

编号：CXQ

版本：V1.0

软件设计说明书

文档作者：张磊

项目名称：畅想器

项目经理：孙笑凡

编写:张磊 2014 年 5 月 6 日

校对:孙笑凡 2014 年 5 月 12 日

目录

目录.....	- 1 -
1.1 编写目的.....	- 3 -
1.2 命名规范.....	- 3 -
1.3 术语定义.....	- 3 -
1.4 参考资料.....	- 4 -
1.5 版本更新信息.....	- 4 -
2.总体设计.....	- 4 -
2.1 硬件运行环境.....	- 4 -
2.2 软件运行环境.....	- 4 -
2.3 子系统模块.....	- 5 -
2.4 子系统功能模块关系.....	- 6 -
2.5 功能模块清单.....	- 6 -
3.数据库设计.....	- 8 -
3.1 数据库表之间的关系.....	- 8 -
3.2 数据库中表名列表.....	- 9 -
3.3 数据库表的详细清单.....	- 9 -
4.功能模块设计.....	- 12 -
4.1 用户注册.....	- 12 -
4.2 会员登陆.....	- 12 -
4.3 用户注销.....	- 13 -
4.4 修改个人信息.....	- 13 -
4.5 搜索好友.....	- 14 -
4.6 添加好友.....	- 14 -
4.7 同意好友请求.....	- 15 -
4.8 下载插件.....	- 15 -

4.9 邀请好友运行插件.....	- 15 -
4.10 从服务器获取好友信息.....	- 16 -
4.11 向服务器发送消息.....	- 16 -
4.12 服务器检查在线用户.....	- 17 -
4.13 服务器与数据库的连接.....	- 17 -
4.14 客户端与客户端的数据通信.....	- 18 -
4.15 插件 1：在线绘图.....	- 18 -
4.16 任务墙模块.....	- 18 -
4.17 文件共享模块.....	- 19 -
4.18 添加好友.....	- 20 -
4.19 删除好友.....	- 20 -
4.20 查询用户.....	- 21 -
4.21 离线好友请求.....	- 21 -
5.调用过程设计.....	- 22 -
5.1 登陆页面.....	- 22 -
5.2 网络通信.....	- 23 -
5.3 初始化用户数据.....	- 23 -
5.4 初始插件系统.....	- 25 -
5.5 加载完成，等待用户启动插件.....	- 26 -
6.角色授权设计.....	- 28 -
6.1 角色授权.....	- 28 -
7.系统错误处理.....	- 29 -
7.1 出错信息.....	- 29 -
7.2 故障预防与补救.....	- 29 -
7.3 系统维护设计.....	- 29 -
8.测试计划.....	- 30 -

1.引言

1.1 编写目的

主要为有在线交流需求的普通用户和有联网工作需求的工作者。

畅想器是多人联网的工作平台，有多人联网绘图、语音通讯、多人工作等应用，充分满足了普通用户的基本在线交流需求。另外，畅想器支持网络条件较差或局域网环境下的交流。

畅想器可以通过网络实现多人联网自动化办公，多人产品设计，网络会议，网络教学等应用。因此可以满足平面设计者，教师，公司管理人员等工作者的在线工作需求。

1.2 命名规范

- (1) 数据库表名、字段名用英文命名。
- (2) 私有变量用“_”开头的小写英文命名，其他成员变量用小写英文字母开头，属性用大写英文字母开头。
- (3) 所有的函数采用了驼峰命名规则。

1.3 术语定义

- (1) 总体结构：软件系统的总体逻辑结构，本系统采用面向对象的方法对系统进行设计。
- (2) 物理数据模型（PDM）：关系数据库的物理设计模型
- (3) 通信协议（TCP）：面向连接（连接导向）的、可靠的、基于 IP

的传输层协议

1.4 参考资料

[1]吕云翔 王昕鹏 邱玉龙. 软件工程——理论与实践

1.5 版本更新信息

版本号	创建者	创建日期	维护者	维护日期	维护纪要
V1.0	孙笑凡	2014-4-1			

2.总体设计

2.1 硬件运行环境

(1)CPU: Core i5

(2)内存大小: 4GDDR3

(3)磁盘空间容量: 500G7200 转

(4)鼠标, 键盘

2.2 软件运行环境

(1)操作系统: win7/win8

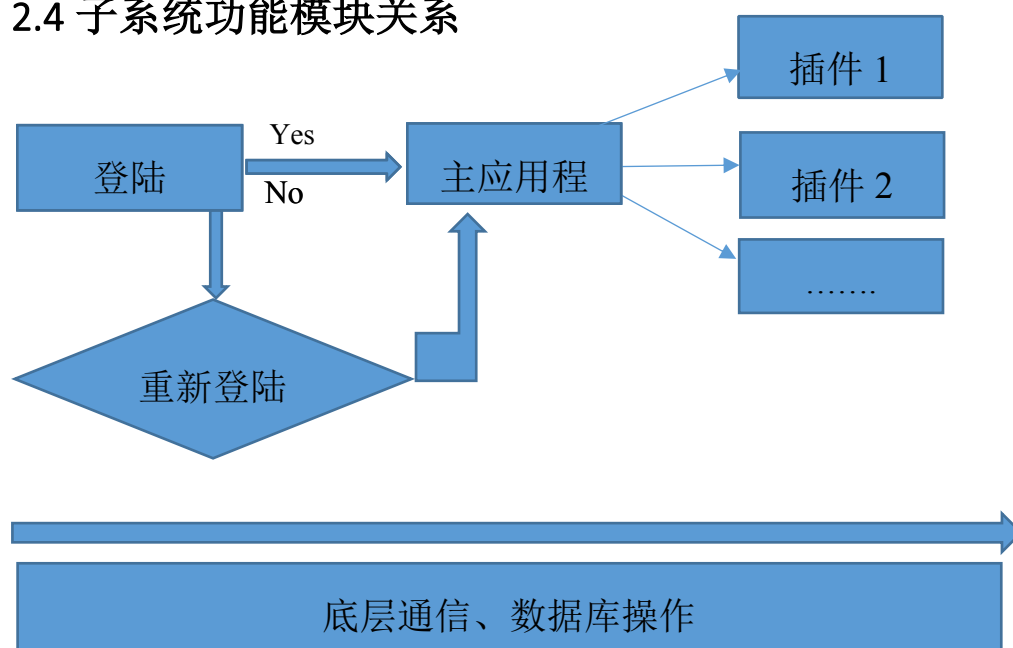
(2)数据库: MySQL/SQL Server

(3)开发工具: Visual Studio2010/2012/2013

2.3 子系统模块

子系统编号	子系统名称	子系统功能简述
SS1	登录模块	包括一个登录界面和一个用户注册界面，是主程序的进入窗口。通过与数据库的交互操作判定该用户注册名是否已经存在或者用户登录的用户名和密码是否正确
SS2	主应用程序	包括一个主应用程序的窗口，实现各种操作的集合以及响应处理。
SS3	底层通信	提供了一种通用的数据传输方式，能快速的建立客户端与服务器之间的通信。
SS4	插件模块	包含各种功能插件，包括多人绘图、文件分享、任务板，实现功能的简洁、快速实现
SS5	数据库操作模块	用于用户登录注册时的数据插入和查询判断，以及服务器用于向客户端返回的各类数据。

2.4 子系统功能模块关系



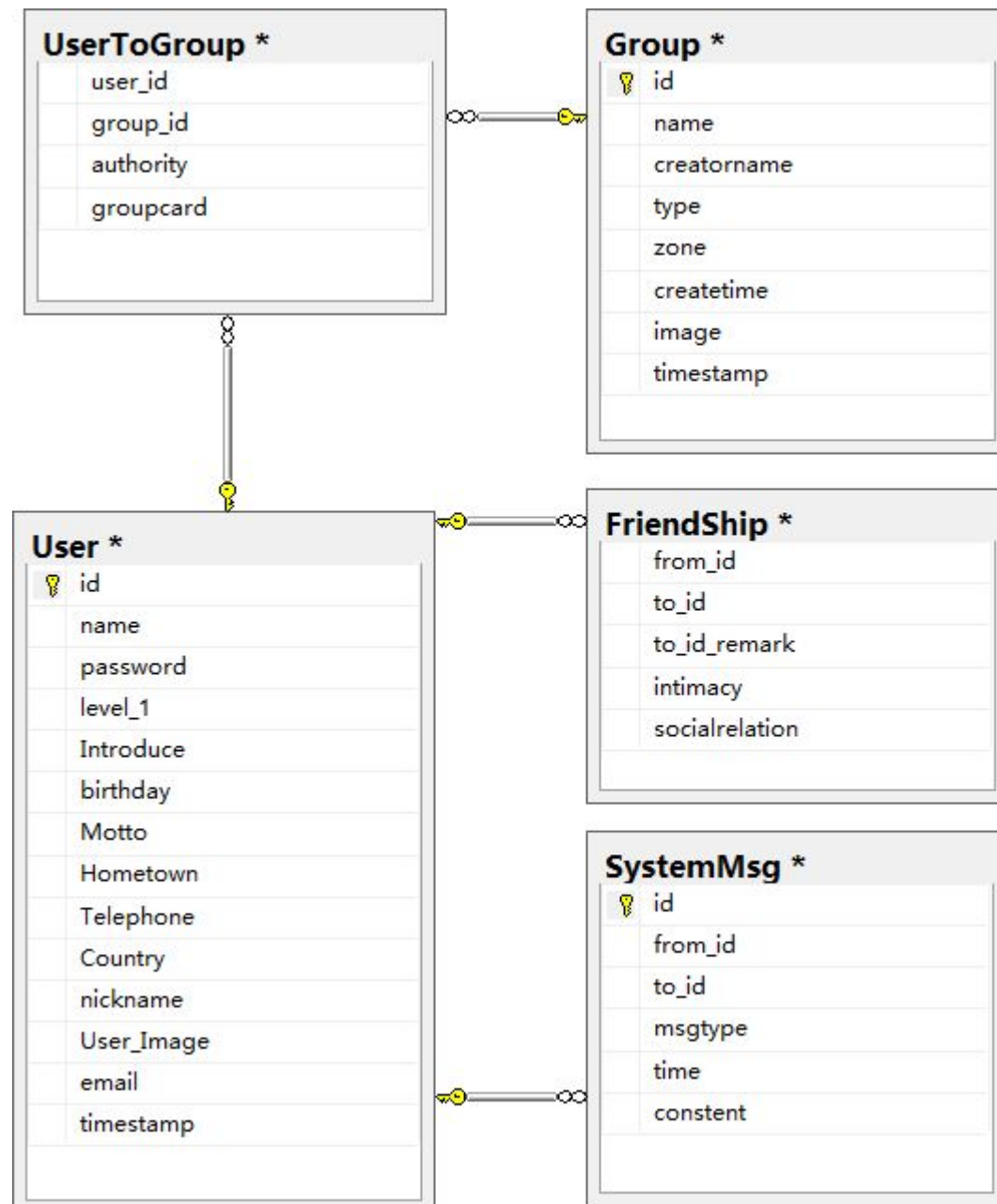
2.5 功能模块清单

模块编号	名称	模块功能描述
SS1-1	用户注册	登录的必要操作
SS1-2	会员登录	用户输入账号密码进行登录
SS1-3	用户注销	用户关掉软件之后
SS1-4	修改个人信息	会员在注册时有另一个页面填写个人信息
SS2-1	搜索好友	根据用户名或者昵称搜索好友
SS2-2	添加好友	用户添加好友至好友列表
SS2-3	同意好友请求	回应好友的加好友请求并返回给服务器
SS2-4	下载插件	从服务器端下载插件
SS2-5	邀请好友运行插件	邀请选择的好友一起运行插件

SS3-1	从服务器获取好友信息	向服务器请求获取好友的相关数据
SS3-2	向服务器发送消息	发送诸如好友请求之类的消息
SS3-3	服务器检查在线用户	在线用户与服务器不互发消息达较长一段时间后自动下线
SS3-4	服务器与数据库的连接	服务器运行期间保持与数据库的连接，不断进行数据的更新操作
SS3-5	客户端与客户端的数据通信	运行插件之后，根据相互间的 IP 位置进行数据通信
SS4-1	插件 1：在线绘图	多个用户共享一个绘图板进行绘图操作
SS4-2	插件 2：任务墙	任务发布板
SS4-3	插件 3：文件分享	用户可从另一个用户的共享文件夹里边取出文件
SS5-1	添加好友，群组	用户添加好友，数据库更新好友关系表
SS5-2	删除好友，群组	用户添加好友，数据库更新好友关系表
SS5-3	查询用户，群组	用户查询用户，数据库返回用户信息
SS5-4	离线好友请求	包括申请添加好友，添加群组等，存储在数据库的消息表中，待用户上线时返用户

3.数据库设计

3.1 数据库表之间的关系



3.2 数据库中表名列表

编号	表名	表功能说明
1	FriendShip	好友关系表
2	Group	群组表
3	SystemMsg	系统消息表
4	User	用户具体信息表
5	UserToGroup	用户所属小组表

3.3 数据库表的详细清单

表 1

FriendShip 表




































	字段	类型	整理	属性	Null	默认	额外	操作
<input type="checkbox"/>	from_id	int(6)			否			      
<input type="checkbox"/>	to_id	int(6)			否			      
<input type="checkbox"/>	to_id_remark	varchar(20)	utf8_bin		是	NULL		      
<input type="checkbox"/>	intimacy	int(6)			否	0		      
<input type="checkbox"/>	socialrelation	varchar(20)	utf8_bin		否	None		      

表 2

Group 表

	字段	类型	整理	属性	Null	默认	额外	操作							
<input type="checkbox"/>	<u>id</u>	int(6)			否		auto_increment								
<input type="checkbox"/>	name	varchar(20)	utf8_bin		否	群组									
<input type="checkbox"/>	creatorname	varchar(20)	utf8_bin		否	群主									
<input type="checkbox"/>	type	varchar(20)	utf8_bin		是	NULL									
<input type="checkbox"/>	zone	varchar(50)	utf8_bin		是	NULL									
<input type="checkbox"/>	createtime	bigint(20)			否										
<input type="checkbox"/>	image	mediumblob		BINARY	否										
<input type="checkbox"/>	timestamp	bigint(20)			否										

表 3

SystemMsg 表

	字段	类型	整理	属性	Null	默认	额外	操作							
<input type="checkbox"/>	<u>id</u>	int(6)			否		auto_increment								
<input type="checkbox"/>	from_id	int(6)			否										
<input type="checkbox"/>	to_id	int(6)			否										
<input type="checkbox"/>	msgtype	int(3)			否										
<input type="checkbox"/>	time	bigint(20)			否										
<input type="checkbox"/>	content	mediumblob		BINARY	是	NULL									

表 4

User 表






















	字段	类型	整理	属性	Null	默认	额外	
<input type="checkbox"/>	<u>id</u>	int(6)			否		auto_increment	 
<input type="checkbox"/>	name	varchar(30)	utf8_bin		否			 
<input type="checkbox"/>	password	varchar(20)	utf8_bin		否			 
<input type="checkbox"/>	Level	int(6)			否	0		 
<input type="checkbox"/>	Introduce	varchar(100)	utf8_bin		否			 
<input type="checkbox"/>	birthday	date			否			 
<input type="checkbox"/>	Motto	varchar(100)	utf8_bin		否			 
<input type="checkbox"/>	Hometown	varchar(30)	utf8_bin		否			 
<input type="checkbox"/>	Telephone	varchar(15)	utf8_bin		否			 
<input type="checkbox"/>	Country	varchar(30)	utf8_bin		否			 
<input type="checkbox"/>	nickname	varchar(20)	utf8_bin		否			 
<input type="checkbox"/>	User_Image	mediumblob		BINARY	是	NULL		 
<input type="checkbox"/>	email	varchar(30)	utf8_bin		否			 
<input type="checkbox"/>	timestamp	bigint(20)			否			 

表 5

UserToGroup 表

	字段	类型	整理	属性	Null	默认	额外	操作						
<input type="checkbox"/>	user_id	int(6)			否									
<input type="checkbox"/>	group_id	int(6)			否									
<input type="checkbox"/>	authority	int(6)			否	0								
<input type="checkbox"/>	groupcard	varchar(20)	utf8_bin		否									

4.功能模块设计

4.1 用户注册

(1) 调用背景：用户第一次使用该软件时，首先需要打开登陆界面，点击注册按钮，进入注册界面，进行会员注册。

(2) 调用描述：

- ① 用户需要首先输入想要注册的用户名，系统会提醒该用户名是否已经被注册。如果没有被注册，则会显示出一个勾，表示该用户名可以注册，否则重新输入用户名进行注册。
- ② 用户需要输入昵称、密码和重复密码
- ③ 用户点击注册按钮，进行会员注册

4.2 会员登陆

(1) 参数：string username=? string password=?

(2) 调用背景：当用户打开登陆界面时，需要输入已经注册过的用户名和用户密码进行登陆。

(3) 调用描述：

- ① 在用户名输入框处输入用户名
- ② 在用户密码输入框处输入用户密码
- ③ 单击登陆按钮，如果用户名和密码已经注册并且输入正确，则登陆成功；否则提醒登录名或者密码输入错误，进行重新登陆。

4.3 用户注销

(1) 参数：int id=?

(2) 调用背景：当用户不需要继续使用软件的时候，可以进行注销，安全退出。

(3) 调用描述：

① 用户进行关闭操作；

② 系统会进行用户的 id 检查，确认用户是否在线。如果不在线，则系统进行用户注销，关闭用户与服务器的连接，进行安全退出。

4.4 修改个人信息

(1) 调用背景：当用户进行成功注册的操作后，表示用户已经成为会员，这时候，会弹出另一个界面，需要会员填入关于自己的更加详细的信息并提交，系统会将其存入数据库

(2) 调用描述：

① 用户需要选择一张图片进行上传操作

② 用户还需要填入所在地区、联系方式、自我陈述等更加详细的信息

③ 用户点击提交按钮，信息修改完成。

4.5 搜索好友

- (1) 参数: string name=?
- (2) 调用背景: 当用户使用该软件的时候, 如果出现好友过多, 当需要查找某个好友时比较困难的情况时, 可以进行按好友名进行查找的功能。
- (3) 调用描述:
 - ① 用户在输入框中输入想要搜索的好友姓名
 - ② 用户点击搜索按钮, 并等待系统的搜索结果
 - ③ 如果存在搜索的用户名, 则会反馈给用户想要查找的好友的信息; 否则会提醒用户不存在所要搜索的这个好友的信息

4.6 添加好友

- (1) 参数: Me.user_id = ?, SelectedItems.user_id = ?, isSendAddFriend = ?
用户 ID, 目标用户 ID, 请求是否被服务器响应。
- (2) 调用背景: 按用户名在用户数据库中查找后, 得到返回的搜索结果列表, 选中列表中用户后, 点击加为好友按钮。
- (3) 页面组成: 分为三部分, 分别是搜索框以及搜索按钮, 搜索结果列表, 加为好友按钮。
- (4) 调用描述: 点击加为好友后, 调用 ServerAPI.AddFriend 功能, 发送添加好友请求, 通过返回的 isSendAddFriend 的值, 告知用户“已经发送添加好友请求”, 和“未发送任何请求”。

4.7 同意好友请求

- (1) 参数: Status=?, user_id=?, messageType=?, 好友请求相应情况 (同意, 拒绝), 目标用户 ID, 影响界面组成的参数。
- (2) 调用背景: 查看当前信息列表后出现。
- (3) 页面组成: 3 个按钮 ok, cancel, check, 一个文本框, 通过 messageType 值的改变, 对前四项进行改动。
- (4) 调用描述: 对于没有处理过的信息, 显示同意与拒绝按钮。如果信息已经被处理, 显示“已拒绝”或“已同意”, 同时显示 button_check, 点击后出现已阅。

4.8 下载插件

- (1) 调用背景: 在主程序窗口点击下载插件
- (2) 页面组成: 下载进度
- (3) 调用描述: 自动下载

4.9 邀请好友运行插件

- (1) 参数: p.ID= ?, Me.user_id = ?, f_list = <??> , 传至服务器端的消息 ID, 邀请人 ID, 被邀请人列表。
- (2) 调用背景: 在主程序窗口, 选中好友后, 点击启动插件。
- (3) 调用描述: 服务器端处理接收到的参数。

4.10 从服务器获取好友信息

- (1) 参数: item.Value=? , Friend.TimeStamp=? , 服务器端好友列表更新时间, 本地端好友列表更新时间。
- (2) 调用背景: 开启主程序时。
- (3) 调用描述: 如果服务器端好友列表新于本地好友列表则更新好友列表。

4.11 向服务器发送消息

- (1) 参数: UserMessage() 它的参数针对向服务器发送的不同消息不同, 包括:

注 册 : UserMessage(RegisterData(User(1, null, username, nickname),password))

登录: UserMessage(LoginData(username, password))

确认在线操作: UserMessage(id)

邀请好友: UserMessage(List<int>)等。

- (2) 调用背景: 在使用注册登录等不同功能时, 调用含不同参数格式的方法向服务器发送消息。
- (3) 调用描述:

① 注 册 : UserMessage(RegisterData(User(1, null, username, nickname),password))

② 登录: UserMessage(LoginData(username, password))

- ③ 确认在线操作: `UserMessage(id)`
- ④ 邀请好友: `UserMessage(List<int>)`

4.12 服务器检查在线用户

- (1) 参数: `id`, 服务器检测的客户端的 `id`。
- (2) 调用背景: 客户端登陆后, 每秒会向服务器端发送一条确认自己在线的消息, 如果 5 秒内未发送, 则认为是客户端掉线。
- (3) 调用描述:
 - ① `SendMessage` 在登陆后, 开始不停的启动自己的一个线程进行消息发送。
 - ② 在发送消息时, 去取服务器端的用户邮箱。
 - ③ 如果有别人发来的 `Message`, 这时就会被取到本地, 并发至 `GetReturn` 类进行处理。

4.13 服务器与数据库的连接

- (1) 参数: 数据库的位置 (`ip`), 数据库用户名, 密码。
- (2) 调用背景: 需要对于数据库数据进行操作。
- (3) 调用描述:
 - ① 所有对数据库的操作用一个类来实现: `MySQL_Manager`
 - ② 连接: 构造函数设置参数, `Open()` 打开连接。
 - ③ 关闭: `Close()` 关闭连接。
 - ④ 执行 SQL 语句: 封装成一个函数 `RunSQL(String)`。

4.14 客户端与客户端的数据通信

- (1) 参数: id, 进行通信的两个或多个客户端的 id
- (2) 调用背景: 两个或多个客户端进行传递消息时, 调用此参数。
- (3) 调用描述:
 - ① 用户在向一个好友发起聊天时, `public SortedDictionary<int, Friend> FriendList`, 将找到好友的 id。
 - ② 在收到消息时, `public List<MessageDone> MessageList` 这个顺序表将承载给你发消息的不同 id。

4.15 插件 1: 在线绘图

- (1) 调用背景: 用户在使用在线绘图功能时, 此插件被调用。
- (2) 调用描述:
 - ① 首先调用 `DreamingPlugin`, 这是在线绘图功能的启动插件。
 - ② 调用 `DreamingApp`, 这是在线绘图功能的实质应用程序。

4.16 任务墙模块

- (1) 参数: `Plugin` 类
- (2) 调用背景: 用户主动启动该功能模块时, 或被邀请调用该模块时
- (3) 调用描述:
 - ① 软件初始化时, 软件在主界面 UI 上增加启动按钮
 - ② 启动按钮被点击或用户接受邀请按钮被点击
 - ③ 插件回调运行, 主程序为 `Plugin` 类成员赋值

④ Pluign 下的 Run 方法启动，执行 Plugin 主函数

(4) 功能：

启动一个任务版界面，显示大家最近需要进行的任务，每个人可以选择其中的一些进行，并且可以显示每个任务都被谁正在执行，进度如何，任务条反面有任务的详细描述。

(5) 界面：

整个界面是一个可拖动元素的 Canvas 组成，里面分布有众多可翻转元素，作为每一个任务的描述，正面是填写任务名称，反面是任务的详细描述框。

4.17 文件共享模块

(1) 参数： Plugin 类

(2) 调用背景： 用户主动启动该功能模块时，或被邀请调用该模块时。

(3) 调用描述：

① 软件初始化时，软件在主界面 UI 上增加启动按钮

② 启动按钮被点击或用户接受邀请按钮被点击

③ 插件回调运行，主程序为 Plugin 类成员赋值

④ Pluign 下的 Run 方法启动，执行 Plugin 主函数

(4) 功能：

用户可以浏览其他用户的共享文件夹下的内容，可以下载其中的离线共享文件，可以设置自己的共享文件夹，可以设置多人同步文件夹。

(5) 主界面： 类似资源管理器一样的界面，可以查看网络上各好友的共享资源。

(6) 设置界面： 两个路径设置文本框，一个参数设置列表。

4.18 添加好友

(1) 参数： `public bool makeFriend(int[] kk)` ， 传入的数组中， 0,1 位放入两个 id 号。

(2) 返回值： 是否执行成功。

(3) 功能： 将两个人设置为好友关系

(4) 调用背景： 用户同意添加某人为好友时， 服务器端执行

(5) 调用描述：

- ① 用户向服务器发送同意添加好友请求
- ② 服务器将两用户设置为好友关系

4.19 删除好友

(1) 参数： `public bool deleteAFriend(int from_id, int to_id)`

(2) 返回值： 是否执行成功

(3) 功能： 删除两人的好友关系

(4) 调用背景： 用户请求删除好友时， 服务器端执行

(5) 调用描述：

- ① 用户向服务器发送同意删除好友请求
- ② 服务器将两用户删除好友关系

4.20 查询用户

(1) 参数: `public List<DrawBitmap.User> getUserByName(String name)`

`name` 查询时的用户名。

(2) 返回值: 查询到的用户列表

(3) 功能: 返回查询到的用户列表

(4) 调用背景: 用户请求查询好友时, 服务器端执行

(5) 调用描述:

- ① 用户向服务器发送查询好友请求
- ② 服务器找到符合要求的好友列表并返回
- ③ 客户端呈现所有好友

4.21 离线好友请求

`public bool SaveUserMessage(int from_id, int to_id, string mdata)`

(1) 参数: 两个用户 id, 一个消息的数据

(2) 返回值: 是否执行成功

(3) 功能: 将用户给其他离线者的消息存入数据库

(4) 调用背景: 某在线用户发消息给离线用户时使用

(5) 调用描述:

- ① 客户端请求发送消息给某用户
- ② 服务器检查到该用户未在线, 将此消息存入数据库中

5.调用过程设计

5.1 登陆页面

在用户登陆时会触发登陆事件，将页面上的数据导入系统中

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    API.Login(Username.Text, Password.Password); //核心登陆方法
    App.Current.MainWindow.Close();
}
```

(1) 参数： 用户名 密码

(2) 返回值： 是否登陆成功

```
public static bool Login(string username,string password)
{
    LoginReturn m = ServerAPI.Login(username, password); //通信，发送登陆信息
    App.data = new AppData();
    var data = App.data;
    if (m != null)
    {
        UserWindow window = new UserWindow();
        App.mainWindow = window;
        App.data.InitLoginData(m); //初始化应用程序数据
        App.mainWindow = window;
        window.Show(); //显示窗口
        return true;
    }
    else return false;
}
```

5.2 网络通信

向服务端发送登陆请求，验证用户名和密码

如果成功，则返回用户数据，用户用此数据进行软件初始化

```
/// 登录
/// 发送格式: UserMessage(2, LoginData(username, password))
/// 接收格式: LoginReturn,要求 LoginReturn 各成员变量不为 null
public static LoginReturn Login(string username, string password)
{
    LoginData ld = new LoginData(username, password);
    lock (_l)
    {
        if (client.Connect()) //连接
        {
            LoginReturn result = client.Sending(new UserMessage(2, ld)) as
LoginReturn; //获取返回数据
            client.Close();
            return result;
        }
    }
    return null;
}
```

5.3 初始化用户数据

返回回来的用户数据保护自己的 id 和时间戳，所有好友的 id 和时间戳，群组 id 和时间戳。

软件首先从本地加载数据，将本地时间戳和网上的时间戳进行对比，

看是否一致最新，若网上版本新，则联网获取最新数据。

软件每次都会检查未处理的数据，并将其获取下来。

/// 初始化数据，先加载自己的数据，然后是好友列表，然后是群组表，插件表

```
public void InitLoginData(LoginReturn data)
```

```
{
```

```
    InitMe(data.myTimeStamp.Key, data.myTimeStamp.Value); //初始化自己的信息
```

```
    //从磁盘加载好友数据
```

```
    //联网获取时间戳
```

```
    // 校验版本
```

```
    // 若不对要联网更新数据
```

```
    InitFriend(data.FriendsTimeStamp); //初始化好友信息
```

```
    //更新在线好友，更新 ip
```

```
    foreach (var item in data.OnlineFriends)
```

```
    {
```

```
        Friend f = null;
```

```
        if (FriendList.TryGetValue(item.Key, out f) && f != null)
```

```
        {
```

```
            f.isOnline = true;
```

```
            //ToDo:
```

```
            f.ip = new System.Net.IPAddress(item.Value);
```

```
        }
```

```
    }
```

```
    //解析离线消息
```

```
    foreach (var item in data.message)
```

```
{  
    GetReturn.ParseMessage(item);  
}  
  
InitGroup();  
InitSending();  
}
```

5.4 初始插件系统

插件系统首先会检索指定路径下的所有文件夹，在这些文件夹下找到插件 dll 并尝试加载，之后会遍历所有已有的插件，检查他们的版本和其依赖的插件 id，如果依赖项不存在，则会联网获取该插件。本操作是递归实现的，如果一个插件被下载后，仍有依赖项未被安装，则会重复该过程。

至此，插件系统初始化完毕，在软件完全退出并再次启动前，不会再重复加载插件系统。

不过每一次主界面启动时，都会调用各个插件的初始化代码，在主界面上添加自己的界面 UI。

```
public void InitPluginSystem()  
{  
    var dic = plugin_dic;  
    if (!isPluginReady)  
    {  
        InitPlugin(); //从磁盘加载插件
```

```

        if (plugin_list == null) return;
        foreach (var item in plugin_list)
        {
            dic.Add(item.ID, item);
        }
    }
    foreach (var item in plugin_dic.Values)
    {
        item.Init();
        item.runhandle += item_runhandle; //添加插件运行回调
        if (item.Dependencies != null)    //检查依赖项
            foreach (var id in item.Dependencies)
            {
                if (!dic.ContainsKey(id))
                {
                    //TODO:
                    //download...
                }
            }
    }
    isPluginReady = true;
}

```

5.5 加载完成，等待用户启动插件

至此，软件登陆完成，用户每次点击界面 UI，都会触发对应的插件对其响应

用户可以使用我们的多人绘图、共享文件、任务版等功能。

`public void RunPlugin()`

(1) 参数： 无

(2) 功能描述：

启动插件 （绘图，任务板，文件分享）

6.角色授权设计

6.1 角色授权

见表 6.1

表 6.1 角色授权

模块	管理员	用户	非注册会员
登录注册模块	●	●	●
主应用程序	●	●	
底层通信	●		
插件模块	●	●	
数据库操作模块	●		

●表示全部权限

7.系统错误处理

7.1 出错信息

- (1)对注册用户输入的各项内容进行有效性、安全性检查，包括数据库检查内容的重复性、正确性，输入框的正则表达式的格式正确性等，减少错误发生的几率
- (2)对程序中运行的异常进行捕获，当捕获到异常信息之后，或者将出错提示消息提供给用户，或者程序自动在后台进行更正，或者程序终止，强制用户退出进行重新登陆
- (3)当用户访问自身以外的权限信息时，将按照统一的格式将出错信息提供给用户

7.2 故障预防与补救

- (1)对数据库和数据库中的文件信息进行加密处理以及备份处理，防止用户的数据丢失或者被窃取
- (2)定期对后台代码进行测试和优化，及时发现故障，减少故障出现的机率

7.3 系统维护设计

- (1)在编码过程中注意良好的编码风格，合适的命名、适量的注释
- (2)编码实现时应采用模块化和分层的思想，提高模块内部的内聚，减

少模块间的耦合性。使系统逻辑结构清晰，从而增强可读性和可维护性

(3)面向数据与面向对象相结合，模块划分符合面向对象思想

8.测试计划

在先期工作中，我们的小组已经进行了 2 项具体的模块功能测试。因此，在后续的测试计划中，我们需要按照计划和现实情况的变化进行双线测试。而单元测试、集成测试、系统测试 3 个方面会是我们小组进行重点设计单元。