数字工程软件架构

——地图数据质检工具（局域网）

学 院：遥感信息工程学院

班 级：2006 班（20F10）

姓 名：马文卓

学 号：2020302131249

目录

1. 项目要求

设计开发用于空间数据质量检查的应用软件工具,该工具运行于局域网环境下的桌面环境；说明及评估所采用的软件架构风格。

1.1 项目目标

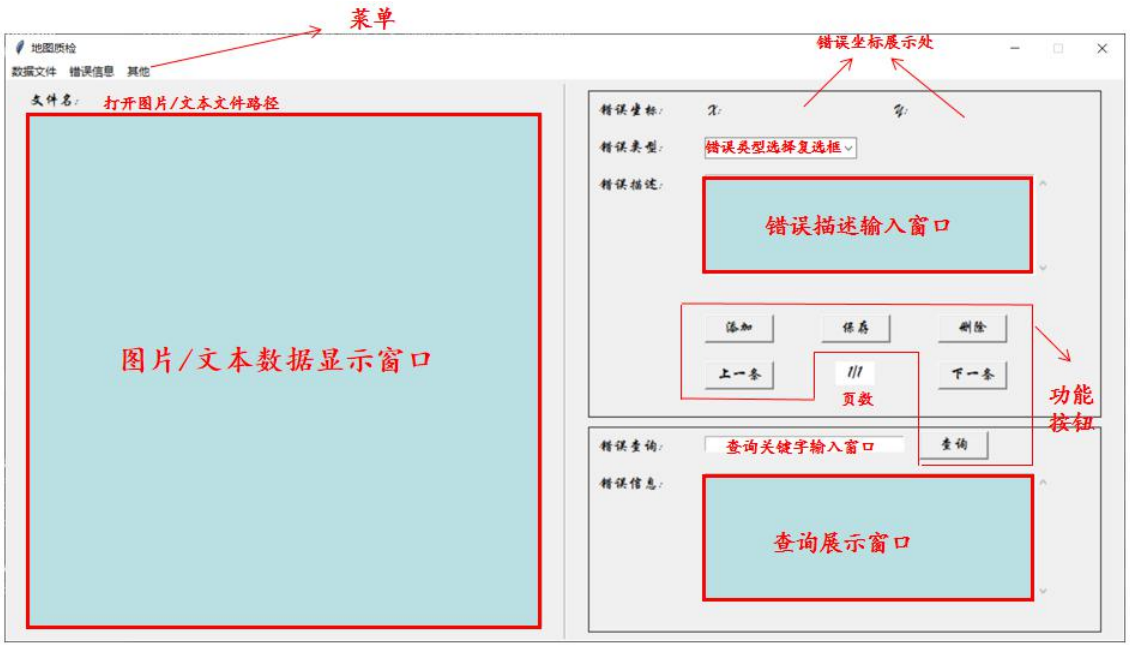
A、地图数据加载：从应用中打开指定的图形或文档类型地图数据文件，并显示在应用程序窗口中。
B、地图数据显示：针对不同类型数据采用不同方式：图形形式的可视化符号显示（主要有栅格及矢量格式类型，原型中至少实现其中一种类型的可视化进行示意）；文本形式直接显示数值内容（原型中可用普通文本文件示意）。
C、错误记录:通过在图形中或文本中指定位置，标记各项错误的位置;指出错误位置后，输入错误类型、错误描述等信息并自动给错误编号；在文件中保存所记录的错误信息。
D、错误记录查询：地图数据加载后，在质检工具中打开已保存的错误信息文件，可通过输入关键字搜索并显示出错情况。
E、用户管理：支持多用户，不同客户端多用户同时使用、同一用户可在不同终端使用；进行用户登录等用户验证
F、用户操作记录：记录每个用户的地图数据提交情况、数据错误登记情况

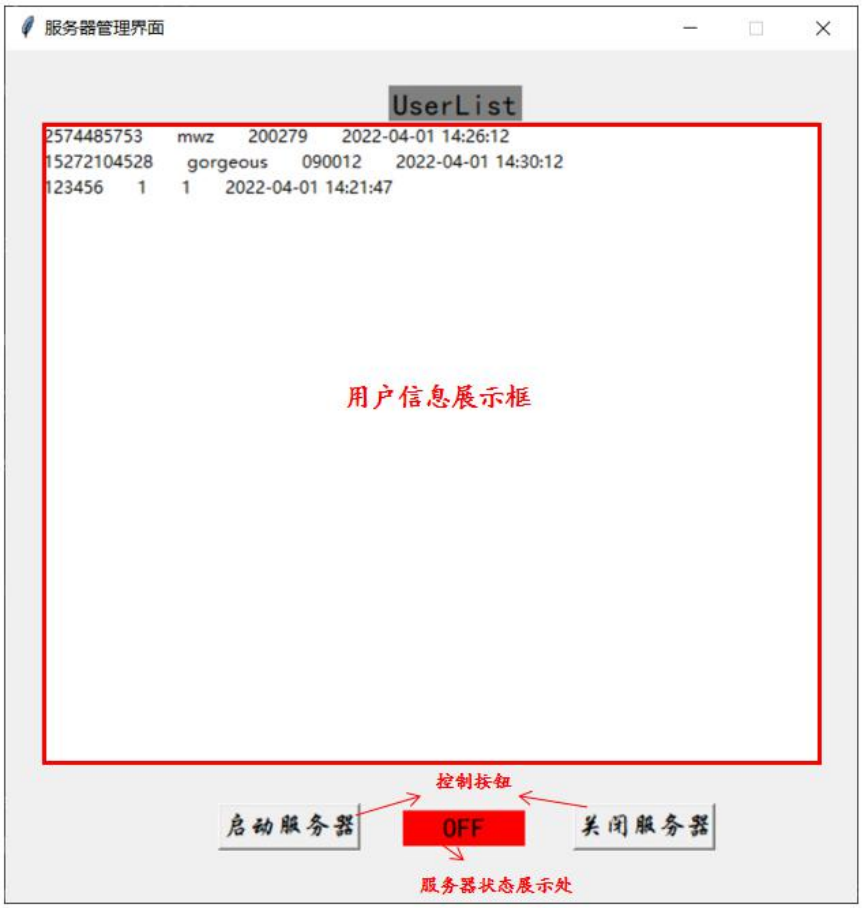
1.2 成果要求

A、软件架构说明。说明所采用的软件架构模式及主要优缺点。
B、软件源码及可执行代码包。提示：独立部署运行的数据构件可选择开源的个人版或开发版 DBMS（如 MySQL 等）

2. 界面说明

2.1 界面 GUI 图




服务器管理界面
UserList
2574485753 mwz 200279 2022-04-01 14:26:12
15272104528 gorgeous 090012 2022-04-01 14:30:12
123456 1 1 2022-04-01 14:21:47
用户信息展示框
控制按钮
启动服务器
OFF
关闭服务器
服务器状态展示处


注册
账号：
ID输入框
密码：
密码输入框
用户名：
用户名输入框
确定
确认按钮

2.2 界面简介

本项目的界面使用的是 Python 的 tkinter 库，主要分为四个界面：客户端主程序界面、客户端登录界面、注册界面、服务端管理界面（如上图依次）。

(1)客户端主程序界面主要分为四大板块：分别为菜单栏、数据显示窗、错误信息操作窗、错误信息查询窗。

菜单栏（mainMenu）包括数据文件、错误信息、其他。错误文件子菜单为打开图像文件、打开文本文件；错误信息子菜单为新建错误文件、打开错误文件、删除错误文件；其他子菜单为帮助、退出。

数据显示窗主要为一个用于显示图片或者文本数据的窗口（Frame）和上面显示文件路径的标签（Label）。

错误信息操作窗（Frame）包括坐标显示标签（Label）、错误类型复选框（combobox）、错误描述输入框（SroclledText）、操作（添加、保存、删除、上一条、下一条）按钮（Button）和页面显示标签（Label）

错误信息查询窗(Frame)包括查询关键字输入框(Text)和查询按钮(Button)以及查询信息展示框（SrolledText）。

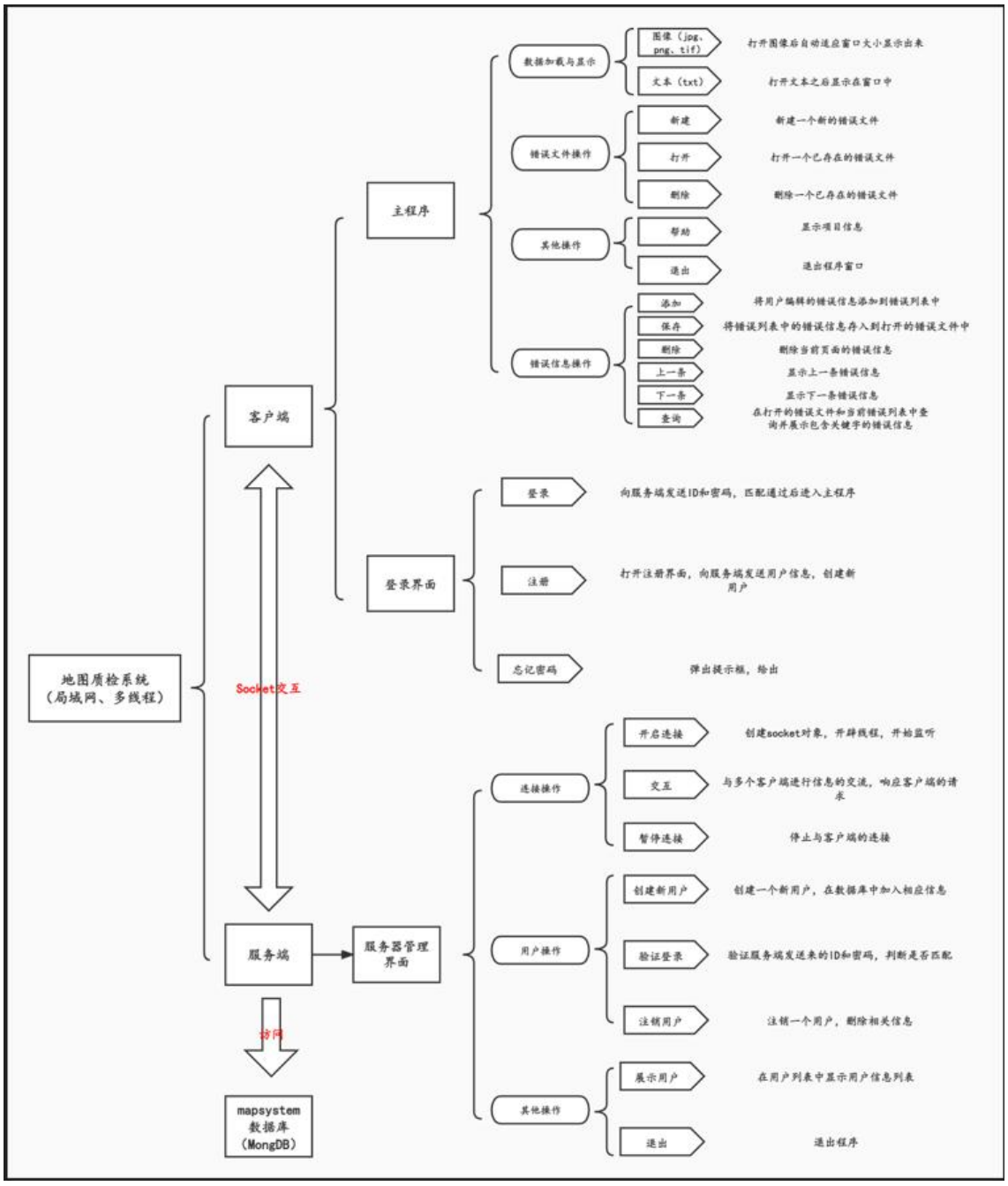
(2) 客户端登录界面由三个 Label、两个输入框（ID 输入框、密码输入框）、三个按钮（注册按钮、登录按钮、忘记密码按钮）组成。

(3) 注册界面主要包括三个输入框（ID 输入框、密码输入框、用户名输入框），一个确认按钮

(4) 服务端管理界面由一个用户展示列表 listbox，和两个按钮（开启、关闭按钮），和两个标签（状态标签、标题标签主城。）

3. 功能介绍

3.1 功能结构图



（高清图见附件）

3.2 账户新建

（1）客户端：在登录界面点击【注册】按钮进入注册界面，输入 ID、密码、用

户名之后点击【确认】，通过套接字客户端将注册信息传递给服务端，如果注册成功给出提示；如果账户已经存在，则给出“账户已被占用”的提示。
（2）服务端:通过套接字接受客户端发送来的注册信息,在MongDB的mapsystem数据库的users集合中寻找ID是否存在,如果不存在,则将注册信息打包为{ID, PassWord, UserName, LastTime}的形式插入其中，并且在user文件夹下面创建用该用户ID命名的文件夹，里面包含了image、txt、erro三个文件夹，分别用于存放该用户上传的图像、文本、错误文件，最后向客户端发送成功标志。

3.3 用户登录

（1）客户端：在登录界面输入ID和密码之后点击【登录】，客户端将登录信息发送给服务端，如果密码正确则启动主程序，否则给出提示
（2）服务端：通过套接字接受登录信息，在数据库中查找，如果密码匹配成功，给客户端返回成功信息，否则返回“无此ID”或者“密码错误”等验证错误信息。

3.4 账户管理

在服务端的用户列表展示界面,会将数据库users集合中的所有注册用户信息显示在界面中。左键双击某个用户，会弹出提示，是否要删除此用户。得到确认之后，系统会删除该用户在数据空中的信息，以及该用户的数据文件夹。

3.5 服务器与客户端连接

（1）客户端：每打开一个客户端，都会创建一个socket对象，去尝试和服务端进行连接，连接完成之后，就可进行交互。
（2）服务端：打开用户管理界面，点击【开启服务器】服务器会创建一个socket对象，开始监听客户端。如果有客户端尝试连接，则开辟一个线程与其进行连接，主线程继续监听，以实现多线程的功能（即可多个客户端同时操作）。点击【关闭服务器】关闭连接，结束服务。

3.6 数据加载与显示

（1）打开并显示图像数据：点击【数据文件】下的【打开图像数据】，即可在弹出的文件对话框中选择想要打开的图像数据文件(tif、png、jpg等格式均可)，打开图像之后会自动显示在数据显示窗并自动适应窗口大小,并且打开文件的路径和文件名会显示在窗口上方。
（2）打开并显示文本数据：点击【数据文件】下的【打开文本数据】，即可在弹出的文件对话框中选择想要打开的文本数据文件（txt格式），打开文本之后会自动显示在数据显示窗并自动适应窗口大小,并且打开文件的路径和文件名会显示在窗口上方。

3.7 错误信息操作

（1）新建错误文件：点击【错误信息】下的【新建错误文件】，即可在弹出的交互式对话框中输入想要新建的文件名（默认为 Erro.txt）。点击确定之后，程序会再此目录下新建一个错误文本文件，并且将其打开以做存储错误信息之用。注意：如果原本就已经有错误文件处于打开状态，则会被关闭；如果原来错误列表中任然有错误信息，则系统给出提示，提醒如果继续新建错误文件则错误列表中的错误信息则会丢失。
（2）打开错误文件：点击【错误信息】下的【打开错误文件】，即可在弹出的文件对话框中选择想要打开的错误文件。选择完毕之后，程序打开选定的错误文件，可以将后面添加的错误信息追加保存到打开错误文件的末尾，也可以在查询中查询已经打开的错误文件中的错误信息。注意：如果原本就已经有错误文件处于打开状态，则会被关闭；如果原来错误列表中任然有错误信息，则系统给出提示，提醒如果继续新建错误文件则错误列表中的错误信息则会丢失。
（3）删除错误文件：点击【错误信息】下的【删除错误错误文件】，即可在打开的文件对话框中选择想要删除的错误文件。程序会给出提示，询问是否删除选择的错误文件，如果用户确定，则删除此错误文件。
（4）添加错误信息：双击【数据显示窗口】中认为出错的位置，在错误【错误坐标展示处】会显示用户所点击的位置坐标，然后用户可以再【错误类型选择复选框】中选择错误类型（名称错误、类型错误、属性错误、其他错误），用户还可以在【错误描述输入框】中输入对错误的描述信息，最后点击【添加】按钮即可把此条错误信息存入错误列表中。
（5）删除错误信息：点击【删除】可以从错误列表中删除当前错误信息展示窗里面的这条错误信息
（6）保存错误信息：点击【保存】可以将当前错误列表中的所有信息保存到已经打开的错误文件中。如果没有已经打开的错误文件，系统就会弹出一个文件对话框，可以选择已有的错误文件保存到其中。注意：保存文件之后当前作物列表就会清空。
（7）查看上一条信息：点击【上一条】可以更新错误信息，展示当前页面的上一条错误信息在错误信息操作窗中。如果是第一条错误信息，则弹出提示窗口，提示已经是第一条错误信息。
（8）查看下一条信息：点击【下一条】可以更新错误信息，展示当前页面的下一条错误信息在错误信息操作窗中。如果是最后一条错误信息，则弹出提示窗口，提示已经是最后一条错误信息。
（9）错误查询：在【查询关键字输入窗口】输入查询关键字，点击【查询】按钮即可在当前打开的错误文件（如果已经打开错误文件）和当前错误列表中查询出所有带有这个关键字的错误信息，并且展示在【查询展示窗口】。

3.8 文件传输

（1）图像文件传输：客户端的用户打开一个图像文件在界面上显示的同时，通过套接字客户端会将图像数据发送给服务端，服务端接受图像之后将其保存在相应用户的 image 文件夹下面。

（2）文本文件传输：客户端的用户打开一个文本文件在界面上显示的同时，通过套接字客户端会将文本数据发送给服务端,服务端接受文本之后将其保存在相应用户的 txt 文件夹下面。
（3）错误文件传输：客户端的用户保存一个错误文件时，通过套接字客户端会将错误数据发送给服务端，服务端接受错误数据之后将其保存在相应用户的 erro 文件夹下面。

3.9 其他功能

（1）展示项目信息：点击【其他】下面的【帮助】，程序会弹出一个窗口，展示作者信息、使用语言、版本信息等
（2）退出程序：点击【其他】下面的【退出】，如果文件还未关闭，则会先关闭文件，再退出程序。
（3）忘记密码：点击登录界面的【忘记密码】，会弹出提示框。

4. 架构风格

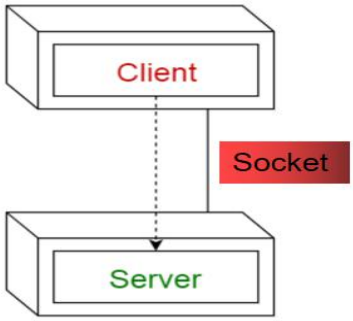
4.1 架构风格概述

本项目主要采用了**事件驱动**风格、**面向对象**风格、**客户机/服务器**风格和 MVC 风格相结合的**异构结构**风格。事件驱动风格主要体现在构件和执行相应事件当中，面向对象风格主要体现在将数据和操作封装在一个一个抽象数据类型或对象当中。这两种风格各有优缺点，在下面将逐一分析。将它们结合到一起，有利于功能的实现和代码的简洁性、重用性、灵活性等。

4.2 风格分析

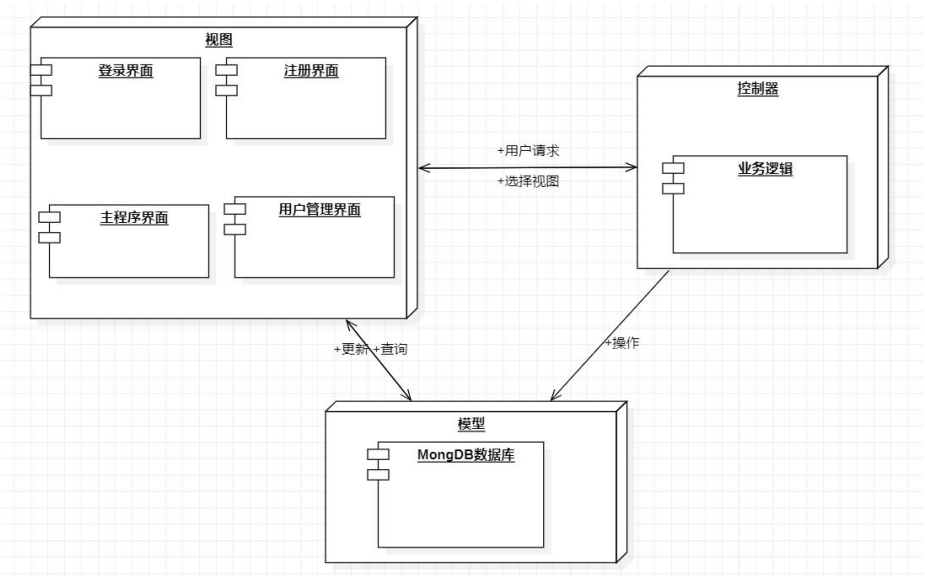
（1）事件驱动风格：构件不直接调用一个过程，而是触发或广播一个或多个事件系统中其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用这个事件中注册的所有过程,即一个事件的触发可能会导致另外一个模块中过程的调用。
优点：
①事件声明者不需要知道那些构件会影响事件，构件之间关联比较弱
②提高软件的复用能力，事件发布及处理都能复用
③系统便于升级，灵活添加事件及处理
缺点：
①构件放弃了对计算机的控制权，完全由系统来决定
②存在数据交换的问题
③数据共享能力低
④系统中各个对象的逻辑关系变得更加复杂
⑤该风格中的正确性验证存在问题
（2）面向对象风格：从现实世界中客观存在的事件出发，强调直接以问题域中的事物为中心来思考问题,根据这些事物的本质特征,将其抽象为系统中的对象,并作为系统的基本构成单位。
优点：
①对象隐藏了其实现细节，所以可以在不影响其他对象地情况下改变对象的实现，不仅使得对象的使用变得简单、方便，而且具有很高的安全性和可靠性。
②设计者可将一些数据存取操作问题分解成一些交互的代理程序集合
缺点：
①当一个对象和其他对象通过过程调用等方式进行交互时,必须知道其他的对象的标识。
②无论何时改变对象的标识，必须修改所有显式调用它的对象，并消除由此带来的一些副作用
（3）**客户机/服务器风格:**该模式由两部分构成：单个服务器端和多个客户端。服务器组件对多个客户端组件提供服务。客户端向服务器端请求服务，服务端提供对应服务给这些客户端。本项目是典型的客户机/服务器风格，多个客户端交由用户使用，用户上传的文件数据以及用户的注册登录信息全部通过 Socket 传

递到服务端，服务端将接收到的信息进行处理和存储，返回给客户端相应的响应。



优点：
①客户端只需要安装客户端，操作简单，不需要关心具体如何实现
②维护简单，只需要维护服务器和数据库
③客户机和服务器运行在不同的计算机上有利于分布式的数据处理和组织，同时也便于实现异构环境和多种不同技术的融合
缺点：
①信息内容和形式单一，所有客户端UI一致
②采用单一服务器，以局域网为中心，难以扩展到Internet

（4）MVC风格：模型——包含核心功能和数据，视图——显示信息给用户（多个视图可被定义），控制器——处理用户输入。它通过分割用户信息的内部陈述和呈现、接受方式来实现，解耦组件并允许高效的代码复用。在本项目中，很显然采用了这种风格。视图（一共三个界面：登录界面、主程序界面、用户管理界面），模型（全部封装在相关函数当中，以实现核心功能），控制器（也封装在相应的方法当中，以处理用户的输入）。



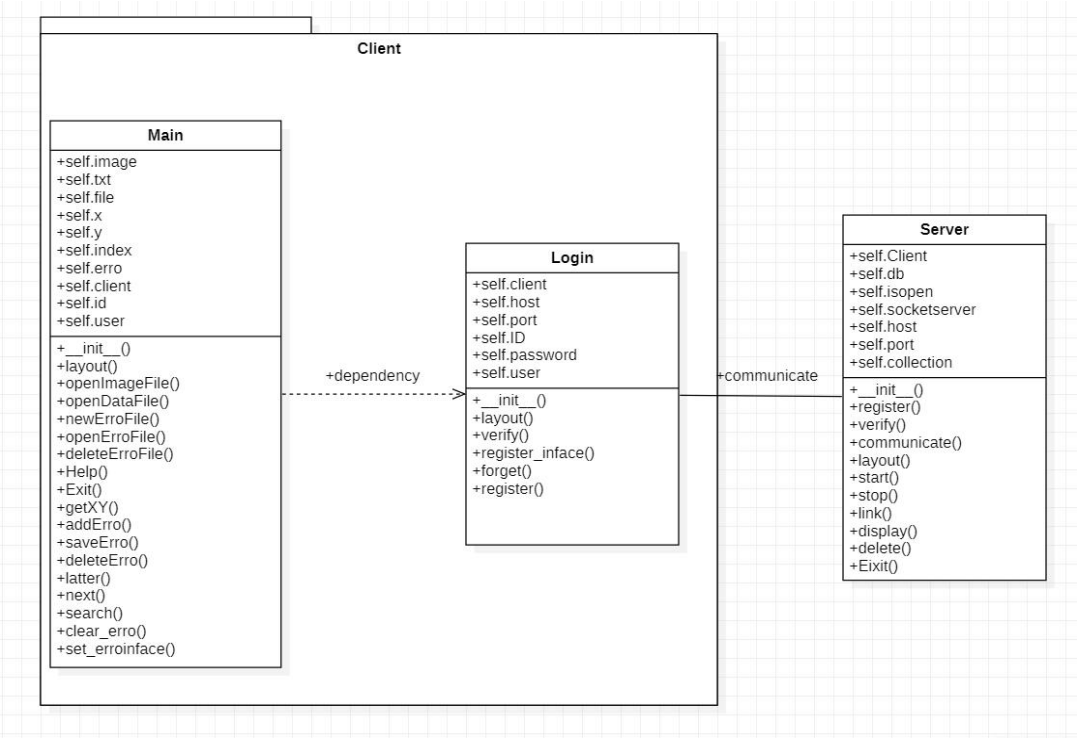
优点：
①多个视图与一个模型相对应
②具有良好的移植性
③系统被分为三个独立的部分，当功能发生变化时只需要修改一个部分
缺点：
①增加了系统设计和运行的复杂性

②视图和控制器连接过于紧密，妨碍了两者的重用
③视图访问模型的效率变低
④频繁访问未变化的数据，降低系统的性能

5. 关键代码

5.1 程序概述

本次项目采取的主要是 Python 语言，所采用的可视化工具为 tkinter 库，服务端和客户端的连接工具采用 socket 完成，数据库使用的是 MongDB 数据库。类图和框架如下：




```
server.py > ...
1   from multiprocessing.dummy.connection import Connection ···
11
12  class Server:#服务器类
13      def __init__(self): ···
26
27      def register(self,id,password,user):#注册用户 ···
40
41      def verify(self,id,password):#验证密码是否正确,返回值为[状态，用户名] ···
57
58      def communicate(self,clientsocket):#负责与客户端的通信 ···
124
125     def layout(self):#服务器界面布局 ···
141
142     def start(self):#启动或停止服务器 ···
150
151     def stop(self):#关闭服务器 ···
158
159     def link(self):#连接客户端 ···
171
172     def display(self):#显示用户列表 ···
177
178     def delete(self,event):#注销用户 ···
192
193     def Exit(self):#退出 ···
197
198
199
200 Server()
201
```

```
class login:#登录类
    def __init__(self):#构造函数…

    def layout(self):#界面布局…

    def verify(self):#登录…

    def register_inface(self):#注册界面…

    def forget(self):#忘记密码…

    def register(self):#创建新用户…


login()
```

```
class Main:#主程序框架类
    def __init__(self):#构造函数…

    def layout(self):#主界面的初始布局设置…

    def openImageFile(self):#打开图像文件并显示…

    def openDataFile(self):#打开数据文件并显示…

    def newErroFile(self):#新建错误文件…

    def openErroFile(self):#打开错误文件…

    def deleteErroFile(self):#删除错误文件…

    def Help(self):#帮助…

    def Exit(self):#退出…

    def getXY(self,event):#获取点击处坐标，并显示…

    def addErro(self):#添加错误到self.erro…

    def saveErro(self):#保存错误文件…

    def deleteErro(self):#删除此条错误信息…

    def latter(self):#上一条…

    def next(self):#下一条…

    def search(self):#在已经打开的错误文件和已经添加的错误信息中查询错误信息…

    def clear_erro(self):#清空错误区…

    def set_erroinface(self,x,y,erro_type,erro_detail):#设置错误界面…
```

5.2 界面设置

主要封装在 layout 函数中，里面实现了各种构件的布局工作。部分关键代码如下（这里仅给出了主程序的界面设置代码，其他界面的代码可以参考附件中的源代码，全部分装在 layout 函数中）：

```
def layout(self):#主界面的初始布局设置
    self.root.title('地图质检')#设置窗口标题
    self.root.geometry('1200x600')#设置窗口大小
    self.root.resizable(width=False,height=False)#大小不可变
    #菜单选项设置
    self.mainmenu=Menu(self.root)#创建主菜单
    self.menuData=Menu(self.mainmenu)#菜单分组
    self.mainmenu.add_cascade(label='数据文件',menu=self.menuData)#添加一个文件分组
    self.menuData.add_command(label='打开图像数据',command=self.openImageFile)#在下面添加选项
    self.menuData.add_command(label='打开文本数据',command=self.openDataFile)
    self.menuErro=Menu(self.mainmenu)
    self.mainmenu.add_cascade(label='错误信息',menu=self.menuErro)
    self.menuErro.add_command(label='新建错误文件',command=self.newErroFile)
    self.menuErro.add_command(label='打开错误文件',command=self.openErroFile)
    self.menuErro.add_command(label='删除错误文件',command=self.deleteErroFile)
    self.menuOther=Menu(self.mainmenu)
    self.mainmenu.add_cascade(label='其他',menu=self.menuOther)
    self.menuOther.add_command(label='帮助',command=self.Help)
    self.menuOther.add_command(label='退出',command=self.Exit)
    self.root.config(menu=self.mainmenu)#设为菜单
    #分割线设置
    self.sep=Separator(self.root,orient=VERTICAL)#竖直分割线
    self.sep.pack(padx=10,pady=3,fill='y',expand=True)
    #框架设置
    self.frame1=tk.Frame(self.root,bd=1,height=550,width=550,relief='solid')#此框架为显示地图窗口
    self.frame1.place(x=25,y=35)
    self.frame2=tk.Frame(self.root,bd=1,height=350,width=550,relief='solid')#此框架为添加错误记录窗口
    self.frame2.place(x=625,y=10)
```

5.3 数据文件操作

（1）打开图像数据：在打开前做了数据清理工作。使用 resize 函数将使图片标签适应窗口大小。然后通过 socket'对象的 send 方法，将图像数据打包成二进制的数据之后传递给服务端（具体见下面注释）。最后和获取坐标的函数 getXY 建立联系。

```
def openImageFile(self):#打开图像文件并显示
    for data in self.frame1.winfo_children():
        data.destroy()  # 清理已打开的数据
    filename=tkf.askopenfilename()#打开文件
    self.label2.config(text=filename)#在上方显示文件路径
    if filename!='':
        #将图像文件传输给服务端储存
        self.client.send('image'.encode("utf-8"))#发送图像数据标志
        time.sleep(0.1)
        self.client.send(self.id.encode("utf-8"))
        time.sleep(0.1)
        fhead = struct.pack(b'128sq', bytes(os.path.basename(filename), encoding='utf-8'),
                            os.stat(filename).st_size)#将xxx.jpg以128sq的格式打包
        self.client.send(fhead)
        time.sleep(0.1)
        f=open(filename,'rb')#以二进制只读打开图像文件
        while True:
            data=f.read(1024)#读入数据
            if not data:#传输完成
                self.client.send('EOF'.encode("utf-8"))
                break
            self.client.send(data)
        f.close()#关闭文件
        #图片显示
        im=Image.open(filename)#打开图片
        im=im.resize((545,545),Image.ANTIALIAS)#调整大小
        self.image=ImageTk.PhotoImage(im)#将图像存入全局变量中
        label_image=tk.Label(self.frame1,image=self.image)#将图像借助标签显示
        label_image.pack()
        label_image.bind("<Double-Button-1>",self.getXY)#双击左键获取坐标
```

（2）打开文本数据：基本与打开图像数据类似。

```
def openDataFile(self):#打开数据文件并显示
    for data in self.frame1.winfo_children():
        data.destroy()  # 清理已打开的数据
    filename=tkf.askopenfilename()#打开文件
    self.label2.config(text=filename)#在上方显示文件路径
    if filename!='':
        with open(filename, encoding='utf-8', errors='ignore') as f:
            content = f.readlines()  # 按行读取
        self.txt=content#将文本数据存到全局变量中
        #把文本文件发送给服务端储存起来
        self.client.send('txt'.encode("utf-8"))#发送文本数据标志
        time.sleep(0.1)
        self.client.send(self.id.encode("utf-8"))
        time.sleep(0.1)
        self.client.send(os.path.basename(filename).encode("utf-8"))
        time.sleep(0.1)
        for i in range(len(content)):
            self.client.send(content[i].encode("utf-8"))#一条一条发送数据
            time.sleep(0.01)#防止黏包
        self.client.send('EOF'.encode("utf-8"))#发送传输完成标志
        # 将读取到的内容放入Listbox
        lb = tk.Listbox(self.frame1,height=30,width=80)
        lb.pack()
        for line in content:
            lb.insert('end', line)
        lb.bind("<Double-Button-1>",self.getXY)#双击左键获取坐标
```

5.4 错误文件操作

（1）新建错误文件：新建前先做数据清理工作，然后获取用户想要新建的文件名，以追加的方式新建错误文件。

```
def newErroFile(self):#新建错误文件
    if self.erro!=[]:
        if messagebox.askokcancel('提示', '确定打开新的错误文件吗？\n如果错误信息未保存\n信息会丢失')==False:
            return None
    self.erro=[]#错误列表清空
    self.index=1#错误索引归零
    self.label12.config(text=str('1/1'))#重置页面
    self.clear_erro()
    if self.file!=None:#若果已打开文件则关闭
        self.file.close()
    filename = askstring('新建错误文件', prompt='输入您创建的文件名', initialvalue='Erro')#打开对话框让用户输入文件名
    self.file=open(filename+'.txt','a+',encoding='utf-8')#新建文件
    self.file.write('12')
    if self.file!=None:#新建成功，给出提示
        messagebox.askokcancel('提示', '成功新建文件')
    else:#新建失败，给出提示
        messagebox.askokcancel('提示', '新建文件失败')
```

（2）打开错误文件：打开文件前先做清理工作，清理完毕之后使用askopenfilename来获取用户想要打开的错误文件。以追加的方式打开该错误文件，然后给出提示。

```
def openErroFile(self):#打开错误文件
    if self.erro!=[]:
        if messagebox.askokcancel('提示', '确定打开新的错误文件吗？\n如果错误信息未保存\n信息会丢失')==False:
            return None
    self.erro=[]#错误列表清空
    self.index=1#错误索引归零
    self.label12.config(text=str('1/1'))#重置页面
    self.clear_erro()
    if self.file!=None:#如果已经有文件打开就将其关闭
        self.file.close()
    filename=tkf.askopenfilename()#打开文件对话框，储存文件名
    self.file=open(filename,'a+',encoding='utf-8')#以追加的方式打开文件
    if self.file!=None:#打开成功，给出提示
       messagebox.askokcancel('提示', '成功打开文件')
    else:#打开失败，给出提示
        messagebox.askokcancel('提示', '打开文件失败')
```

（3）删除错误文件：使用 askokcancel 询问用户是否删除文件，得到确认之后使用 os.path.exists()方法删除错误文件。

```
def deleteErroFile(self):#删除错误文件
    filename =tkf.askopenfilename()  # 文件路径
    if messagebox.askokcancel('提示', '是否删除'+filename)==True:
        os.remove(filename)#删除文件
        if os.path.exists(filename):  # 如果文件存在
            messagebox.askokcancel('提示', '删除文件失败')
        else:
            messagebox.askokcancel('提示', '删除文件成功')
```

5.5 错误信息操作

（1）获取错误坐标：使用 event 的 x 和 y 属性直接获取点击处的坐标，并且通过标签的 config 方法显示在相应区域。

```
def getXY(self,event):#获取点击处坐标，并显示
    self.x=event.x#存储坐标
    self.y=event.y
    self.label10.config(text=str(self.x))#改变标签中的数值
    self.label11.config(text=str(self.y))
```

（2）添加错误信息：添加前先判断坐标值是否合理，合理的话才添加，并且改变页面标签信息。

```
def addErro(self):#添加错误到self.erro
    if self.x>=0 and self.y>=0:#坐标合理时
      erro_type=self.combobox.get()#获取错误类型
      erro_detail=self.text1.get(1.0,END)#获取错误描述
      self.erro+=[[self.x,self.y,erro_type,erro_detail]]#加入一条错误
      self.clear_erro()#清空
      self.index=len(self.erro)+1
      self.label12.config(text=str(str(self.index)+'/'+str(len(self.erro)+1)))#更新下面的页面信息
    else:
        messagebox.askokcancel('提示', '请双击图像以获得一个合理的坐标')
```

（3）删除错误信息：现判断是否还有错误信息在错误列表中，如果没有给出提示，如果有则使用 pop 函数删除，删除的同时改变页面所展示的信息。

```
def deleteErro(self):#删除此条错误信息
    if len(self.erro)==0:#如果没有错误信息
        messagebox.askokcancel('提示', '无错误信息')
    elif len(self.erro)==self.index:#如果是最后一项错误信息，则特殊处理（清空页面）
        self.erro.pop(self.index-1)
        self.clear_erro()#清空界面
        self.label12.config(text=str(self.index)+'/'+str(len(self.erro)+1))#更新下面的页面信息
    else:
        if self.index>=len(self.erro)+1:
            messagebox.askokcancel('提示', '此条信息未编辑')
        else:
            print(self.index)
            self.erro.pop(self.index-1)
            self.label12.config(text=str(self.index)+'/'+str(len(self.erro)+1))#更新下面的页面信息
            self.set_erroinface(self.erro[self.index-1][0],self.erro[self.index-1][1],self.erro[self.index-1][2],self.erro[self.index-1][3])
```

（4）保存错误信息：如果没有错误文件打开，则使用 asksaveasfilename 获取用户想要存储的文件名，并且以追加的方式打开文件。使用 write 韩式将错误列表中的错误信息按照一定的格式写入打开错误文件的末尾。保存完毕之后，首先将错误文件的文件名、用户 id 发送给服务端，再打开保存的错误文件，将其通过 socket 的 send 方法一行一行的发送给服务端存储起来。

```
def saveErro(self):#保存错误文件
    if self.file==None:#没有新建或者打开文件
        self.filename=tkf.asksaveasfilename()#弹出保存文件对话框
        if self.filename=='':#如果没有选择文件
            return None
        self.file=open(self.filename,'a+',encoding='utf-8')
    for i in range(len(self.erro)):
        self.file.write('('+str(self.erro[i][0])+','+str(self.erro[i][1])+')    '+self.erro[i][2]+'    '+self.erro[i][3])#写入信息
    messagebox.askokcancel('提示', '文件保存成功')
    self.file.close()
    #将保存的错误文件传送给服务器
    self.client.send('erro'.encode("utf-8"))#发送错误文件传输标志
    time.sleep(0.1)
    self.client.send(self.id.encode("utf-8"))#发送账号
    time.sleep(0.1)
    self.client.send(os.path.basename(self.filename).encode("utf-8"))
    time.sleep(0.1)
    f=open(self.filename,'r',encoding='utf-8')#打开错误文件
    while True:
        content=f.read()
        if not content:#传输完成
            self.client.send('EOF'.encode("utf-8"))#发送传输完成标志
            break
        self.client.send(content.encode("utf-8"))#一条一条发送数据
        time.sleep(0.01)#防止黏包
    f.close()
    #清除工作
    self.clear_erro()
    self.erro=[]
    self.file=None
    self.index=1
    self.filename=''
    self.label12.config(text=str('1/1'))#重置页面
```

（5）查看上一条错误信息：先判断是否为第一条信息，如果是就给出提示；如果不是就把上一条信息展示出来。

```
def latter(self):#上一条
    if self.index==1:
        messagebox.askokcancel('提示', '已经是第一条')
    else:
        self.index-=1
        self.label12.config(text=str(self.index)+'/'+str(len(self.erro)+1))#更新下面的页面信息
        self.set_erroinface(self.erro[self.index-1][0],self.erro[self.index-1][1],self.erro[self.index-1][2],self.erro[self.index-1][3])#
```

（6）查看下一条错误信息：先判断是否为最后一条信息，如果是则给出提示；否则，跳转到下一条信息，并且显示出来。

```
def next(self):#下一条
    if self.index==len(self.erro)+1:
            messagebox.askokcancel('提示', '已经是最后一条')
    else:
        if self.index==len(self.erro):#如果是倒数第二条，则清空界面（最后一条还未编辑）
            self.index+=1
            self.label12.config(text=str(self.index)+'/'+str(len(self.erro)+1))#更新下面的页面信息
            self.clear_erro()
        else:
            self.index+=1
            self.label12.config(text=str(self.index)+'/'+str(len(self.erro)+1))#更新下面的页面信息
            self.set_erroinface(self.erro[self.index-1][0],self.erro[self.index-1][1],self.erro[self.index-1][2],self.erro[self.index-1][3])
```

（7）查询错误信息：这里获取输入框里面的关键字之后要注意获取的关键字是带有换行符”\n”的，所以先去掉之后再使用 find 函数在错误文件和错误列表中查询。

```
def search(self):#在已经打开的错误文件和已经添加的错误信息中查询错误信息
    self.text3.delete(1.0,END)#清空展示框
    search_content=self.text2.get(1.0,END)#获取输入的关键词
    search_content=search_content[0:len(search_content)-1]#上面读取的字符串最后带有\n一定要去除
    search_result=[]#储存查询结果
    if self.file!=None:#如果有错误文件打开，则先在错误文件中查询
        self.file.seek(0,0)#将指针调到文件开始
        erro_content=self.file.readlines()#按行读取
        for i in range(len(erro_content)):
            if erro_content[i].find(search_content)>=0:#如果关键字被包含在这条错误信息中
                search_result+=[erro_content[i]]
    for i in range(len(self.erro)):#遍历所有错误信息
        #将每条错误信息串联起来便于查询，中间用空格隔开避免数字连接造成查询出现偏差
        erro_content=str(self.erro[i][0])+'  '+str(self.erro[i][1])+'  '+self.erro[i][2]+'  '+self.erro[i][3]
        if erro_content.find(search_content)>=0:#如果关键字被包含在这条错误信息中
            search_result+=['('+str(self.erro[i][0])+','+str(self.erro[i][1])+')    '+self.erro[i][2]+'    '+self.erro[i][3]]
    for i in range(len(search_result)):#展示查询结果
        self.text3.insert(END,search_result[i])
```

5.6 登录操作

（1）客户端：首先获取输入框中的登录信息，之后向服务端发送登录验证标志。后续继续发送用户账号和密码，等待服务器验证。这里连续发送消息时，中间插入 time.sleep（0.1）是为了防止黏包。

```
def verify(self):#登录
    self.ID=self.entry1.get()#获取ID
    self.password=self.entry2.get()#获取密码
    if self.ID!='':
        self.client.send('verify'.encode("utf-8"))#发送标志
        time.sleep(0.1)
        #发送数据，以二进制的形式发送数据，所以需要进行编码
        self.client.send(self.ID.encode("utf-8"))#发送账号
        time.sleep(0.1)#防止黏包
        self.client.send(self.password.encode("utf-8"))#发送密码
        msg = self.client.recv(1024)#接收服务端返回的数据，需要解码
        if msg.decode("utf-8")=='successful':#密码正确
            self.user= self.client.recv(1024)#接受用户名
            self.root.withdraw()#关闭窗口
            messagebox.showinfo('WELCOM','欢迎您,亲爱的'+self.user.decode("utf-8"))
            Main(self.ID,self.user,self.client)#启动主程序
            print(1)
            self.client.close()#关闭客户端
            sys.exit() #退出程序
        elif msg.decode("utf-8")=='wrong':#密码错误
            messagebox.askokcancel('提示', '密码错误')
        elif msg.decode("utf-8")=='nothing':#没有用户
            messagebox.askokcancel('提示','没有此用户')
    else:
        messagebox.askokcancel('提示','账号不能为空')
```

（2）服务端：接受客户端传递过来的信息，使用 find_one 函数在数据库中查找配对，返回状态参数和用户名。（具体见注释）

```
def verify(self,id,password):#验证密码是否正确,返回值为[状态，用户名]
    self.collection=self.db.users#指定集合
    result=self.collection.find_one({'ID':id})#查询id
    if result==None:#没有查询结果
        return [0,None]#表示没有此用户
    elif password==result['PassWord']:#如果用户输入的密码和查询出的密码一致
        t=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())#获取当前时间
        condition = {'ID':id}
        result=self.collection.find_one(condition)
        result['LastTime'] =t
        self.collection.update(condition, result) #修改上一次登陆时间
        self.lb.delete(0,END)
        self.display()
        return [1,result['UserName']]#表示匹配一致
    elif password!=result['PassWord']:#不一致
        return [-1,None]#表示匹配不一致
```

5.7 注册操作

（1）客户端：获取输入框的注册信息，然后传递给客户端，进行注册。

```
def register(self):#创建新用户
    self.client.send('register'.encode("utf-8"))#发送标志
    time.sleep(0.1)
    #获取注册信息
    id=self.entry3.get()
    password=self.entry4.get()
    user=self.entry5.get()
    #向服务器发送注册信息
    self.client.send(id.encode("utf-8"))#发送账号
    time.sleep(0.1)#防止黏包
    self.client.send(password.encode("utf-8"))#发送密码
    time.sleep(0.1)
    self.client.send(user.encode("utf-8"))#发送用户名
    msg=self.client.recv(1024)#接收服务器反馈的消息
    if msg.decode("utf-8")=='successful':
        messagebox.askokcancel('提示','注册成功')
        self.winNew.withdraw()#关闭注册窗口
    if msg.decode("utf-8")=='fail':
        messagebox.askokcancel('提示','注册失败（该账号已经被占用）')
```

（2）服务端：根据传递过来的信息，调用 verify 函数在数据库中查找，如果没有此账号，则使用 insert 方法插入数据库。同时在 user 目录下使用 os 中的 makedirs 方法创建该用户的文件夹。并且返回状态参数，0 表示注册成功，-1 表示 ID 已经被占用。

```
def register(self,id,password,user):#注册用户
    if self.verify(id,password)[0]==0:#如果数据库中没有此id
        data={'ID':id,'PassWord':password,'UserName':user,'LastTime':''}#传入的用户信息
        self.collection.insert(data)#存入数据库中的users集合
        os.makedirs('.\\user\\'+id)#创建该用户的数据文件夹
        os.makedirs('.\\user\\'+id+'\\txt')#创建该用户的文本文件夹
        os.makedirs('.\\user\\'+id+'\\image')#创建该用户的图像文件夹
        os.makedirs('.\\user\\'+id+'\\erro')#创建该用户的错误数据文件夹
        self.lb.delete(0,END)
        self.display()
        return 0#表示注册成功
    else:
        return -1#表示ID已经被占用
```

5.8 连接与交互

（1）客户端：每开启一个客户端就启动一个 socket 对象，尝试与服务端进行连接。

```
def __init__(self):#构造函数
    #创建一个客户端的socket对象
    self.client = socket.socket()#socket.AF_INET, socket.SOCK_STREAM
    self.host = "192.168.73.1"#设置服务端的ip地址为服务端ip地址192.168.73.1
    self.port = 5000#设置端口
    self.client.connect((self.host,self.port))#连接服务端
```

（2）服务端：

①启动服务器：这里使用了 Thread 实现多线程

```
def start(self):#启动或停止服务器
    if self.isopen==0:#如果是关闭状态，则开启
        self.isopen=1#状态改变
        self.label2.config(text='ON',background='green')
        t=threading.Thread(target=self.link)#开辟一个线程连接客户端
        t.start()
    else:
        messagebox.askokcancel('提示','服务器已启动')
```

```
def link(self):#连接客户端
    #创建服务端的socket对象socketserver
    self.socketserver = socket.socket()
    self.host = '192.168.73.1'#本机ip地址
    self.port = 5000
    self.socketserver.bind((self.host, self.port))#绑定地址（包括ip地址和端口号）
    self.socketserver.listen(5)#设置监听
    #启动客户端和服务端的交互
    while True:
        clientsocket,addr = self.socketserver.accept()#等待客户端的连接,accept()函数会返回一个元组,元素1为客户端
        t=threading.Thread(target=self.communicate,args=(clientsocket,))#为一个客户端开辟一个线程开辟一个线程
        t.start()#启动线程
```

②与客户端交互：不断地循环接受客户端的消息，有消息传来判断是要执行什么功能，进入相应模块，调用相应函数，实现相应功能。

```
def communicate(self,clientsocket):#负责与客户端的通信
    while True:#循环接受
        msg=clientsocket.recv(1024)#接受客户端发来的消息
        strmsg=msg.decode("utf-8")#将接受的数据解码
        if strmsg=='register':#如果是要注册新用户
            msg_id=clientsocket.recv(1024)#接收账户
            strmsg_id=msg_id.decode("utf-8")
            msg_password=clientsocket.recv(1024)#接收密码
            strmsg_password=msg_password.decode("utf-8")
            msg_user=clientsocket.recv(1024)#接收用户名
            strmsg_user=msg_user.decode("utf-8")
            if self.register(strmsg_id,strmsg_password,strmsg_user)==0:#注册成功
                str='successful'
                clientsocket.send(str.encode("utf-8"))#给客户端发送成功标志
            else:#注册不成功
                str='fail'
                clientsocket.send(str.encode("utf-8"))#给客户端发送失败标志
        if strmsg=='verify':#如果是要验证登录
            msg_id=clientsocket.recv(1024)#接收账户
            strmsg_id=msg_id.decode("utf-8")
            msg_password=clientsocket.recv(1024)#接收密码
            strmsg_password=msg_password.decode("utf-8")
            result=self.verify(strmsg_id,strmsg_password)#调用验证函数，得到判断结果列表
            if result[0]==0:#没有这个用户
                clientsocket.send('nothing'.encode("utf-8"))#给客户端发送没有此用户
            if result[0]==-1:#密码不匹配
                clientsocket.send('wrong'.encode("utf-8"))#给客户端发送密码错误
            if result[0]==1:#匹配成功
                clientsocket.send('successful'.encode("utf-8"))#给客户端发送匹配成功
                time.sleep(0.1)
                clientsocket.send(result[1].encode("utf-8"))#并且发送用户名
```

```
if strmsg=='erro':#如果是传输错误数据文件
    msg_id=clientsocket.recv(1024).decode("utf-8")#接受id
    msg_filename=clientsocket.recv(1024).decode("utf-8")#接受文件名
    print(msg_filename)
    file=open('.\\user\\'+msg_id+'\\erro\\'+msg_filename,'w')#在该用户的目录下创建一个文件
    while True:
        msg_content=clientsocket.recv(1024).decode("utf-8")#接受内容
        if msg_content=='EOF':#传输完成
            break
        file.write(msg_content)#写入数据
    file.close()#关闭文件
if strmsg=='txt':#如果是传输文本文件
    msg_id=clientsocket.recv(1024).decode("utf-8")#接受id
    msg_filename=clientsocket.recv(1024).decode("utf-8")#接受文件名
    file=open('.\\user\\'+msg_id+'\\txt\\'+msg_filename,'w')#在该用户的目录下创建一个文件
    while True:
        msg_content=clientsocket.recv(1024).decode("utf-8")#接受内容
        if msg_content=='EOF':#传输完成
            break
        file.write(msg_content)#写入数据
    file.close()#关闭文件
if strmsg=='image':#如果传输的是图像文件
    msg_id=clientsocket.recv(1024).decode("utf-8")#接受id
    fileinfo_size = struct.calcsize('128sq')
    buf = clientsocket.recv(fileinfo_size)  # 接收图片名
    if buf:
        filename, filesize = struct.unpack('128sq', buf)
        fn = filename.decode().strip('\x00')
        fp = open('.\\user\\'+msg_id+'\\image\\'+fn, 'wb')#在相应的位置打开一个文件写入数据
        while True:
            msg_content=clientsocket.recv(1024)#接受内容
            if msg_content=='EOF':#传输完成
                break
            fp.write(msg_content)  # 写入图片数据
        fp.close()
```

5.9 用户管理

将存放用户列表的 ListBox 与双击左键绑定，响应函数为 delete。Delete 会使用数据库集合对象的 delete 函数将该用户从数据库中删除，然后使用 shutil 的 rmtree 函数将其数据文件夹请删除，最后刷新页面。

```
def delete(self,event):#注销用户
    index=self.lb.curselection()#记录被选中的序号
    content=self.lb.get(index)#获取这一项的内容
    self.collection=self.db.users#指定集合
    i=0
    for i in range(len(content)):
        if content[i]==' ':
            break
    id=content[0:i]#截取用户id
    if messagebox.askokcancel('提示','确定要注销此用户吗？')==True:
        shutil.rmtree('.\\user\\'+id)#删除该用户的文件夹
        self.collection.remove({'ID':id})#从数据库中删除
        self.lb.delete(0,END)
        self.display()#刷新页面
```

5.10 其他操作

（1）帮助：使用提示窗口的形式给出信息

```
def Help(self):#帮助
    messagebox.showinfo('帮助','作者：马文卓\n版本:2.0\n语言:Python\n库:tkinter')
```

（2）退出：退出前先询问是否退出

```
def Exit(self):#退出
    if messagebox.askokcancel('提示', '您确定退出程序吗？')==True:#弹出提示框等待选择
        if self.file!=None:#如果文件还未关闭，则关闭文件
            self.file.close()
        self.root.withdraw()#如果确认则退出窗口
        sys.exit() #退出程序
```

（3）忘记密码：给出提示

```
def forget(self):#忘记密码
    messagebox.askokcancel('帮助', '联系邮箱:2574485753@qq.com')
```

6. 结果展示

代码运行结果见附件《运行视频.mp4》，详细代码见附件《client.py 和 Server.py》，运行文件为《client.exe 和 Server.exe》，项目相关图示均见附录。

7. 总结评价

本项目主要实现了局域网环境下的的地图质检系统,实现了多用户同时登录使用，以及相同用户不同终端使用的功能。

完成度方面,实现了所有功能要求,基本达到了项目最初的预期。与此同时,还添加了一些优化功能，如帮助、退出等，以及一系列的边界限制操作，大大的增加了软件的鲁棒性和可用性，赋予了其高度的稳定性。

在重用性上本项目得到了极大的体现，main 类的主框架基本沿用之前的单机版，只是在里面加入了文件传输功能。除此之外，添加了 login 类和 server 类，以完成整个项目。

本项目还有较强的扩展性，后续还可以向互联网版去发展。