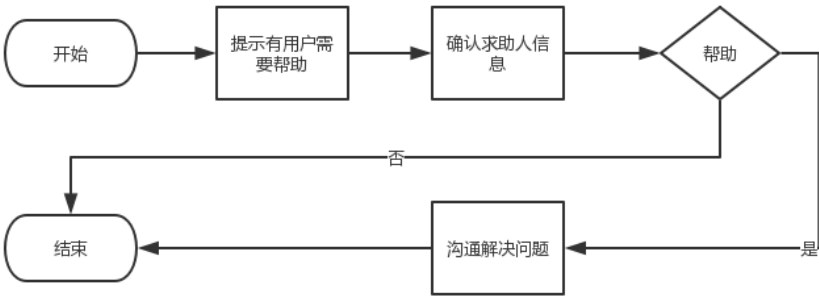
1.5.5 购买会员



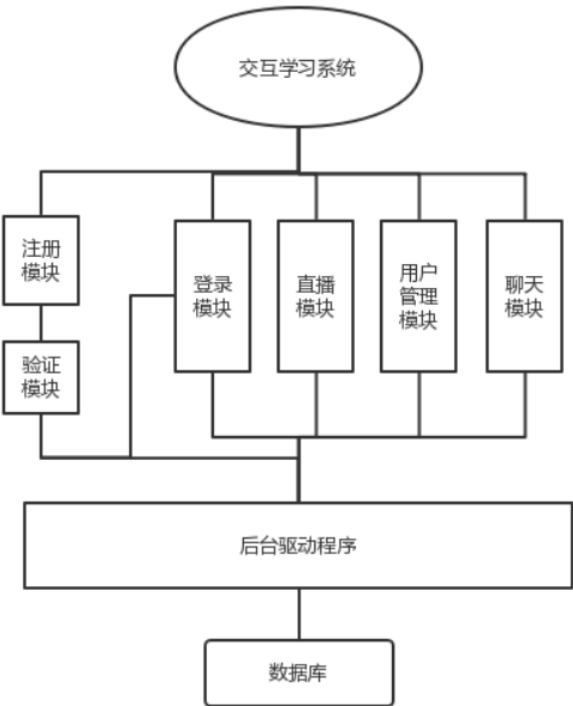
1.5.6 求助



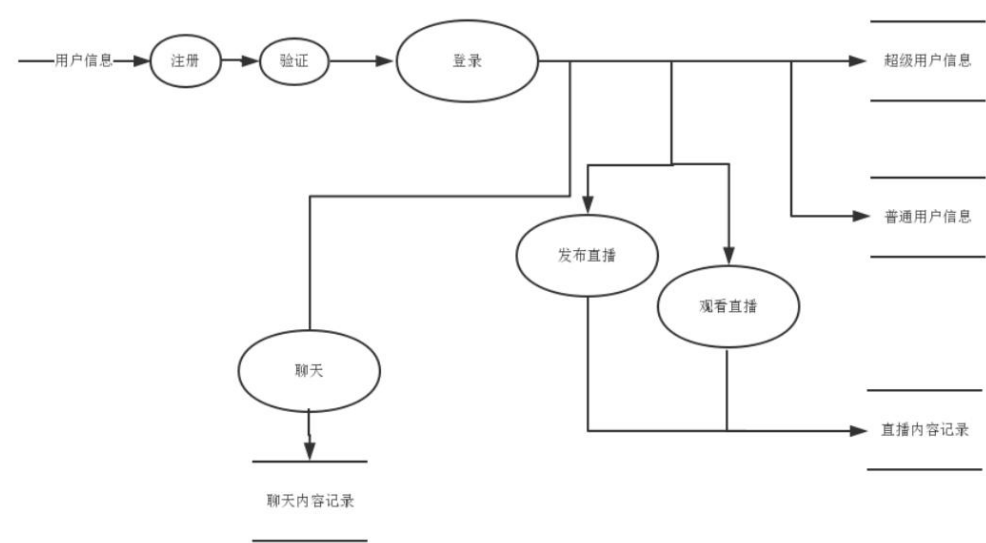
1.5.7 提供帮助



1.6 系统结构图



1.7 系统数据流图



第二章 用户使用

本系统为在线中英交互式学习系统，为需要学习和提升语言能力的用户提供了一种在线学习方式。用户使用部分主要是向该系统的使用者说明该系统的使用方式。

2.1 软件运行条件

该应用软件仅支持在 windows 系统上运行，支持的 windows 操作系统版本见第一章运行环境部分。

该应用软件运行的硬件最低配置：内存 1GB，硬盘 100GB，需要网卡、声卡、摄像头、麦克风等。

该软件必须在网络连接良好的环境下运行，如果没有网络连接，该软件不能正常使用。

2.2 使用说明

2.2.1 注册与登录

用户进入该系统后，系统显示主页页面，主页页面中包含聊天室列表和直播列表，在用户未登录状态下不能查看详细内容。

用户点击系统右上方“登录”按钮，系统弹出登录界面，提示用户输入用户名及密码，如果用户输入匹配，则成功登录，否则系统弹出提示框提示“用户名或者密码错误”，用户需要重新输入。

用户登陆后进入主页面，主页面左边一栏是“个人中心”按钮，点击“个人中心”可以查看当前登录用户的各项信息。

2.2.2 个人中心的使用

用户点击左边一栏的“个人中心”按钮，进入个人主页面，主页面中包括基本信息、我的活动、历史记录、会员中心、能力认证、账号管理、退出登录菜单。

用户点击“基本信息”可以查看当前用户的信息，点击“修改”按钮，可以对信息进行修改，修改后点击“保存”即可以保存修改。

用户点击“我的活动”可以查看当前用户加入的聊天室、创建的聊天室信息。

用户点击“历史记录”可以查看当前用户与他人聊天的记录。

用户点击“会员中心”可以查看当前的会员期限，可以在会员中心充值会员。

用户点击“能力认证”可以查看当前用户认证情况，输入本人的语言登记及证明材料可以申请认证，认证通过后，可以在这里查看认证的状态。

用户点击“账号管理”可以修改当前密码。

用户点击“退出登录”可以退出当前已登录的账号。

2.2.3 聊天室的使用

用户可以在主页查看聊天室列表，会员用户可以点击“加入聊天室”进入聊天室进行聊天。对于非会员用户，点击该按钮后，系统会弹出“您现在不是会员，不能进入聊天室”，此时用户需要先完成充值才能进入聊天室。

用户可以申请新的聊天室，点击左上方“创建聊天室”按钮，用户输入对于该聊天室的描述等内容向系统提出申请，如果申请通过，系统将会提示用户“聊天室申请成功”并在个人中心“我的活动”中体现。

2.2.4 求助与帮助

用户点击主页面左边一栏的“求助”按钮，进入求助信息页。在求助信息页面中输入用户求助的描述、学科等信息，点击下方“开始查找”按钮查找匹配的可提供帮助的用户。检索完成后，页面上将会显示符合要求的用户，当前用户可以查看这些用户的信息，点击“聊天”按钮可以向该用户发起连线求助。

被求助的用户会收到连线消息，用户点击“接受”按钮即可和求助用户建立一对一的连线，此时连线的两个用户可以通过文本、语音、视频的方式进行交流。如果用户有其他的安排不能够和求助用户建立一对一联系，那么第一点击“拒绝”按钮拒绝该用户的求助请求。

第三章 测试手册

3.1 测试策略

测试是为了发现程序中的错误而执行程序的过程，好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。成功的测试是发现了迄今为止尚未发现的错误的测试。在测试时要遵循相应的测试准则，比如所有的测试均应该追溯到用户需求，把 Pareto 原理应用到软件测试中。Pareto 原理说明，测试发现的错误中的 80%很可能是由程序中 20%的模块造成的。当然，问题是怎样找出这些可疑的模块并彻底地测试它们。

常见的测试方法有两种，黑盒测试和白盒测试。其中黑盒测试是在已知系统应该具有的函数的前提下进行，用于测试每个功能是否能正常使用。白盒测试要

求测试者完全知道程序的结构和处理算法,用于检测程序中的主要通路是否能按照预定要求正确工作。

鉴于本系统的开发环境,所以测试时采用对应的测试环境,即在 Windows 操作系统下进行测试。

系统环境	客户端	
硬件环境	CPU	Intel 1.80GHZ (含) 以上
	CPU 二级缓存	2MB (含) 以上
	内存	1G (含) 以上并支持扩展
	硬盘	100G (含) 以上
	显卡	主板集成 (必备) 或独立显卡
	网络连接	因特网 Internet 或局域网 (Local Area Network, LAN)
	网卡	100M (或千兆) 以太网卡
软件环境	操作系统	Windows 10 (8u51 and above) Windows 8.x (Desktop) Windows 7 SP1 Windows Vista SP2 Windows Server 2008 R2 SP1 (64-bit) Windows Server 2012 and 2012 R2 (64-bit)

在对本系统的功能进行测试时,采用黑盒测试方法。在对每个模块进行测试时,采用白盒测试法。模块测试时,要把测试重点放到以下五个方面:模块接口,局部数据结构,重要的执行通路,出错处理通路,边界条件。

当把模块组装成系统时,需要进行集成测试。本系统采用渐增式测试方法,即把下一个要测试的模块同已经测试好的那些模块结合起来进行测试,测试完后再把下一个应该测试的模块结合进来进行测试。

最后,在软件开发完毕后,还要进行确认测试,即验收测试,用来验证软件的有效性。软件的有效性就是指,软件的功能和性能如同用户所期待的那样。

3.2 测试方案和预期的测试结果

测试方案中包括测试时使用的输入数据 (测试用例),还包括每组输入数据预定要检验的功能,以及每组输入数据预期应得到的正确输出。

3.2.1

测试用例编号：	01	测试功能：	用户注册功能
测试方法：	黑盒测试	所属模块：	注册登录模块
测试用例（输入数据）： 用户名 密码			
预期测试结果： 若用户名或密码不符合相应的要求，也应有提示信息。 所有信息均正确后，用户点击确认按钮后，应该返回一个注册成功提示，并跳到登陆页面。			

3.2.2

测试用例编号：	02	测试功能：	用户登录功能
测试方法：	黑盒测试	所属模块：	用户注册登录模块
测试用例（输入数据）： 用户名 密码 验证码			
预期测试结果： 若用户输入的用户名或密码错误，则应给用户相应的提示信息，让用户修改。 若用户输入的用户名和密码均正确，则在用户点击登陆按钮后，应该跳转到登陆成功后的页面。			

3.2.3

测试用例编号：	03	测试功能：	修改用户信息
测试方法：	黑盒测试	所属模块：	用户模块
测试用例（输入数据）： 修改的用户信息，比如昵称，密码，身份证，年龄等			
预期测试结果： 若用户修改的部分符合数据库表中对该字段设置的要求，则当用户提交修改信息后，修改数据库表，并提示用户修改成功，否则，提示用户输入的信息不正确，让用户重新输入。 若出现用户信息无法连接到数据库的情况，则应由编码人员进行调试代码。			

3.2.4

测试用例编号：	05	测试功能：	直播功能
测试方法：	黑盒测试	所属模块：	用户模块
测试用例（输入数据）： 进行能力认证后的用户输入直播信息，比如直播开始时间，直播主题，进行直播间的申请。			
预期测试结果： 如果系统查看直播者输入的信息后发现有直播间，则把直播用户的直播信息放到主页面上的直播信息列表中，若没有直播间，则提示用户直播间已满，让用户等一段时间再申请。			

3.2.5

测试用例编号：	05	测试功能：	直播功能
测试方法：	黑盒测试	所属模块：	用户模块
测试用例（输入数据）： 进行能力认证后的用户输入直播信息，比如直播开始时间，直播主题，进行直播间的申请。			
预期测试结果： 如果系统查看直播者输入的信息后发现有直播间，则把直播用户的直播信息放到主页面上的直播信息列表中，若没有直播间，则提示用户直播间已满，让用户等一段时间再申请。			

3.2.6

测试用例编号：	06	测试功能：	充值功能
测试方法：	黑盒测试	所属模块：	充值模块
测试用例（输入数据）： 用户选择充值金额			
预期测试结果： 根据用户选择的充值金额对应的虚拟币数额，来更新用户的虚拟币数额，并据此把用户身份进行修改。若用户是按月充值的，则从充值时开始，更新用户的会员到期时间。若用户是按年充值的，则从充值时开始，更新用户的会员到期时间。			

3.2.7

用例编号：	7	测试功能：	查看历史记录
测试方法：	黑盒测试	所属模块：	用户模块
预置条件： 1. 用户已经注册登陆 2. 用户曾经发布过求助或替别人解答过或者两者均无			
操作步骤： 1. 用户成功登陆 2. 点击查看			
接收标准： 1. 后台接收请求后，调用相应的历史记录表，并进行查询			
备注：无			
测试结果和结论： 若用户从未进行求助或解答，则提示用户历史记录为空。 否则，显示用户的历史记录。			

3.2.8

用例编号：	8	测试功能：	查看用户信息
测试方法：	黑盒测试	所属模块：	用户模块
预置条件： 1. 用户已经注册完毕，并登陆系统 2. 用户已经对自己的信息进行了完善或只保存了注册时的信息			
操作步骤： 1. 用户登陆成功 2. 点击查看用户信息			
接收标准： 后台接收到请求后，从数据库中调用相应的用户信息表，并把结果反馈回来。			
备注：无			
测试结果和结论： 显示用户的基本信息和完善后的信息，如果用户没有完善信息，则未完善后的信息也需显示出来。			

3.3 测试进度计划

测试活动	计划开始日期	实际开始日期	实际结束日期
模块测试	2018/9/15		
集成测试	2018/9/18		
验收测试	2018/9/24		

第四章 实现计划

4.1 目的

通过对系统软件的总体设计，明确工程与开发的实际走向与进度，形成有效的掌握与把控，并找到最佳方案。通过具体的实现计划保证系统整合的效率。

4.2 进度计划

第一周：小组讨论本系统的需求，并撰写需求分析文档。

第二周：根据需求分析文档，制定总体设计，并撰写相关文档，其中包括：系统说明、用户手册、测试手册、详细的实现计划、数据库设计等文档。

第三周：根据总体设计制定详细设计，并进行初步的编码实现。

第四周：整合系统，进行演示。

第五章 数据库设计

5.1 引言

5.1.1 编写目的

本文档主要对在线英语中文交互学习系统在设计阶段对数据库设计的说明和描述。本文档的主要读者应为后续设计阶段的设计人员、实现阶段的编码人员和实际使用时的系统管理员。

5.1.2 背景

本次需要开发的系统以“在线英语中文交互学习系统”作为本次系统的命名，最终的产品是生产出符合本需求文档的英语在线学习系统。

5.2 表列表

序号	中文名称	物理表名
1	用户表	userTable
2	直播表	liveTable
3	内容表	contentTable

5.3 表内容

5.3.1 用户表

字段列表

序号	中文名称	列名	数据类型	长度
1	用户名	userId	VarChar	36
2	用户昵称	userName	VarChar	36
3	密码	passWord	VarChar	50
4	身份证	idCard	Int	50
5	年龄	age	int	5
6	兴趣语言	languageType	VarChar	10
7	地址	address	VarChar	40

5.3.2 直播表

字段列表

序号	中文名称	列名	数据类型	长度
1	用户昵称	userName	VarChar	36
2	用户名	userId	VarChar	36
3	直播房间号	houseId	VarChar	36
4	观看直播人数	watchPeopleNumber	VarChar	36
5	直播开始时间	beginTime	VarChar	36

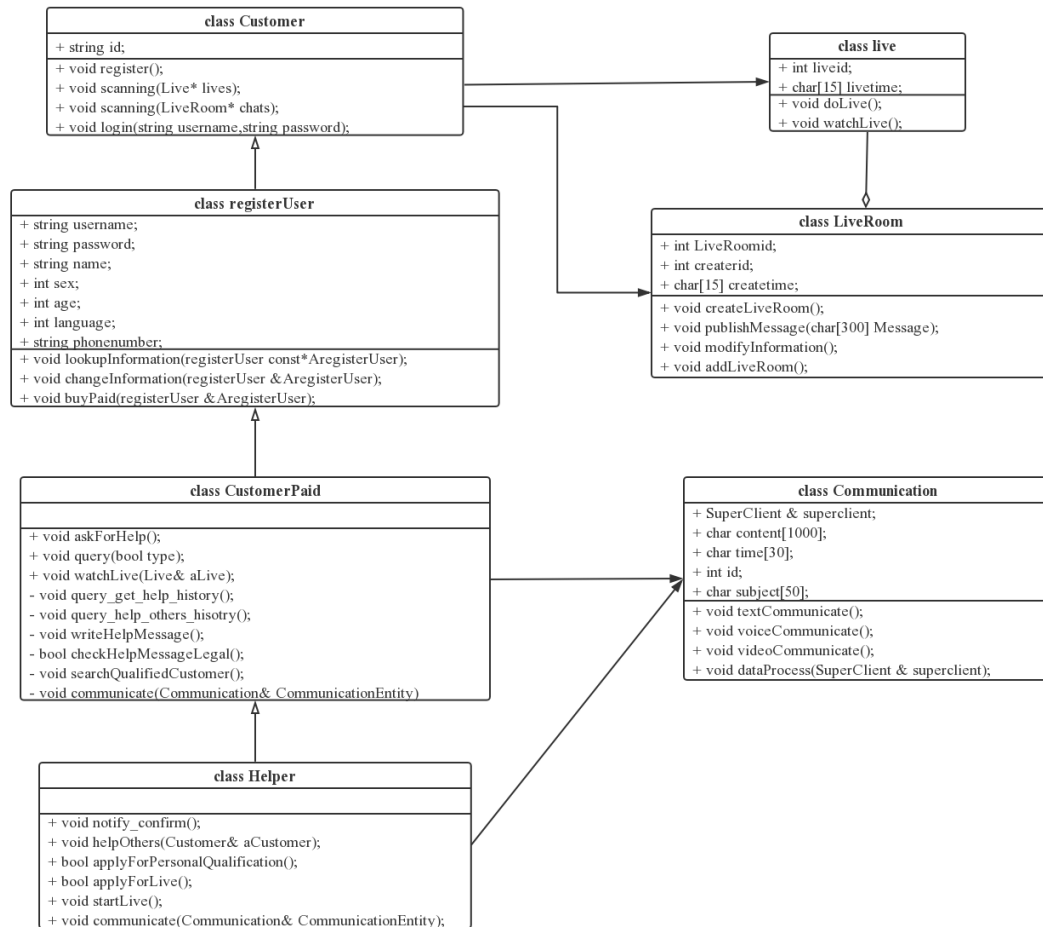
5.3.3 内容表

字段列表

序号	中文名称	列名	数据类型	长度
1	用户昵称	userNameev	VarChar	36
2	用户名	userId	VarChar	36
3	聊天内容	chatContext	VarChar	500
4	聊天时间	chatTime	VarChar	36
5	字体	font	VarChar	36
6	字体颜色	fontColor	VarChar	36
7	字体大小	fontSize	VarChar	36

详细设计

1 类图



2 模块设计

本系统可划分为用户模块、数据库管理模块、直播间模块以及聊天室模块。

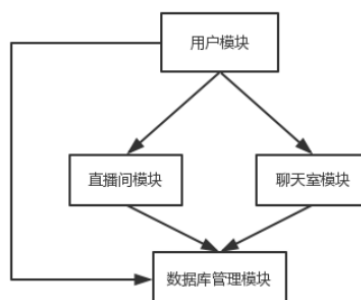
用户模块主要负责管理 4 类不同类型的用户。

数据库管理模块主要负责对于数据的更新存储等功能。

直播间模块主要负责直播相关功能。

聊天室模块主要负责聊天室相关功能。

各模块之间的调用关系：



3 模块中类的实现

3.1 用户模块

实现该模块的类：游客类，注册用户类，付费用户类，解答人类。

```
class Customer:
{
    public: id;
    public: void register();
           void scanning(LiveRoom* chats);
           void login(string username, string password);
}
```

注：本部分中所有类的实现都以伪代码的形式表示，并不代表具体的实现语言。

3.1.1 游客类

(1) 类的属性

游客类（基类）包含的属性包括用户 id。

用户 id: 每位用户作为游客访问应用各项内容时，系统都会给该游客随机分配 id，作为该用户的唯一标识。

(2) 类的方法

游客类中包含注册、浏览、登录。

i) 注册

游客通过调用注册函数，完成用户的注册，包括用户名，用户昵称，密码，手机等基本信息，作为登录注册用户的凭证，并把注册的信息存入数据库。

ii) 浏览

游客通过调用浏览函数，可以完成浏览应用各个模块的功能，包括浏览直播列表，浏览聊天室列表，通过浏览函数，游客在聊天室列表里面选择进入聊天室内交流。

iii) 登录

游客通过调用登录函数实现注册用户的登录，通过用户输入的用户名和密码与数据库中的注册用户信息进行比较，核对无误后登录成为注册用户，注册用户继承游客类的所有属性和方法。

3.1.2 注册用户类

```
class registerUser : public Customer
{
    private: string username;
            string password;
            string name;
            int sex;
```

```

    int age;
    int language;
    string phonenumber;
public:
    void lookupInformation(registerUser const*AregisterUser);
    void changeInformation(registerUser &AregisterUser);
    void buyPaid(registerUser &AregisterUser);
}

```

（1）类的属性

注册用户类包含的属性包括用户名，密码，用户昵称，用户性别，用户年龄，用户倾向语言，用户手机号等。

用户名：用户通过注册登录为注册用户后，用户名为用户注册时所填的唯一标识用户名。

密码：用户通过注册登录为注册用户后，用户名为用户注册时所填的唯一标识用户名。

用户昵称：用户注册时所填的昵称。

用户性别：用户注册时所填的男/女

用户年龄：用户注册时所填的年龄。

用户倾向语言：用户注册时所填的倾向所学的语言。

用户手机号：用户注册时所填的手机号。

（2）类的方法

i)查看和修改用户信息

注册用户通过调用查看和修改用户信息方法可以对用户的相关属性（用户昵称，用户性别，用户性别，用户倾向语言和用户手机号等）进行查看和修改。

ii)购买会员

注册用户通过调用购买会员函数，完成会员付费后开通会员，将注册用户在数据库中升级成为付费用户，转向付费用户类。

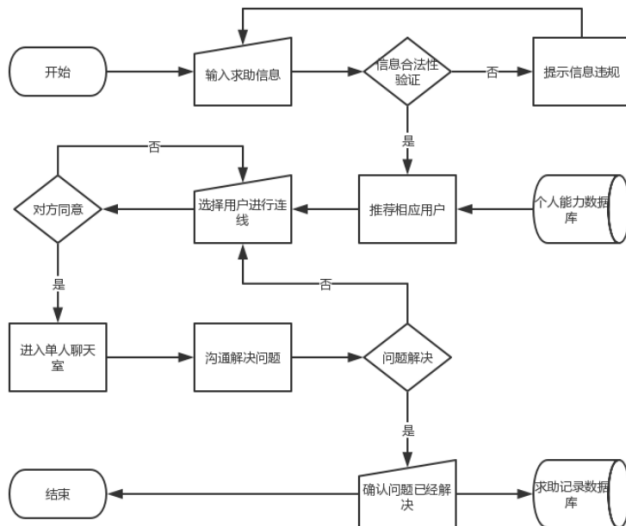
3.1.3 付费用户

付费用户可以作为求助者搜索指定条件的、有一定资质的用户，然后向其发起求助。在搜索特定类型的用户时，系统需要先检查其搜索条件是否合法、是否包含敏感语句，以防不符合主旋律的内容在平台上传播。若搜索条件合法，则查询数据库，显示符合条件的用户列表。然后就可以选择某一用户发送求助请求，发送过程由 **Communication** 类提供接口。

付费用户也可以查询两种历史记录：施助记录和得到帮助的记录。其中涉及到的对数据库的查询操作需要调用超级用户类。

```
class UserPaid : public registerUser
{
public:
    void askForHelp()
    {
        writeHelpMessage();
        checkHelpMessageLegal();
        searchQualifiedUser();
        communicate(Communication& CommunicationEntity);
    };
    void query(bool type)
    {
        if(type)
            query_get_help_history();
        else
            query_help_others_hisotry();
    };
private:
    void query_get_help_history();
    void query_help_others_hisotry();
    void writeHelpMessage();
    bool checkHelpMessageLegal();
    void searchQualifiedUser();
    void communicate(Communication& CommunicationEntity)
}
}
```

流程 1：求助



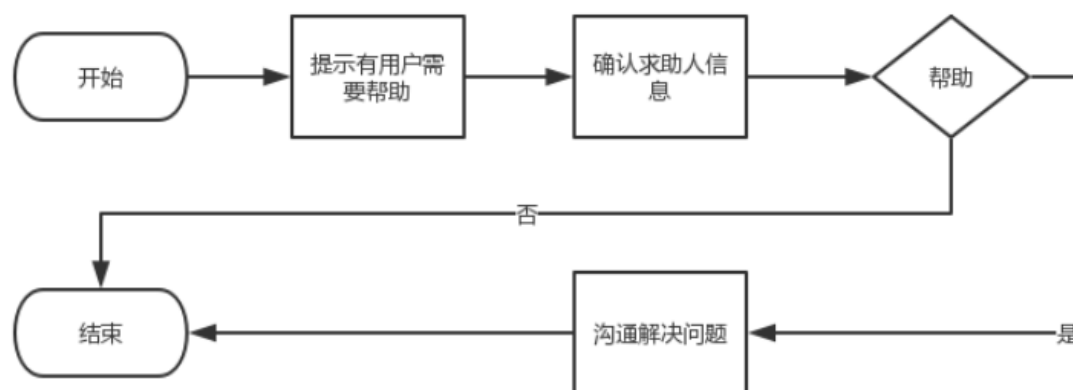
3.1.4 解答人

解答人作为一个帮助他人的角色，可以接收到他人的求助信息（弹窗形式），然后决定是否答应该请求。一旦点击“同意”，则调用成员函数 `helpOthers()`，在其内部调用 `Communication` 类开始双方之间的交流。

解答人在帮助他人之前需要进行资质认证,通过上传材料然后人工审核的方式实现。资质信息由超级用户修改。

```
class Helper : public UserPaid
{
public:
    void notify_confirm();
    void helpOthers(User& aUser);
    bool applyForPersonalQualification();
    void communicate(Communication& CommunicationEntity);
}
```

流程 2: 解答人响应求助信息



3.2 数据库管理类

3.2.1 超级用户类

(1) 类属性

i) 用户名: 管理员 id, 作为超级用户登录所用, 为整个系统的唯一标识的超级用户 id 名。

ii) 密码: 超级用户登录时所需要的验证密码。

iii) 数据管理类: 当存在系统数据操作时, 调用超级用户类, 进而调用该数据管理类, 进行数据相关的操作。

(2) 类方法

i) 修改超级用户 id: 超级用户通过修改 id 来更换超级用户的 id 名字。

ii) 修改密码: 超级用户通过该方法进行密码的修改, 从而进行接下来的数据管理操作。

iii) 调用数据管理类: 进行系统中数据的管理操作。

```
class superuser
{
```



```

public:
    int superuserid;
    char superuserpass;
    datamanege D;

    void changesuperid(superuserid);
    void changesuperpass(superuserpass);
    void datamanege();
};

```

3.2.2 数据管理类:

(1) 类方法

- i) 游客更新信息: 更新游客相关信息, 例如游客 ID 账号、等相关信息。
- ii) 注册用户更新信息: 更新注册用户相关信息, 例如注册用户 ID 账号、等相关信息。
- iii) 付费用户更新信息: 更新付费用户相关信息, 例如付费用户 ID 账号、等相关信息。
- iv) 聊天室更新信息: 更新聊天室相关信息, 例如聊天室 ID 账号、聊天用户名等相关信息。

```

class datamanege
{
public:
    void datamanege::chcustomer();
    void datamanege::chregcustomer();
    void datamanege::chuserpaid();
    void datamanege::chhelper();
    void datamanege::chcommunication();
};

```

3.3 聊天室模块

3.3.1 聊天室类

(1) 类的属性

聊天室的属性包括聊天室的 id, 聊天对象的用户名, 聊天时间, 聊天内容, 聊天主题。

i) 聊天室的 id: 每个聊天室被创建时, 系统会分配给该聊天室一个 id, 用于唯一标识该聊天室, 其为整形。

ii) 聊天双方的用户名: 聊天时, 系统会记录聊天双方的用户名, 便于管理。用户名用字符串来表示。

iii) 聊天时间: 聊天开始后, 由系统记录开始时间和结束时间。时间是一个

字符串形式，表示创建的年，月，日，时，分。

iv) 聊天内容：系统会记录聊天的内容，便于用户查看聊天记录。

v) 聊天主题：用户在进行聊天之前，会对聊天主题进行描述。系统会记录聊天主题，便于发布到聊天室目录栏中。

(2) 类的方法

聊天室类包括创建聊天室，语音聊天，文本聊天，视频聊天，以及数据处理等函数。

i) 创建聊天室：求助者或解答者进行解答时，由其中一方创建该聊天室。调用该方法后创建一个新的聊天室，并把聊天的主题，聊天 id 等信息存储到数据库中。

ii) 文本聊天：解答用户或求助者选择该方式进行聊天时，系统会记录聊天开始时间，结束时间，并把聊天记录也存储起来。

iii) 语音聊天：解答用户或求助者选择该方式进行聊天时，系统会记录聊天开始时间，结束时间。

iv) 视频聊天：解答用户或求助者选择该方式进行聊天时，系统会记录聊天开始时间，结束时间。

v) 数据处理：该函数会调用超级用户，让超级用户来管理聊天信息。

vi) 退出聊天室：解答者或求助者在交流完毕后，任何一方均可随意退出聊天室，当聊天室没人了，由系统对该聊天室进行回收。

```
Class Communication
{
Public:
Void createRoom();
Void textCommunicate();
Void voiceCommunicate();
Void videoCommunicate();
Void dataProcess(SuperClient & superclient);
Private:
SuperClient & superclient;
Char content[1000];
Char time[15];
Int id;
Char subject[50];
}
```

3.3.2 类之间的关系

(1) 哪些类对该类进行调用？

付费用户对该类进行调用，付费用户可以进行求助，也可以进行解答。当求助者和解答者进行交流时，会用到聊天室。

(2) 该类调用了哪些类？

聊天的信息需要系统记录下来，故让该类调用超级用户，由超级用户完成相应工作。

用户手册

1 引言

1.1 编写目的

用户手册是详细描述软件的功能、性能和用户界面，使开发者以外的用户了解到如何使用该软件。编写用户手册的目的是为了帮助用户更好地了解和使用该软件，提高用户和软件的亲和度。用户手册的编写不仅使软件项目的开发者的工作有章可寻，同时完整和规范的用户手册更有助于提醒用户使用过程中应该注意的一些问题。

1.2 项目背景

本项目来自于软件工程课程实践课的作业要求，让学生实践在软件工程课程所学理论并对开发流程熟悉，进而养成团队合作的良好习惯，为以后走上工作岗位积累经验。开发人员共六人。在 [github](#) 上进行团队合作开发。

1.3 定义

聊天室：网络聊天室刚开始的时候是以文本聊天为主，后来出现了语音聊天，到了今天，视频、语音、文字都很好的融入到了聊天工具当中。

1.4 参考资料

- a 需求规格说明书
- b 总体设计说明书
- c 详细设计说明书
- d 测试计划

2 软件概述

本软件是一个基于视频、音频及文本等多特征融合的在线英语中文交互式学习系统，用户通过该系统实现英语和中文的学习以及结交更多的朋友。中美两国的用户通过寻找合适的同龄人，足不出户就可实现一对一的语言学习和交友目标。

2.1 目标

为用户提供一个快速高效地解决用户实际问题的平台。对于那些为提升自身能力而进行学习的用户，本软件为用户匹配出相应的伙伴进行学习，便于用户在最短时间内进入学习状态。对于那些需要实时了解某地信息的用户，比如出国留学，本软件可为用户匹配出那些当地的本地用户，由用户自由进行交流，获得在网

上搜索不到的信息。总之，本软件是桥梁，沟通那些有需求却得不到满足的用户。

2.2 功能

当用户注册登陆后，并且发布需求后，本软件会根据用户需求推荐按最适合该用户的学习交友资源。之后，用户就可以与推荐的用户进行交流，为了提升用户体验，用户可以与推荐的用户通过多种方式进行交，可以一对一地进行交流。

2.3 性能

a.数据精确度

输入数据中有用户名，密码。其中用户名 32 位的字母或数字组成的字符串，密码是 20 位的字母或数字组成的字符串。

用户的个人信息中，昵称是 32 位的字母或数字组成的字符串，年龄是 0 到 150 之间的整数。

个人特长是长度为 100 的字符串。

充值时的充值金额对于国内用户月费为 10 元，年费为 100 元。对于美国用户，月费为 2 美元，年费为 15 美元。其他国家的则根据当地经济发展水平斟酌进行定价。

聊天时间记录时，是用长度为 12 的字符串来记录年月日分秒。

b.时间特性

当用户进行注册登陆时，系统的响应时间为 1s,否则就提示出错信息。

本软件基于 windows 系统，不同版本的系统上运行此软件，处理时间也有所不同。

本软件使用的数据库是 MYSQL，数据传输发生在客户端和服务器之间，传输时间的长短受到服务器本身性能的影响。

c.灵活性

本软件在 windows 系统下运行，在不同版本下变更时，软件的运行环境会发生些许变化，此时软件的处理时间等会有些许不同，但总体看来，差别不大。

3 运行环境

3.1 硬件

软件系统运行时所需的硬件最小配置：

计算机型号	戴尔灵越 15 5000
主存容量	1G（含）以上并支持扩展
外存储器容量	100G（含）以上
媒体	麦克风，声卡
输入、输出设备	键盘，鼠标，摄像头，显示器
网络设备	网卡

3.2 支持软件

计算机型号	戴尔灵越 15 5000
主存容量	1G（含）以上并支持扩展
外存储器容量	100G（含）以上
媒体	麦克风，声卡
输入、输出设备	键盘，鼠标，摄像头，显示器
网络设备	网卡

4 使用说明

4.1 运行系统

运行可执行程序 onlineStudySystem.exe 即可运行本系统，打开系统将进入系统的主页面。系统主页面将显示直播列表、聊天室列表。

4.2 输入

用户可在主界面上选择注册或者登录，在执行注册和登录操作时，对用户输入的数据有一定的限制和要求。

用户在使用聊天室功能聊天以及进行一对一连线时，参与交流的用户可以输入聊天数据，系统对用户输入的聊天数据有一定的限制和要求。

4.2.1 输入数据背景

注册界面的输入数据包括注册用户名和密码，来自使用系统的用户的输入。该输入数据将被存入数据库中。系统将数据保存进入数据库之前，首先检查数据库中是否有相同的用户名，如果不存在冲突，则将相关信息存入数据库，否则提示用户更改用户名输入数据。

登录界面的输入数据包括注册用户名和密码，来自使用该系统的用户的输入。该输入数据将被用来在数据库中进行查询。系统将查询输入用户名数据在数据库中是否存在，如果不存在会提示用户更改用户名输入数据；如果存在，系统将输入数据在数据库中查询，比较用户名和密码是否和已有存储数据相同，如果相同那么登录成功，否则提示用户更改密码输入数据。

聊天室功能以及一对一连线功能的输入数据来自用户通过键盘或其他设备输入的聊天内容。该输入数据将被记录在聊天记录中。

4.2.2 输入数据格式

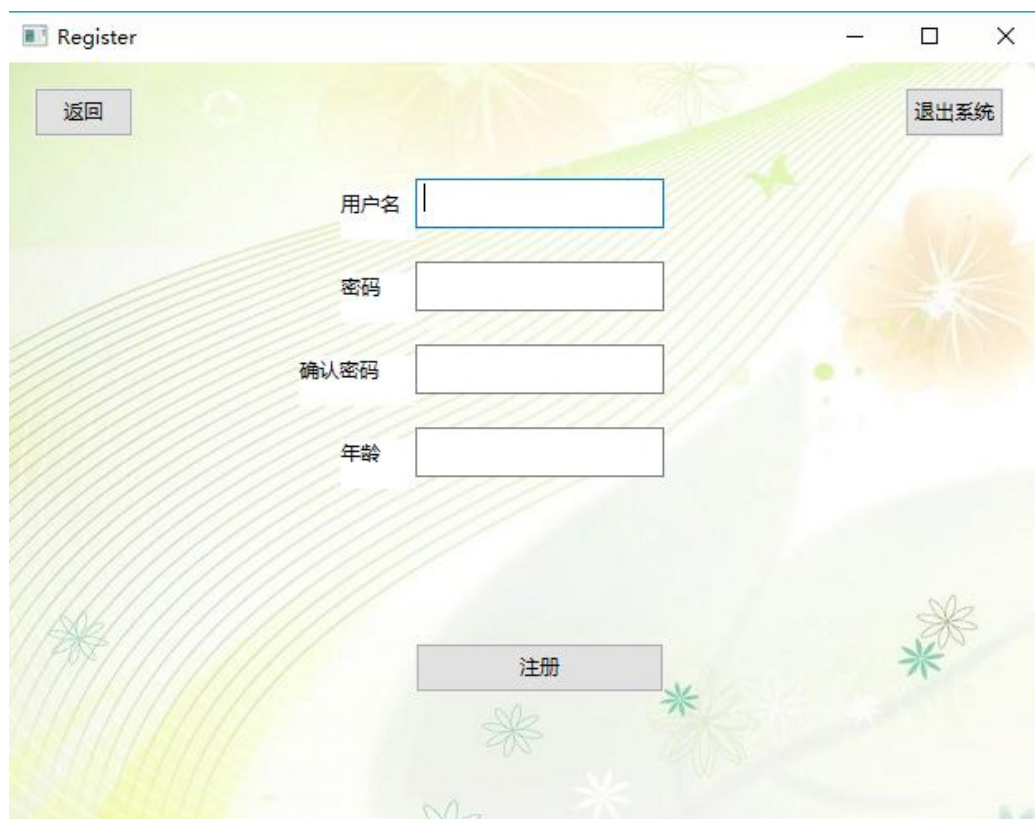
注册界面的输入数据包括用户名和密码。输入用户名的数据类型为字符串类型，可由数字、字母和下划线组成，用户名长度不超过 32 位。输入密码的数据类型为字符串类型，可由数字、字母和下划线组成，密码长度不超过 20 位。

登录界面的输入数据包括用户名和密码。输入用户名和密码的格式要求同注册界面。

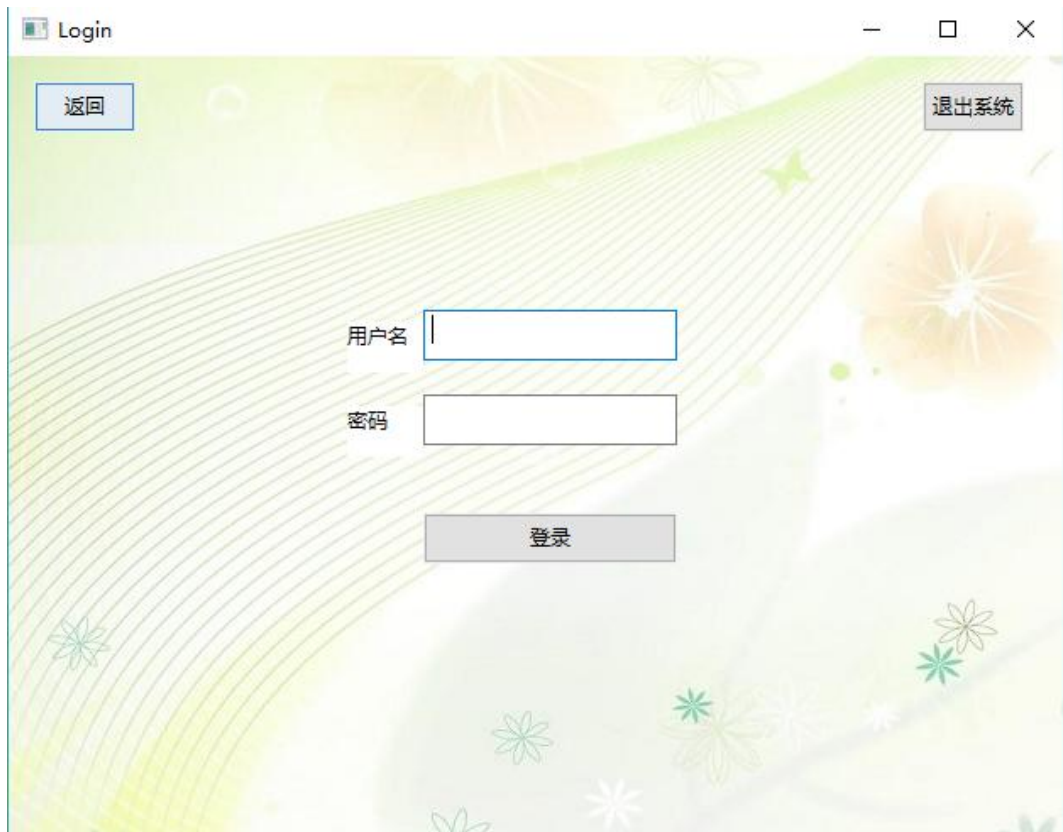
聊天室以及一对一连线的输入数据为用户输入的聊天内容。聊天内容的数据类型为字符串类型，可由数字、字母、文字以及其他字符构成，每一条聊天内容的长度不超过 1000 个字符。

4.2.3 输入举例

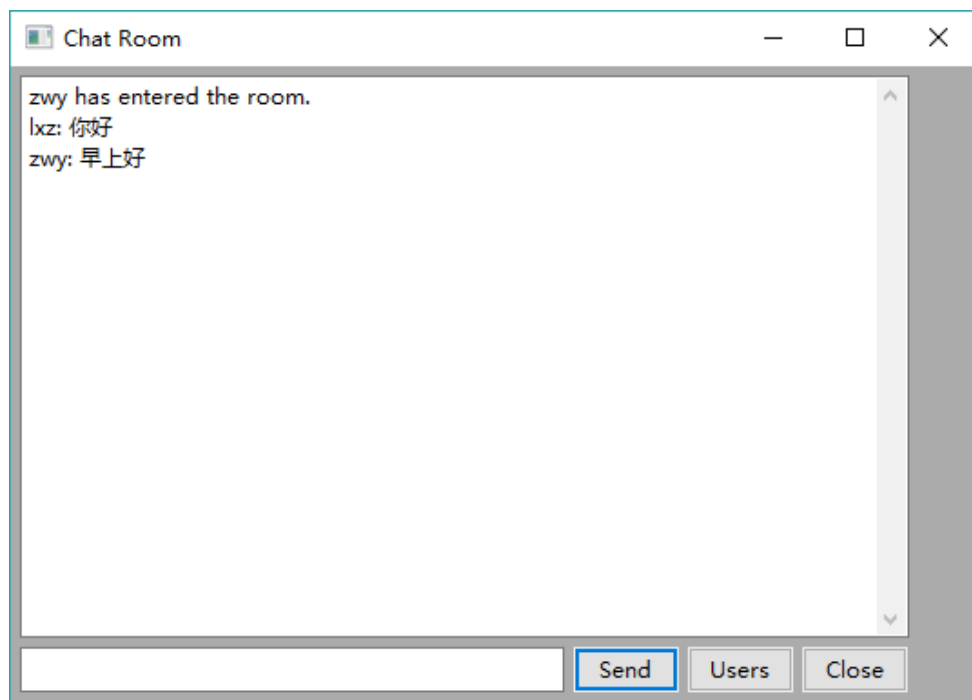
(1) 注册界面

A screenshot of a web-based registration form titled "Register". The form is set against a light green background with a subtle floral pattern. It contains four input fields: "用户名" (Username), "密码" (Password), "确认密码" (Confirm Password), and "年龄" (Age). Each field is accompanied by a small label to its left. A "注册" (Register) button is positioned below the input fields. In the top-left corner, there is a "返回" (Return) button, and in the top-right corner, there is a "退出系统" (Exit System) button. The window has standard OS controls (minimize, maximize, close) in the top-right corner.

(2) 登录界面



(3) 聊天界面



4.3 显示结果

本系统在不同的界面会显示不同的内容。

当用户点击“注册”按钮，系统将显示注册页面，用户注册成功时，系统将弹出“注册成功”的提示信息，用户注册失败，系统将弹出注册失败的相关信息。

当用户点击“登录”按钮，系统将显示登录页面，用户登录成功时，系统将直接进入登录后界面，用户登录失败，系统将弹出登录失败的相关信息。

当用户进入聊天室或者进行一对一连线时，系统显示相应的聊天界面和相关信息。

4.4 出错和恢复

系统可能出现的错误信息、含义以及用户应该采取的措施如下：

(1) 未登录不能使用该功能

出现该错误的原因是用户进入系统后未登录就是用直播间或者聊天室功能。如果用户没有登录，用户是不能进入直播间或者聊天室的，此时系统将对用户进行提醒，用户需要先登录才能使用这些后续功能。

遇到该错误信息，用户应采取的措施是进行登录或者注册后再进行登录。

(2) 注册失败

注册失败的原因可能有两种。第一种，用户输入的用户名或者密码不符合输入数据的要求，有除数字、字母和下划线以外的字符出现或者超过规定的长度。第二种，用户输入的用户名和密码符合输入数据的要求，但是用户名已在数据库中存在。

遇到该错误信息，用户应采取的处理措施为修改自己输入的用户名或者密码重新进行注册，如果注册成功，系统将提示“注册成功”提示信息。

(3) 登录失败

登录失败的原因可能有两种。第一种，用户输入的用户名之前并没有被注册，在数据库中没有该用户名对应的信息。第二种，用户输入的用户名和密码不匹配，与数据库中记录的对应用户名的密码不同。

遇到该错误信息，用户应采取的处理措施为修改自己的用户名或者密码，重新进行登录，如果登录成功，系统将进入登陆后界面。

(4) 非会员不能使用该功能

出现该错误的原因是非会员用户未付费直接使用某些付费功能。用户必须先付费成为会员后才能是用付费功能。

遇到该错误信息，用户应采取的处理措施是进入会员中心进行会员充值，充值完成后，再选择进入相应的功能。

(5) 网络连接出错

出现该错误的原因是网络连接出错导致不能正常接收和发送信息。

遇到该错误信息，用户应采取的处理措施是检查自己的网络连接情况，重新连接网络后重新加载相应的功能。

（6）系统停止响应

系统在运行过程中，由于各种外界因素的影响，可能会出现停止响应的情况。

遇到该错误信息，用户应采取的处理措施是关闭程序后将程序重新启动。

5 运行步骤

（1）用户运行可执行程序进入主页面。



在该界面用户可以选择进行注册或者是登录操作。

（2）新用户进行注册。

用户在注册界面输入相应的注册信息，完成注册。

Register

返回 退出系统

用户名

密码

确认密码

年龄

注册

(3) 已注册用户可以在登录界面完成登录。

Login

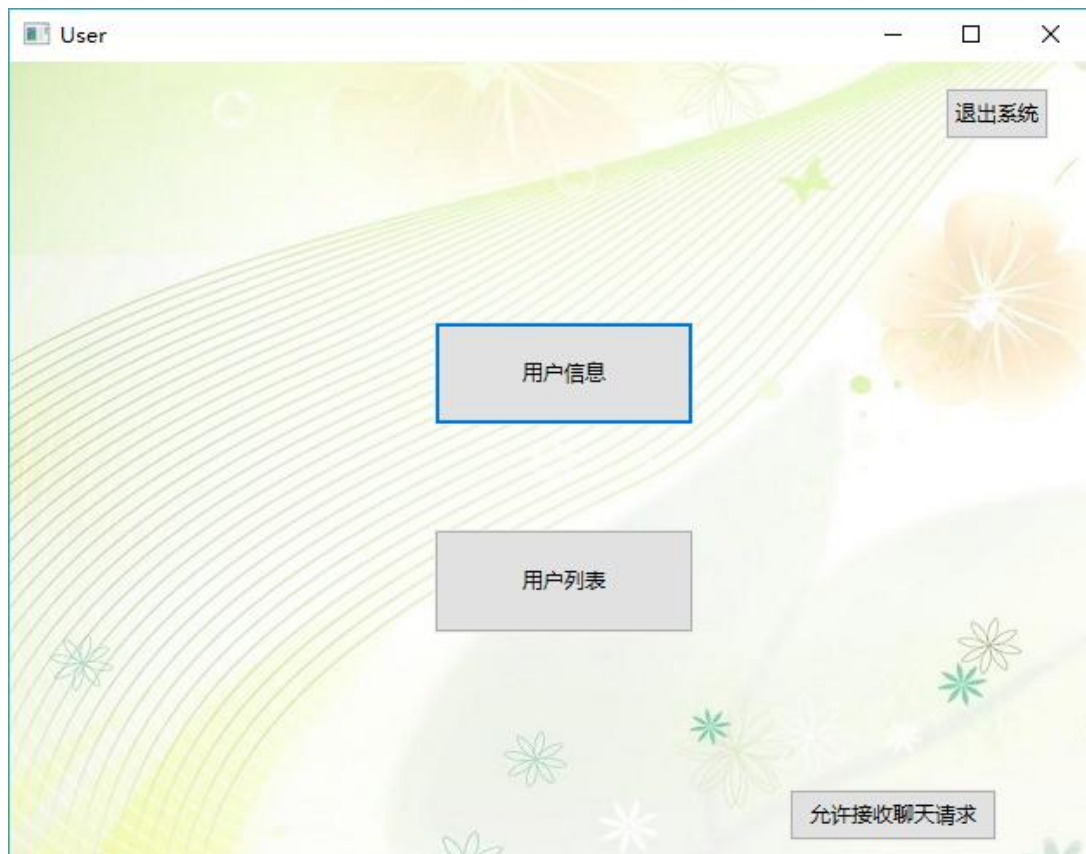
返回 退出系统

用户名

密码

登录

(4) 用户完成注册登陆后，可以选择查看个人信息或者其他用户信息。



(5) 用户可以在用户列表中选择相应用户进行聊天。



6 非常规过程

(1) 网络连接失败

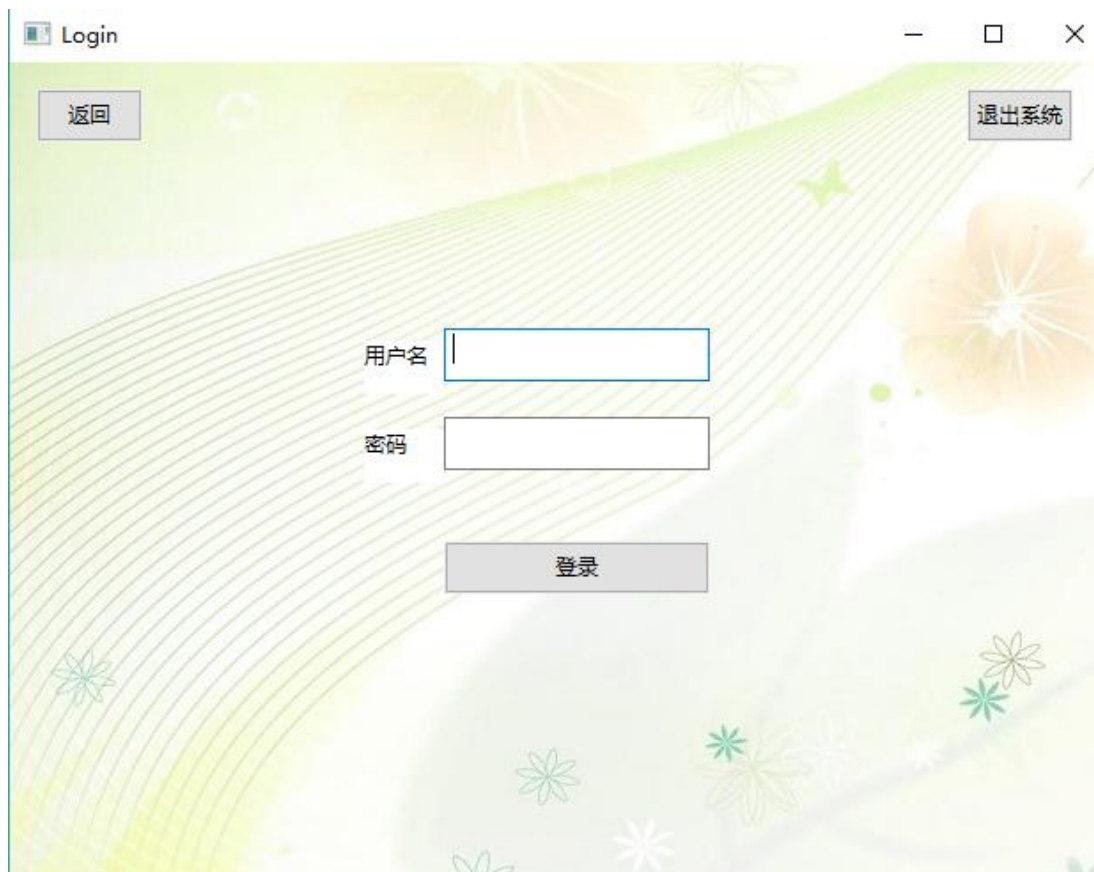
当网络连接中断时，系统的功能无法正常使用。此时，用户需要检查本机的网络连接状况，重新连接网络之后重新加载相应的功能。如果仍然无法正常使用，请重启系统。

(2) 系统停止响应

当系统受到某些外界因素影响时，可能会出现系统停止响应的情况。此时，用户需要重新启动系统。

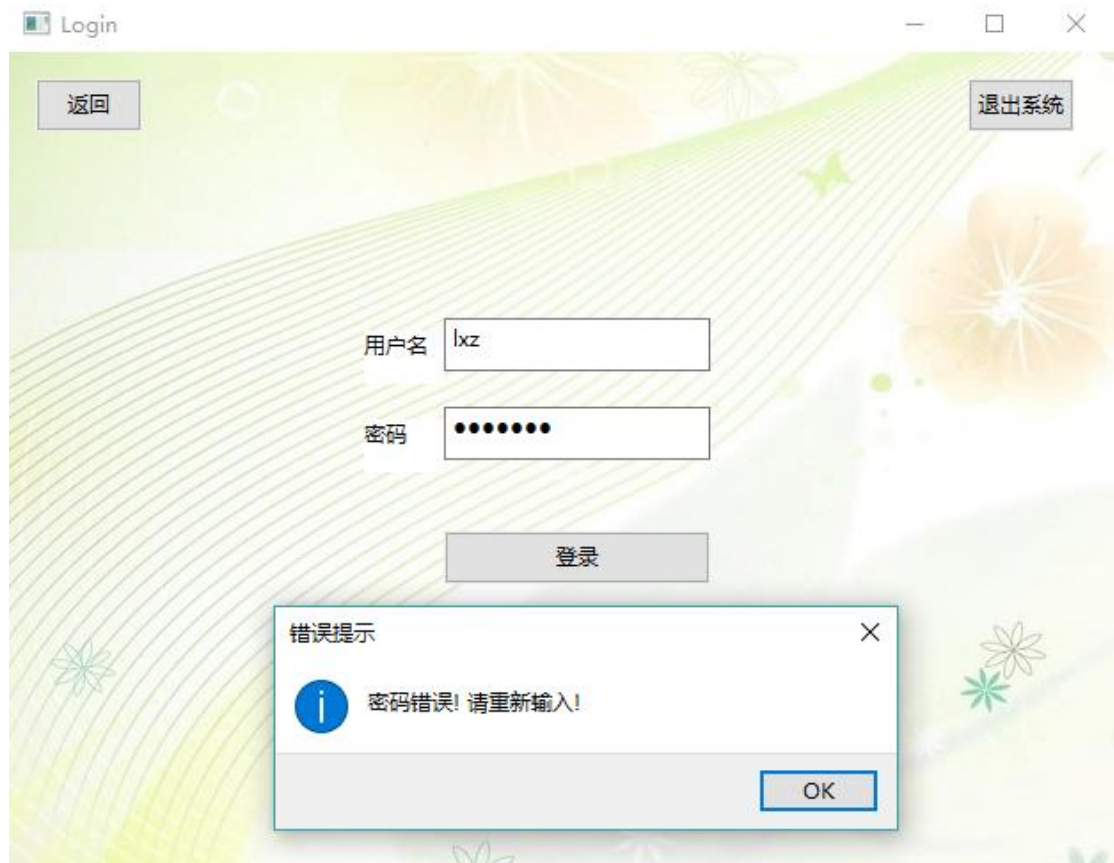
7 用户操作举例

(1) 用户登录

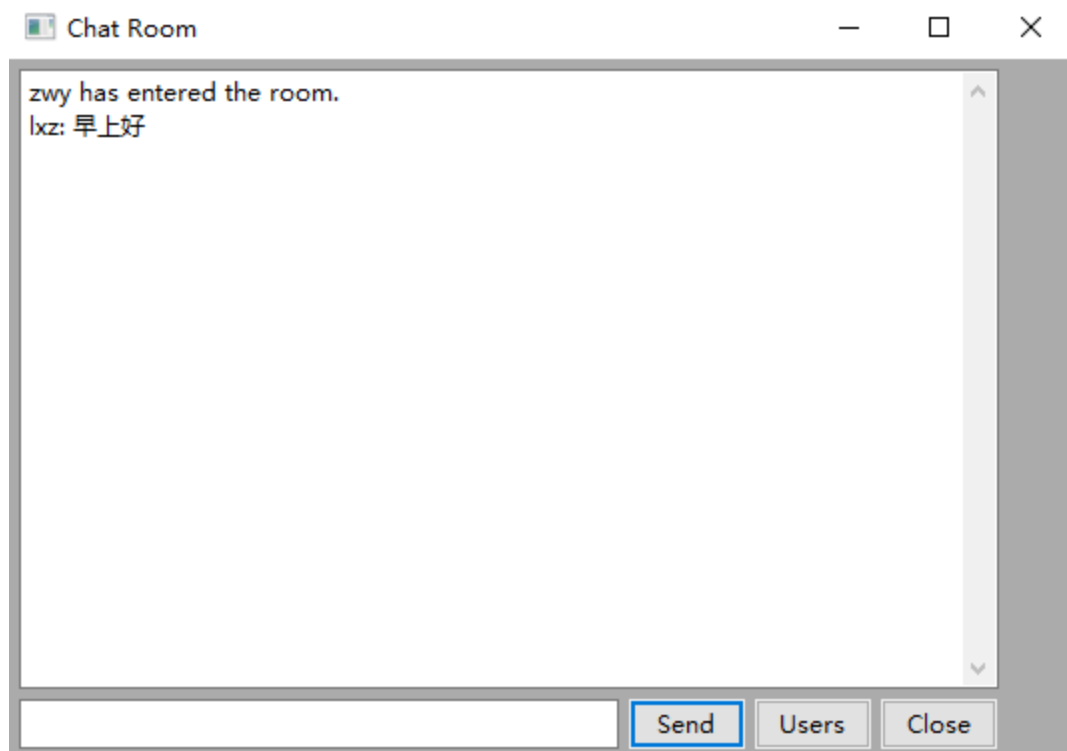


The image shows a 'Login' window with a decorative background featuring green and yellow wavy lines and small floral patterns. The window has a title bar with the text 'Login' and standard window controls (minimize, maximize, close). Inside the window, there are two buttons at the top: '返回' (Return) on the left and '退出系统' (Exit System) on the right. In the center, there are two input fields: the top one is labeled '用户名' (Username) and the bottom one is labeled '密码' (Password). Below these fields is a '登录' (Login) button.

如果用户登录时，密码输入错误，那么将弹出登录失败的提示信息。



(2) 聊天室的使用



在文本输入框中输入信息，点击 Send 即可发送消息。

系统实现代码

服务器代码

```
# -*- coding: utf-8 -*-
import asynchat
import asyncore
import time

PORT = 6666

class EndSession(Exception):
    pass

class ChatServer(asyncore.dispatcher):

    def __init__(self, port):
        asyncore.dispatcher.__init__(self)
        self.create_socket()
        self.set_reuse_addr()
        self.bind(('', port))
        self.listen(5)
        self.users = {}
        self.chat_room = {}
        for i in range(100):
            self.chat_room[i] = ChatRoom(self)
        self.main_panel = MainRoom(self)
        self.user_panel = UserRoom(self)

    def handle_accept(self):
        conn, addr = self.accept()
        ChatSession(self, conn)

class ChatSession(asynchat.async_chat):

    def __init__(self, server, sock):
        asynchat.async_chat.__init__(self, sock)
        self.server = server
        self.set_terminator(b'\n')
        self.data = []
        self.name = None
        self.enter(MainRoom(server))

    def enter(self, room):
        try:
            cur = self.room
        except AttributeError:
            pass
        else:
            cur.remove(self)
        self.room = room
        room.add(self)

    def collect_incoming_data(self, data):
```

```

        # 接收客户端的数据
        self.data.append(data.decode("utf-8"))

    def found_terminator(self):
        # 当客户端的一条数据结束时的处理
        line = ''.join(self.data)
        self.data = []
        try:
            self.room.handle(self, line.encode("utf-8"))
        except EndSession:
            self.handle_close()

    def handle_close(self):
        async_chat.async_chat.handle_close(self)
        self.enter(quitRoom(self.server))

class CommandHandler:

    def unknown(self, session, cmd):
        # 响应未知命令
        # 通过 async_chat.async_chat.push 方法发送消息
        session.push(('Unknown command {} \n'.format(cmd)).encode("utf-8"))

    def handle(self, session, line):
        line = line.decode()
        # 命令处理
        if not line.strip():
            return
        parts = line.split(' ', 1)
        cmd = parts[0]
        try:
            line = parts[1].strip()
        except IndexError:
            line = ''
        # 通过协议代码执行相应的方法
        method = getattr(self, 'do_' + cmd, None)
        try:
            method(session, line)
        except TypeError:
            self.unknown(session, cmd)

class Room(CommandHandler):

    def __init__(self, server):
        self.server = server
        self.sessions = []
        self.n_chatroom = 0

    def add(self, session):
        # 一个用户进入房间
        self.sessions.append(session)

    def remove(self, session):
        # 一个用户离开房间
        self.sessions.remove(session)

```



```

def broadcast(self, line):
    # 向所有的用户发送指定消息
    # 使用 asynchat.asyn_chat.push 方法发送数据
    for session in self.sessions:
        session.push(line)

def do_quit(self, session, line):
    raise EndSession

class MainRoom(Room):

    def add(self, session):
        # 用户连接成功的回应
        Room.add(self, session)
        # 使用 asynchat.asyn_chat.push 方法发送数据
        session.push(b'Connect Success')

    def do_login(self, session, line):
        # 用户登录逻辑
        cmd = line.split(' ')
        if len(cmd) < 2:
            session.push(b'UserName or Password Empty')
            return

        name = cmd[0]
        password = cmd[1]

        if name in self.server.users:
            session.push(b'User Login')
        else:
            rst = self.judge_user(name, password)
            if rst == 0:
                session.push(b'UserName Not Exist')
            elif rst == 1:
                session.push(b'Wrong Password')
            else:
                session.name = name
                session.enter(self.server.user_panel)

    def do_register(self, session, line):
        # 用户注册逻辑
        cmd = line.split(' ')
        if len(cmd) < 4:
            session.push(b'Some Empty')
            return

        name = cmd[0]
        password = cmd[1]
        password1 = cmd[2]
        age = cmd[3]
        if self.Exist_user(name) == True:
            session.push(b'Username Exist')
        elif password != password1:
            session.push(b'Password Not Same')
        elif int(age) < 0 or int(age) > 200:
            session.push(b'Age Error')

```

```

else:
    with open('info.log', 'a') as f:
        info = name + ' ' + password + ' ' + age + ' ' + '\n'
        f.write(info)
    session.enter(self.server.main_panel)

def judge_user(self, name, password):
    # 0:no user 1:user wrong pwd 2:right pwd
    with open('info.log', 'r') as f:
        all_info = f.readlines()
        for i in range(len(all_info)):
            all_info[i] = all_info[i].strip('\n')
            cur_name = all_info[i].split(' ')[0]
            if cur_name == name:
                cur_pwd = all_info[i].split(' ')[1]
                if password == cur_pwd:
                    f.close()
                    return 2
            else:
                f.close()
                return 1
    return 0

def Exist_user(self, name):
    # True: Username exist
    with open('info.log', 'r') as f:
        all_info = f.readlines()
        for i in range(len(all_info)):
            all_info[i] = all_info[i].strip('\n')
            cur_name = all_info[i].split(' ')[0]
            if cur_name == name:
                f.close()
                return True
    return False

class UserRoom(Room):

    def add(self, session):
        # 用户连接成功的回应
        Room.add(self, session)
        # 使用 asyncio.async_chat.push 方法发送数据
        session.push(b'Connect Success')

    def do_info(self, session, line):
        name = line.strip()
        info = None
        with open('info.log', 'r') as f:
            all_info = f.readlines()
            for i in range(len(all_info)):
                all_info[i] = all_info[i].strip('\n')
                cur_name = all_info[i].split(' ')[0]
                if cur_name == name:
                    info = all_info[i]
                    break
        session.push(info.encode("utf-8"))

    def do_list(self, session, line):

```

```

info_list = None
msg = ''
with open('info.log', 'r') as f:
    info_list = f.readlines()

for i in range(len(info_list)):
    if session.name == info_list[i].split(' ')[0]:
        continue
    msg = msg + info_list[i]
msg = msg + '|'
session.push(msg.encode("utf-8"))

def do_wchat(self, session, line):
    if self.get_spec(session.name) == 0:
        session.push(b'not spec')
        return
    for other in self.sessions:
        if self.online(line) == False:
            session.push(b'not online')
            return
        elif self.get_spec(line) == 0:
            session.push(b'other not spec')
            return
        else:
            session.enter(self.server.chat_room[self.n_chatroom])
            for other in self.sessions:
                if other.name == line:
                    other.push(b'receivev')
                    other.enter(self.server.chat_room[self.n_chatroom])

def do_video(self, session, line):
    if self.get_spec(session.name) == 0:
        session.push(b'not spec')
        return
    for other in self.sessions:
        if self.online(line) == False:
            session.push(b'not online')
            return
        elif self.get_spec(line) == 0:
            session.push(b'other not spec')
            return
        else:
            session.push(b'sendv')
            for other in self.sessions:
                if other.name == line:
                    other.push(b'receivev')

def online(self, chatname):
    for other in self.sessions:
        if other.name == chatname:
            return True
    return False

def get_spec(self, name):
    with open('info.log', 'r') as f:
        infos = f.readlines()

```

```

        for i in range(len(infos)):
            if name == infos[i].strip().split(' ')[0]:
                if infos[i].strip().split(' ')[3] == '0':
                    return 0
                else:
                    return 1

class quitRoom(Room):

    def add(self, session):
        try:
            del self.server.users[session.name]
        except KeyError:
            pass

class ChatRoom(Room):

    def add(self, session):
        # 广播新用户进入
        self.server.users[session.name] = session
        Room.add(self, session)
        self.broadcast((session.name + ' has entered the
room.\n').encode("utf-8"))

    def remove(self, session):
        # 广播用户离开
        Room.remove(self, session)
        self.broadcast((session.name + ' has left the
room.\n').encode("utf-8"))

    def do_say(self, session, line):
        # 客户端发送消息
        print(session.name, ' say: ', line)
        self.broadcast((session.name + ': ' + line + '\n').encode("utf-8"))

    def do_look(self, session, line):
        # 查看在线用户
        session.push(b'Online Users:\n')
        for other in self.sessions:
            session.push((other.name + '\n').encode("utf-8"))

    def do_logout(self, session, line):
        session.enter(self.server.user_panel)

if __name__ == '__main__':

    s = ChatServer(PORT)
    try:
        print("chat serve run at '0.0.0.0:{0}'".format(PORT))
        asyncio.loop()
    except KeyboardInterrupt:
        print("chat server exit")

```

客户端代码

client.py

```
# -*- coding:utf-8 -*-

import wx
import telnetlib
from time import sleep
import _thread as thread
from videomain import VideoChat
import threading

Port = 6666
Server_address = '101.5.138.154'

class MainFrame(wx.Frame):

    def __init__(self, parent, id, title, size):
        wx.Frame.__init__(self, parent, id, title)
        self.SetSize(size)
        self.Center()
        titleT = wx.StaticText(self, label="欢迎来到在线学习系统!", pos=(135,
45), size=(120, 30))
        font = wx.Font(25, wx.DEFAULT, wx.NORMAL, wx.BOLD)
        titleT.SetBackgroundColour("White")
        titleT.SetFont(font)
        self.registerButton = wx.Button(self, label = "注册", pos = (245,
155), size = (150, 60))
        self.loginButton = wx.Button(self, label = "登录", pos = (245, 295),
size = (150, 60))
        self.quitButton = wx.Button(self, label="退出系统", pos=(540, 15),
size=(60, 30))
        self.quitButton.Bind(wx.EVT_BUTTON, self.exit)
        self.loginButton.Bind(wx.EVT_BUTTON, self.login)
        self.registerButton.Bind(wx.EVT_BUTTON, self.register)
        self.Bind(wx.EVT_ERASE_BACKGROUND, self.OnEraseBack)
        self.Show()

    def OnEraseBack(self, event):
        dc = event.GetDC()
        if not dc:
            dc = wx.ClientDC(self)
            rect = self.GetUpdateRegion().GetBox()
            dc.SetClippingRect(rect)
        dc.Clear()
        bmp = wx.Bitmap("background.jpg")
        dc.DrawBitmap(bmp, 0, 0)

    def login(self, event):
        self.Close()
        LoginFrame(None, 2, title="Login", size=(640, 500))

    def register(self, event):
        self.Close()
        RegisterFrame(None, 3, title='Register', size=(640, 500))
```

```

def exit(self, event):
    self.Close()

class RegisterFrame(wx.Frame):

    def __init__(self, parent, id, title, size):
        wx.Frame.__init__(self, parent, id, title)
        self.SetSize(size)
        self.Center()
        self.userNameLabel = wx.StaticText(self, label = "用户名", pos = (200,
77), size = (45, 30))
        self.userName = wx.TextCtrl(self, pos = (245, 70), size = (150, 30))
        self.pwdLabel = wx.StaticText(self, label = "密码", pos = (200, 127),
size = (45, 30))
        self.password = wx.TextCtrl(self, pos = (245, 120), size = (150, 30),
style = wx.TE_PASSWORD)
        self.pwd1Label = wx.StaticText(self, label="确认密码", pos=(175,
177), size=(70, 30))
        self.password1 = wx.TextCtrl(self, pos=(245, 170), size=(150, 30),
style = wx.TE_PASSWORD)
        self.ageLabel = wx.StaticText(self, label = "年龄", pos = (200, 227),
size = (45, 30))
        self.age = wx.TextCtrl(self, pos = (245, 220), size = (150, 30))
        self.userNameLabel.SetBackgroundColour("White")
        self.pwdLabel.SetBackgroundColour("White")
        self.pwd1Label.SetBackgroundColour("White")
        self.ageLabel.SetBackgroundColour("White")
        self.backButton = wx.Button(self, label="返回", pos=(15, 15),
size=(60, 30))
        self.backButton.Bind(wx.EVT_BUTTON, self.back)
        self.quitButton = wx.Button(self, label="退出系统", pos=(540, 15),
size=(60, 30))
        self.quitButton.Bind(wx.EVT_BUTTON, self.exit)
        self.RegisterButton = wx.Button(self, label='注册', pos=(245, 350),
size=(150, 30))
        self.RegisterButton.Bind(wx.EVT_BUTTON, self.register)
        self.Bind(wx.EVT_ERASE_BACKGROUND, self.OnEraseBack)
        self.Show()

    def back(self, event):
        self.Close()
        MainFrame(None, id = wx.ID_ANY, title="Online Learning System",
size=(640, 500))

    def OnEraseBack(self, event):
        dc = event.GetDC()
        if not dc:
            dc = wx.ClientDC(self)
            rect = self.GetUpdateRegion().GetBox()
            dc.SetClippingRect(rect)
        dc.Clear()
        bmp = wx.Bitmap("background.jpg")
        dc.DrawBitmap(bmp, 0, 0)

    def register(self, event):    # 判定两次密码是否一致, 信息写入

```

```

# 登录处理
try:
    con.open(Server_address, Port, timeout=10)
    response = con.read_some()
    if response != b'Connect Success':
        self.showDialog("错误提示", "连接失败")
        return
    con.write(('register ' + str(self.userName.GetLineText(0)) + '
'
            + str(self.password.GetLineText(0)) + '
'
            + str(self.password1.GetLineText(0)) + '
'
            + str(self.age.GetLineText(0)) + '
\n').encode("utf-8"))
    response = con.read_some()
    if response == b'Some Empty':
        self.showDialog("错误提示", "有选项为空！请填写完整!")
    elif response == b'Username Exist':
        self.showDialog("错误提示", "该用户名已存在!")
    elif response == b'Password Not Same':
        self.showDialog("错误提示", "两次密码输入不一致！请重新输入!")
    elif response == b'Age Error':
        self.showDialog("错误提示", "所填年龄不在合法范围之内！请重新
输入!")
    else:
        self.showDialog("提示信息", "恭喜注册成功!")
        self.Close()
        MainFrame(None, 5, title='Online Learning System', size=(640,
500))
except Exception:
    self.showDialog("错误提示", "连接失败")

def showDialog(self, title, content):
    box = wx.MessageDialog(None, content, title, wx.OK)
    answer = box.ShowModal()
    box.Destroy()

def exit(self, event):
    self.Close()

class LoginFrame(wx.Frame):

    def __init__(self, parent, id, title, size):
        wx.Frame.__init__(self, parent, id, title)
        self.SetSize(size)
        self.Center()
        self.userNameLabel = wx.StaticText(self, label = "用户名", pos = (200,
157), size = (45, 30))
        self.userName = wx.TextCtrl(self, pos = (245, 150), size=(150, 30))
        self.pwdLabel = wx.StaticText(self, label = "密码", pos = (200, 207),
size = (45, 30))
        self.password = wx.TextCtrl(self, pos = (245, 200), size = (150, 30),
style = wx.TE_PASSWORD)
        self.loginButton = wx.Button(self, label = "登录", pos = (245, 270),
size = (150, 30))
        self.backButton = wx.Button(self, label="返回", pos=(15, 15),

```

```

size=(60, 30))
    self.quitButton = wx.Button(self, label="退出系统", pos=(540, 15),
size=(60, 30))
    self.quitButton.Bind(wx.EVT_BUTTON, self.exit)
    self.backButton.Bind(wx.EVT_BUTTON, self.back)
    self.loginButton.Bind(wx.EVT_BUTTON, self.login)
    self.Bind(wx.EVT_ERASE_BACKGROUND, self.OnEraseBack)
    self.userNameLabel.SetBackgroundColour("White")
    self.pwdLabel.SetBackgroundColour("White")
    self.Show()

def OnEraseBack(self, event):
    dc = event.GetDC()
    if not dc:
        dc = wx.ClientDC(self)
        rect = self.GetUpdateRegion().GetBox()
        dc.SetClippingRect(rect)
    dc.Clear()
    bmp = wx.Bitmap("background.jpg")
    dc.DrawBitmap(bmp, 0, 0)

def login(self, event):
    # 登录处理
    try:
        con.open(Server_address, Port, timeout=10)
        response = con.read_some()
        if response != b'Connect Success':
            self.showDialog("错误提示", "连接失败")
            return
        con.write(('login ' + str(self.userName.GetLineText(0)) + ' '
                  + str(self.password.GetLineText(0)) +
'\n').encode("utf-8"))
        response = con.read_some()
        if response == b'UserName or Password Empty':
            self.showDialog("错误提示", "用户名或密码为空!")
        elif response == b'User Login':
            self.showDialog("错误提示", "该用户已登录系统!")
        elif response == b'UserName Not Exist':
            self.showDialog("错误提示", "该用户不存在!")
        elif response == b'Wrong Password':
            self.showDialog("错误提示", "密码错误! 请重新输入!")
        else:
            self.Close()
            UserPanel(None, 4, 'User', (640, 500),
str(self.userName.GetLineText(0)))
    except Exception:
        self.showDialog("错误提示", "连接失败")

def showDialog(self, title, content):
    box = wx.MessageDialog(None, content, title, wx.OK)
    answer = box.ShowModal()
    box.Destroy()

def back(self, event):
    self.Close()

```



```

        MainFrame(None, id=wx.ID_ANY, title="Online Learning System",
size=(640, 500))

    def exit(self, event):
        self.Close()

class UserPanel(wx.Frame):

    def __init__(self, parent, id, title, size, username):
        wx.Frame.__init__(self, parent, id, title)
        self.username = username
        self.SetSize(size)
        self.Center()
        self.infoButton = wx.Button(self, label="用户信息", pos=(245, 150),
size=(150, 60))
        self.listButton = wx.Button(self, label="用户列表", pos=(245, 270),
size=(150, 60))
        self.receiveButton = wx.Button(self, label="允许接收聊天请求",
pos=(450, 420), size=(120, 30))
        self.quitButton = wx.Button(self, label="退出系统", pos=(540, 15),
size=(60, 30))
        self.quitButton.Bind(wx.EVT_BUTTON, self.exit)
        self.infoButton.Bind(wx.EVT_BUTTON, self.getinfo)
        self.listButton.Bind(wx.EVT_BUTTON, self.getlist)
        self.receiveButton.Bind(wx.EVT_BUTTON, self.allow)
        self.Bind(wx.EVT_ERASE_BACKGROUND, self.OnEraseBack)
        self.Show()

    def OnEraseBack(self, event):
        dc = event.GetDC()
        if not dc:
            dc = wx.ClientDC(self)
            rect = self.GetUpdateRegion().GetBox()
            dc.SetClippingRect(rect)
        dc.Clear()
        bmp = wx.Bitmap("background.jpg")
        dc.DrawBitmap(bmp, 0, 0)

    def getinfo(self, event):
        con.write(('info ' + self.username + '\n').encode("utf-8"))
        res = con.read_some().decode("utf-8").split(' ')
        InfoFrame(None, 7, "User Info", (640, 500), res)

    def getlist(self, event):
        con.write(('list \n').encode("utf-8"))
        res = con.read_until(b'|').decode("utf-8")
        UserList(None, 11, "User List", (640, 500), res)

    def allow(self, event):

        if self.is_spec(self.username) == 0:
            self.showDialog("错误提示", "您还不是会员，需要充值后才能执行该操
作!")

        return
        response = con.read_some()
        if response == b'receivew':

```

```

        ChatFrame(None, id=wx.ID_ANY, title="Chat Room", size=(550,
390))
        elif response == b'receivev':
            vc = VideoChat()
            vc.do_videochat()

    def is_spec(self, name):
        with open('info.log', 'r') as f:
            info = f.readlines()
            for i in range(len(info)):
                if name == info[i].strip().split(' ')[0]:
                    if info[i].strip().split(' ')[3] == '0':
                        return 0
                    else:
                        return 1

    def exit(self, event):
        self.Close()

    def showDialog(self, title, content):
        box = wx.MessageDialog(None, content, title, wx.OK)
        answer = box.ShowModal()
        box.Destroy()

class UserList(wx.Frame):
    def __init__(self, parent, id, title, size, res):
        wx.Frame.__init__(self, parent, id, title)
        self.SetSize(size)
        self.Center()
        user_list = res.strip('|').strip().split('\n')
        self.scroller = wx.ScrolledWindow(self, -1)
        self.scroller.SetScrollbars(0, 1, 0, 160 * len(user_list) + 30)
        titleT = wx.StaticText(self.scroller, label="用户列表", pos=(270,
15), size=(100, 30))
        font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.BOLD)
        titleT.SetFont(font)
        for i in range(len(user_list)):
            tinfo = user_list[i].split(' ')
            tname = tinfo[0]
            tage = tinfo[2]
            if tinfo[3] == '0':
                tspec = "否"
            else:
                tspec = "是"
            ypos = (130 + 30) * i + 60
            usrT = wx.StaticText(self.scroller, label="用户名: " + tname, pos
= (70, ypos - 5), size=(100, 30))
            ageT = wx.StaticText(self.scroller, label="年龄: " + tage,
pos=(120, ypos + 30), size=(100, 30))
            specT = wx.StaticText(self.scroller, label="是否会员: " + tspec,
pos=(120, ypos + 65), size=(100, 30))
            font = wx.Font(12, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
            usrT.SetFont(font)
            ageT.SetFont(font)
            specT.SetFont(font)
            wchatButton = wx.Button(self.scroller, label="和 " + tname + "

```

```

文字聊天", pos=(370, ypos + 15), size=(100, 30))
    wchatButton.Bind(wx.EVT_BUTTON, self.wchat)
    vchatButton = wx.Button(self.scroller, label="和 " + tname + "
视频聊天", pos=(370, ypos + 55), size=(100, 30))
    vchatButton.Bind(wx.EVT_BUTTON, self.vchat)
    self.backButton = wx.Button(self.scroller, label="返回", pos=(15,
15), size=(60, 30))
    self.backButton.Bind(wx.EVT_BUTTON, self.back)
    self.Show()

    def wchat(self, event):
        label = event.GetEventObject().GetLabel()
        name = label.split(' ')[1]
        con.write(('wchat ' + name + '\n').encode("utf-8"))
        response = con.read_some()
        # print("res:", response)
        if response == b'not spec':
            self.showDialog("错误提示", "您还不是会员, 需要充值后才能执行该操
作!")
        elif response == b'not online':
            self.showDialog("错误提示", "该用户不在线!")
        elif response == b'other not spec':
            self.showDialog("错误提示", "该用户不是会员, 无法与其聊天!")
        else:
            ChatFrame(None, id=wx.ID_ANY, title="Chat Room", size=(550,
390))

    def vchat(self, event):
        label = event.GetEventObject().GetLabel()
        name = label.split(' ')[1]
        con.write(('video ' + name + '\n').encode("utf-8"))
        response = con.read_some()
        # print("res:", response)
        if response == b'not spec':
            self.showDialog("错误提示", "您还不是会员, 需要充值后才能执行该操
作!")
        elif response == b'not online':
            self.showDialog("错误提示", "该用户不在线!")
        elif response == b'other not spec':
            self.showDialog("错误提示", "该用户不是会员, 无法与其聊天!")
        elif response == b'sendv':
            vc = VideoChat()
            vc.do_videochat()

    def back(self, event):
        self.Close()

    def showDialog(self, title, content):
        box = wx.MessageDialog(None, content, title, wx.OK)
        answer = box.ShowModal()
        box.Destroy()

class ChatFrame(wx.Frame):

    def __init__(self, parent, id, title, size):

```

```

# 初始化, 添加控件并绑定事件
wx.Frame.__init__(self, parent, id, title)
self.SetSize(size)
self.Center()
self.chatFrame = wx.TextCtrl(self, pos=(5, 5), size=(490, 310),
style=wx.TE_MULTILINE | wx.TE_READONLY)
self.message = wx.TextCtrl(self, pos=(5, 320), size=(300, 25))
self.sendButton = wx.Button(self, label="Send", pos=(310, 320),
size=(58, 25))
self.usersButton = wx.Button(self, label="Users", pos=(373, 320),
size=(58, 25))
self.closeButton = wx.Button(self, label="Close", pos=(436, 320),
size=(58, 25))
# 发送按钮绑定发送消息方法
self.sendButton.Bind(wx.EVT_BUTTON, self.send)
# Users 按钮绑定获取在线用户数量方法
self.usersButton.Bind(wx.EVT_BUTTON, self.lookUsers)
# 关闭按钮绑定关闭方法
self.closeButton.Bind(wx.EVT_BUTTON, self.close)
thread.start_new_thread(self.receive, ())
self.Show()

def send(self, event):
# 发送消息
message = str(self.message.GetLineText(0)).strip()
if message != '':
    con.write(('say ' + message + '\n').encode("utf-8"))
    self.message.Clear()

def lookUsers(self, event):
# 查看当前在线用户
con.write(b'look\n')

def close(self, event):
# 关闭窗口
con.write(b'logout\n')
# con.close()
self.Close()

def receive(self):
# 接受服务器的消息
while True:
    sleep(0.6)
    result = con.read_very_eager()
    if result != '':
        self.chatFrame.AppendText(result)

class InfoFrame(wx.Frame):
def __init__(self, parent, id, title, size, info):
wx.Frame.__init__(self, parent, id, title)
self.SetSize(size)
self.Center()
self.username = info[0]
self.pwd = info[1]

```

```

        self.age = info[2]
        if info[3] == '0':
            self.spec = "否"
        else:
            self.spec = "是"
        self.usrT = wx.StaticText(self, label = "用户名: " + self.username,
pos = (210, 100), size = (100, 30), style = wx.ALIGN_CENTER)
        self.pwdT = wx.StaticText(self, label = "密码: " + self.pwd, pos=(210,
170), size=(100, 30), style = wx.ALIGN_CENTER)
        self.ageT = wx.StaticText(self, label = "年龄: " + self.age, pos=(210,
240), size=(100, 30), style = wx.ALIGN_CENTER)
        self.specT = wx.StaticText(self, label = "是否会员: " + self.spec,
pos=(210, 310), size=(100, 30), style = wx.ALIGN_CENTER)
        font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.NORMAL)
        self.usrT.SetFont(font)
        self.pwdT.SetFont(font)
        self.ageT.SetFont(font)
        self.specT.SetFont(font)
        self.usrT.SetBackgroundColour("White")
        self.pwdT.SetBackgroundColour("White")
        self.ageT.SetBackgroundColour("White")
        self.specT.SetBackgroundColour("White")
        self.payButton = wx.Button(self, label="会员充值", pos=(360, 305),
size=(60, 30))
        self.payButton.Bind(wx.EVT_BUTTON, self.pay)
        self.backButton = wx.Button(self, label="返回", pos=(285, 395),
size=(60, 30))
        self.backButton.Bind(wx.EVT_BUTTON, self.back)
        self.Bind(wx.EVT_ERASE_BACKGROUND, self.OnEraseBack)
        self.Show()

    def pay(self, event):
        payFrame(None, 10, "payment", (300, 300))

    def back(self, event):
        self.Close()

    def OnEraseBack(self, event):
        dc = event.GetDC()
        if not dc:
            dc = wx.ClientDC(self)
            rect = self.GetUpdateRegion().GetBox()
            dc.SetClippingRect(rect)
        dc.Clear()
        bmp = wx.Bitmap("background.jpg")
        dc.DrawBitmap(bmp, 0, 0)

class payFrame(wx.Frame):

    def __init__(self, parent, id, title, size):
        wx.Frame.__init__(self, parent, id, title)
        self.SetSize(size)
        self.Center()
        mypanel = wx.Panel(self, 8, size = size)
        image = wx.Image('pay.png',wx.BITMAP_TYPE_PNG)

```

```

        mypic = image.ConvertToBitmap()
        wx.StaticBitmap(mypanel, 9, bitmap = mypic, pos = (95, 60))
        self.backButton = wx.Button(mypanel, label = "返回", pos=(115, 200),
size=(60, 30))
        self.backButton.Bind(wx.EVT_BUTTON, self.back)
        self.Show()

    def back(self, event):
        self.Close()

if __name__ == '__main__':
    app = wx.App()
    con = telnetlib.Telnet()
    MainFrame(None, -1, title = "Online Learning System", size = (640, 500))
    app.MainLoop()

```

videomain.py

```

import sys
import time
import argparse
from vchat import Video_Server, Video_Client
from achat import Audio_Server, Audio_Client

class VideoChat():

    def __init__(self):
        self.parser = argparse.ArgumentParser()
        self.parser.add_argument('--host', type=str,
default='101.5.140.114')
        self.parser.add_argument('--port', type=int, default=10087)
        self.parser.add_argument('--noself', type=bool, default=False)
        self.parser.add_argument('--level', type=int, default=1)
        self.parser.add_argument('-v', '--version', type=int, default=4)
        self.args = self.parser.parse_args()
        self.IP = self.args.host
        self.PORT = self.args.port
        self.VERSION = self.args.version
        self.SHOWME = not self.args.noself
        self.LEVEL = self.args.level
        self.vclient = Video_Client(self.IP, self.PORT + 2, self.SHOWME,
self.LEVEL, self.VERSION)
        self.vserver = Video_Server(self.PORT, self.VERSION)
        self.aclient = Audio_Client(self.IP, self.PORT + 4, self.VERSION)
        self.aserver = Audio_Server(self.PORT + 3, self.VERSION)

    def do_videochat(self):
        self.vclient.start()
        self.vserver.start()
        self.aclient.start()
        self.aserver.start()
        while True:
            time.sleep(1)

```

vchat.py

```

from socket import *
import threading
import cv2
import sys
import struct
import pickle
import time
import zlib
import numpy as np

class Video_Server(threading.Thread):

    def __init__(self, port, version) :
        threading.Thread.__init__(self)
        self.setDaemon(True)
        self.ADDR = ('', port)
        if version == 4:
            self.sock = socket(AF_INET ,SOCK_STREAM)
        else:
            self.sock = socket(AF_INET6 ,SOCK_STREAM)

    def __del__(self):
        self.sock.close()
        try:
            cv2.destroyAllWindows()
        except:
            pass

    def run(self):
        print("VEDIO server starts...")
        self.sock.bind(self.ADDR)
        self.sock.listen(1)
        conn, addr = self.sock.accept()
        print("remote VEDIO client success connected...")
        data = "".encode("utf-8")
        payload_size = struct.calcsize("L")
        cv2.namedWindow('Remote', cv2.WINDOW_NORMAL)
        while True:
            while len(data) < payload_size:
                data += conn.recv(81920)
            packed_size = data[:payload_size]
            data = data[payload_size:]
            msg_size = struct.unpack("L", packed_size)[0]
            while len(data) < msg_size:
                data += conn.recv(81920)
            zframe_data = data[:msg_size]
            data = data[msg_size:]
            frame_data = zlib.decompress(zframe_data)
            frame = pickle.loads(frame_data)
            cv2.imshow('Remote', frame)
            if cv2.waitKey(1) & 0xFF == 27:
                break

class Video_Client(threading.Thread):
    def __init__(self ,ip, port, showme, level, version):
        threading.Thread.__init__(self)
        self.setDaemon(True)

```

```

self.ADDR = (ip, port)
self.showme = showme
if level == 0:
    self.interval = 0
elif level == 1:
    self.interval = 1
elif level == 2:
    self.interval = 2
else:
    self.interval = 3
self.fx = 1 / (self.interval + 1)
if self.fx < 0.3:
    self.fx = 0.3
if version == 4:
    self.sock = socket(AF_INET, SOCK_STREAM)
else:
    self.sock = socket(AF_INET6, SOCK_STREAM)
self.cap = cv2.VideoCapture(0)
print("VEDIO client starts...")

def __del__(self) :
    self.sock.close()
    self.cap.release()
    if self.showme:
        try:
            cv2.destroyAllWindows()
        except:
            pass

def run(self):
    while True:
        try:
            self.sock.connect(self.ADDR)
            break
        except:
            time.sleep(3)
            continue
    if self.showme:
        cv2.namedWindow('You', cv2.WINDOW_NORMAL)
    print("VEDIO client connected...")
    while self.cap.isOpened():
        ret, frame = self.cap.read()
        sframe = cv2.resize(frame, (0, 0), fx=self.fx, fy=self.fx)
        if self.showme:
            cv2.imshow('You', sframe)
            if cv2.waitKey(1) & 0xFF == 27:
                self.showme = False
                cv2.destroyAllWindows()
        # sframe = cv2.resize(frame, (0, 0), fx=self.fx, fy=self.fx)
        data = pickle.dumps(frame)
        zdata = zlib.compress(data, zlib.Z_BEST_COMPRESSION)
        try:
            self.sock.sendall(struct.pack("L", len(zdata)) + zdata)
        except:
            break

```



```

        for i in range(self.interval):
            self.cap.read()

```

achat.py

```

from socket import *
import threading
import pyaudio
import wave
import sys
import zlib
import struct
import pickle
import time
import numpy as np

CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 0.5

class Audio_Server(threading.Thread):
    def __init__(self, port, version) :
        threading.Thread.__init__(self)
        self.setDaemon(True)
        self.ADDR = ('', port)
        if version == 4:
            self.sock = socket(AF_INET ,SOCK_STREAM)
        else:
            self.sock = socket(AF_INET6 ,SOCK_STREAM)
        self.p = pyaudio.PyAudio()
        self.stream = None
    def __del__(self):
        self.sock.close()
        if self.stream is not None:
            self.stream.stop_stream()
            self.stream.close()
        self.p.terminate()
    def run(self):
        print("AUDIO server starts...")
        self.sock.bind(self.ADDR)
        self.sock.listen(1)
        conn, addr = self.sock.accept()
        print("remote AUDIO client success connected...")
        data = "".encode("utf-8")
        payload_size = struct.calcsize("L")
        self.stream = self.p.open(format=FORMAT,
                                   channels=CHANNELS,
                                   rate=RATE,
                                   output=True,
                                   frames_per_buffer = CHUNK
                                   )

        while True:
            while len(data) < payload_size:

```

```

        data += conn.recv(81920)
        packed_size = data[:payload_size]
        data = data[payload_size:]
        msg_size = struct.unpack("L", packed_size)[0]
        while len(data) < msg_size:
            data += conn.recv(81920)
        frame_data = data[:msg_size]
        data = data[msg_size:]
        frames = pickle.loads(frame_data)
        for frame in frames:
            self.stream.write(frame, CHUNK)

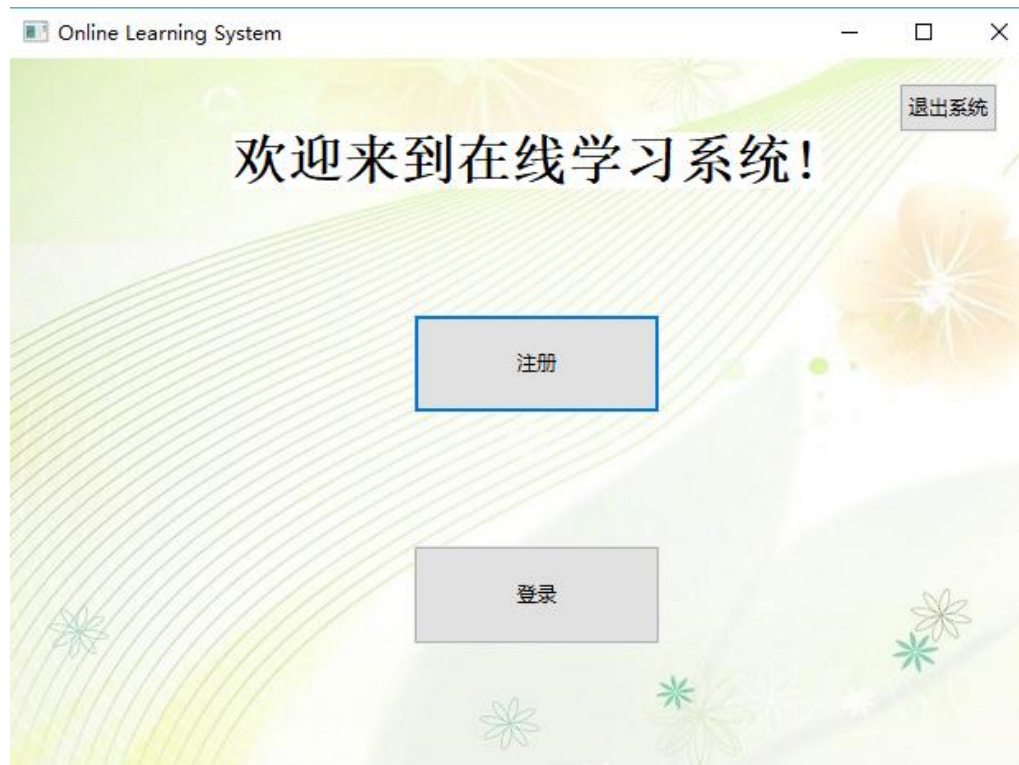
class Audio_Client(threading.Thread):
    def __init__(self, ip, port, version):
        threading.Thread.__init__(self)
        self.setDaemon(True)
        self.ADDR = (ip, port)
        if version == 4:
            self.sock = socket(AF_INET, SOCK_STREAM)
        else:
            self.sock = socket(AF_INET6, SOCK_STREAM)
        self.p = pyaudio.PyAudio()
        self.stream = None
        print("AUDIO client starts...")
    def __del__(self):
        self.sock.close()
        if self.stream is not None:
            self.stream.stop_stream()
            self.stream.close()
        self.p.terminate()
    def run(self):
        while True:
            try:
                self.sock.connect(self.ADDR)
                break
            except:
                time.sleep(3)
                continue
        print("AUDIO client connected...")
        self.stream = self.p.open(format=FORMAT,
                                   channels=CHANNELS,
                                   rate=RATE,
                                   input=True,
                                   frames_per_buffer=CHUNK)
        while self.stream.is_active():
            frames = []
            for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
                data = self.stream.read(CHUNK)
                frames.append(data)
            senddata = pickle.dumps(frames)
            try:
                self.sock.sendall(struct.pack("L", len(senddata)) +
senddata)
            except:
                break

```

系统运行与测试

1 系统运行

1.1 系统主界面



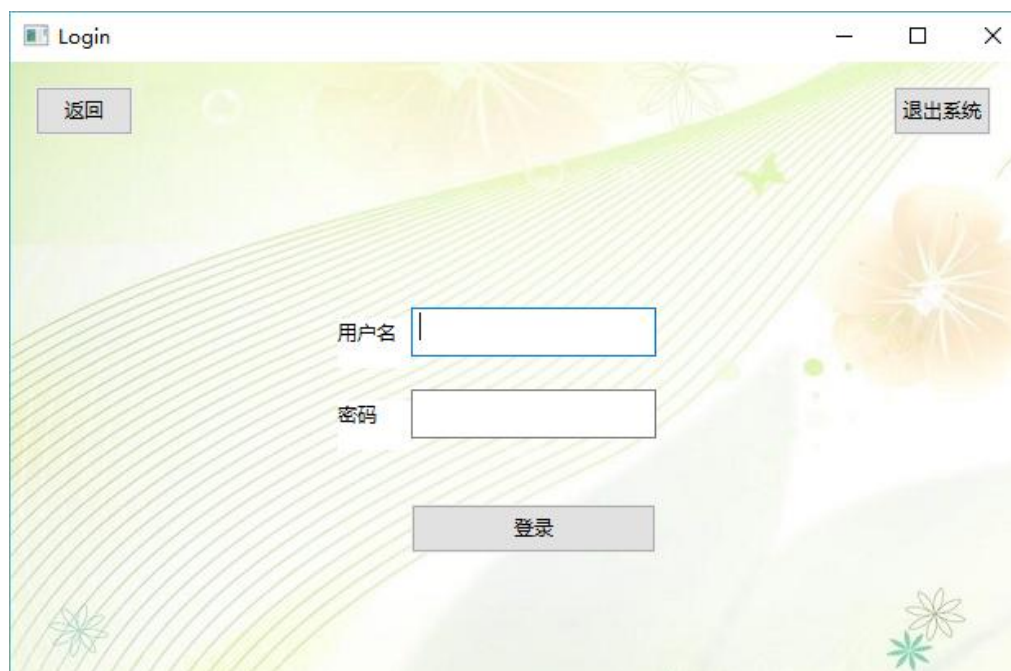
用户进入系统主界面可以选择进行注册或者登陆。

1.2 用户注册



用户在注册界面输入用户名、密码、确认密码、年龄信息即可进行注册。

1.3 用户登录



用户在登录界面输入用户名和相应密码即可进行登录。

1.4 用户界面



用户登录成功后进入用户主界面，在该界面用户可以查看本用户信息或者查看系统中所有用户情况。

1.5 用户信息



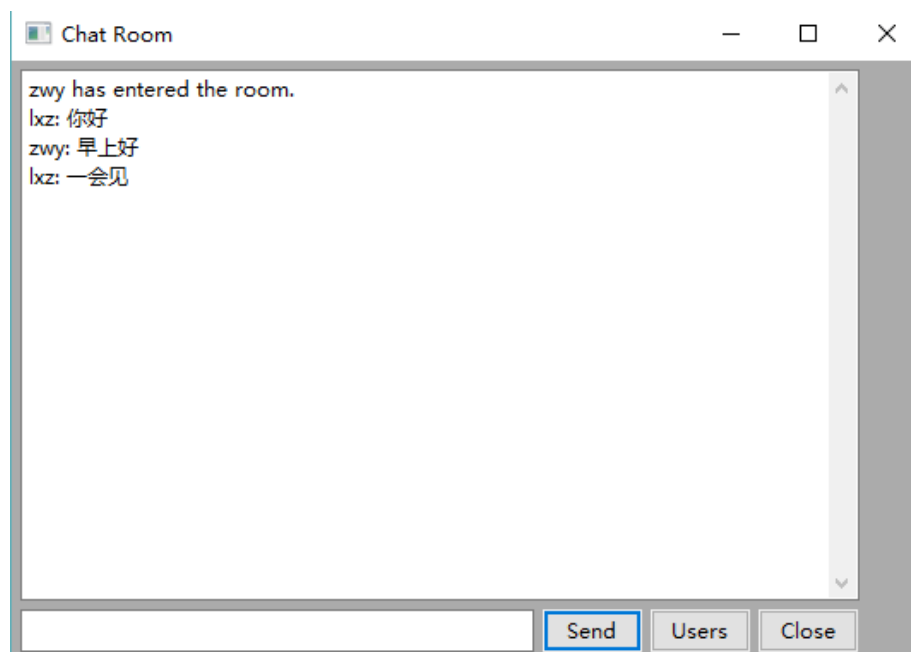
用户在用户信息界面将看到自己的全部信息，并可以在这里进行会员充值。

1.6 用户列表



用户在用户列表界面可以看到整个系统中除了自己以外的所有用户的相关信息，可以通过窗口右侧的滚动条向下滑动查看所有的用户信息。在该界面，用户可以选择和系统中其他用户进行文字聊天或者视频聊天。

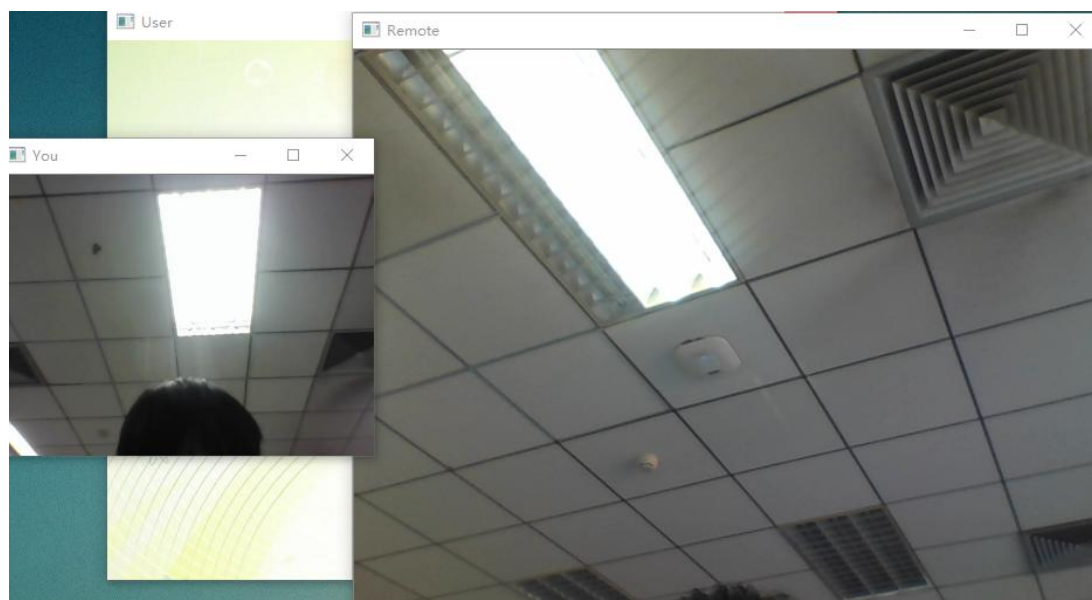
1.7 文字聊天



用户进入文字聊天室以后，可以在下方输入想要聊天的内容，点击 Send 即可发送聊天内容，聊天内容会被对方接收。

1.8 视频聊天

用户向对方发送视频聊天请求后，如果双方符合视频聊天的条件，那么双方用户可以进入视频聊天界面。



在视频聊天界面中，Remote 窗口显示对方摄像头中的视频，You 窗口显示用户自身的摄像头视频。

2 系统测试

2.1 系统主界面

用户打开本系统之后，客户端将与本系统服务器端连接，如果连接失败，系统将提示连接失败的信息。

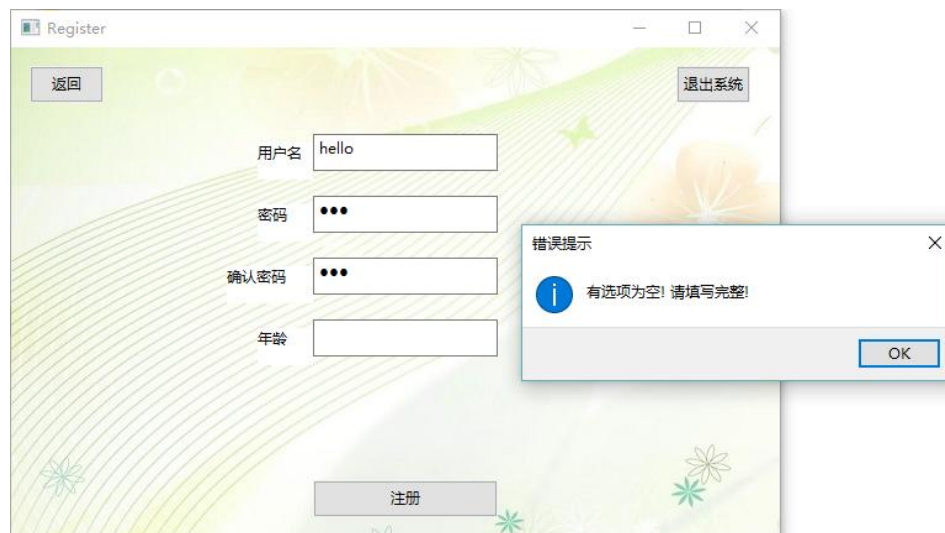
2.2 用户注册

用户在注册过程中需要输入用户的用户名、密码、确认密码以及年龄信息，系统将检查用户输入是否符合要求，如果符合要求则将信息存储完成注册。在系统测试过程中，测试用户注册过程可能出现的各种错误情况。

用户提交注册信息后，系统将信息上传服务器，对信息进行判断。首先检查用户名是否已被注册过，如果已被注册，则系统提示“用户名已存在”错误信息。



在注册过程中，所有信息栏必须全部填写，不能为空，否则系统将提示“有选项为空”错误信息。



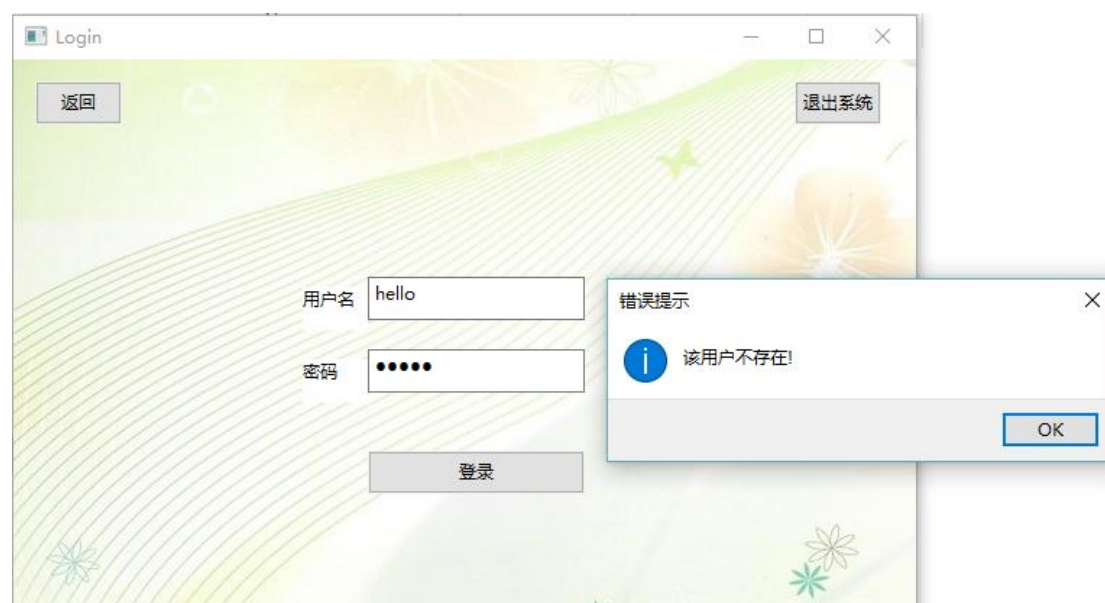
注册时，密码与确认密码必须填写一致，否则系统将提示“两次输入密码不一致”错误信息。



2.2 用户登录

用户登录时需要输入用户名和相应的密码，系统检查用户名是否存在，用户名和密码是否匹配等条件，如果检查无误，则用户可以正常登录。在系统测试过程中，测试用户登录时可能出现的各种错误情况。

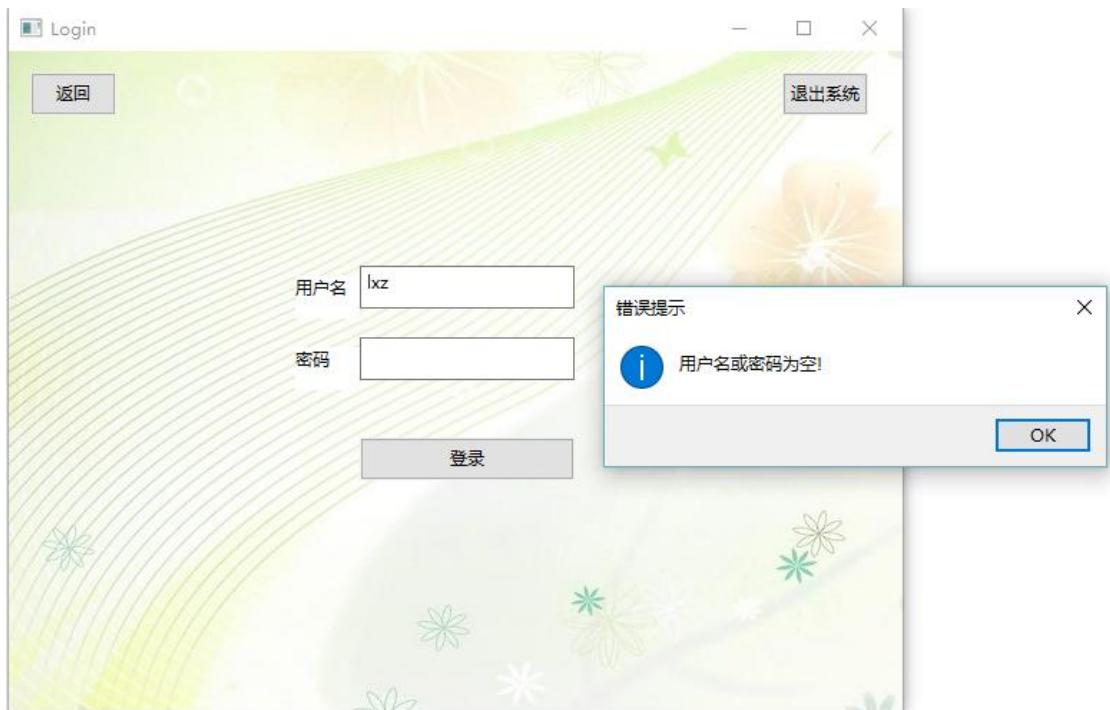
如果用户登录时还未注册，系统将提示“用户不存在”错误信息。



如果用户登录时用户名与密码不匹配，则系统将提示“密码错误”错误信息。



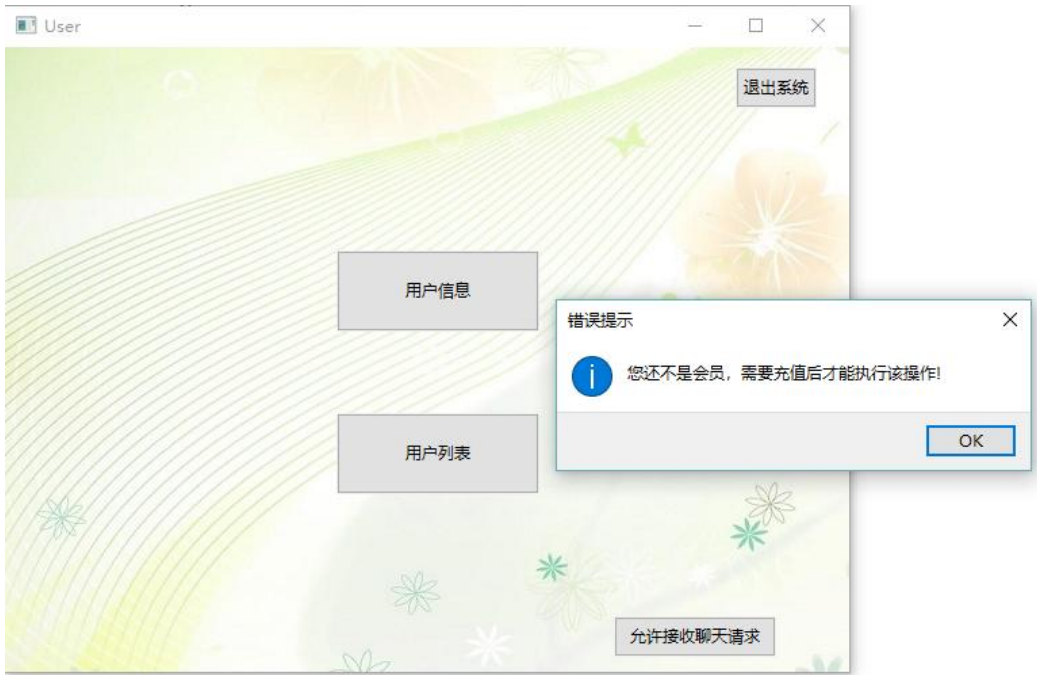
如果用户登录时，用户名和密码中某一项未填写，则系统将显示“用户名或密码为空”错误信息。



2.3 用户接收聊天请求

用户进入用户界面后，如果允许接收他人的聊天请求，那么可以点击下方“允许接收聊天请求”按钮，此时系统将检查用户是否为会员，如果用户为会员则监听其他用户的聊天请求，否则将提示用户“您不是会员”，需要用户先充值才能继续使用该功能。

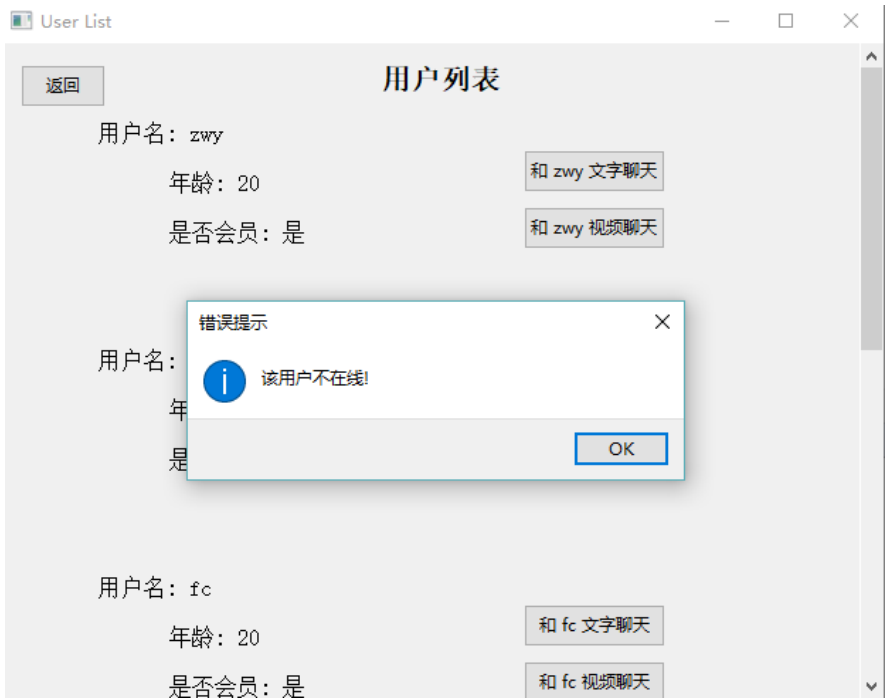
在系统测试过程中，主要测试用户不是会员时系统给出的提示信息是否正确，此时系统应给出“您不是会员”错误信息。



2.4 用户发送聊天请求

当用户在列表中向系统中其他用户发送文字或视频聊天请求时，系统会检查对方是否在线，对方是否是会员等条件。

系统测试时，会测试聊天对象不在线或者为非会员的情况。如果聊天对象用户未登录系统，则系统将提示“用户不在线”错误信息。



如果聊天对象用户不是会员，那么系统将提示“对方不是会员”错误信息。

