**项目开发文档**

**项目名称： 气象数据分析**

**项目组长：** 尚达

**项目组员：** 王贺盟 施泽宇 徐帅 何文龙

**起止时间：** 6月29日-7月12日

**指导老师：** 李宇

**文档目录**

1. **项目介绍**
   1. 项目背景
   2. 功能需求
   3. 项目成员分工
2. **项目设计与实现**
   1. 项目计划
   2. 项目实施
   3. 版本控制
3. **使用手册**
4. **项目测试**
5. **项目总结**

**1.项目介绍**

1.1项目背景

本次实训要求完成一个基于天气历史数据的分析预测前端程序和后台服务两部分，需要提交的材料包括：项目文档，项目代码，答辩PPT。本次实训项目是基于时间序列模型对历史气象数据进行分析，数据采集NCDC气象官方网站。主要功能是对北京地区1980年至2020年的气象数据进行ETL清洗后，使用spark对数据进行筛选，筛选出某年中每一天的最低气温，最高气温和平均气温，然后使用Python对数据进行时间序列分析，预测一周的气温走势。最后，使用web终端对分析的结果进行可视化展示。

1.2功能需求

根据数据形成可视化报表

获取数据层中的数据

应用层层

数据层

数据建模

数据.csv→选择合适的模型进行分析预测

数据处理

历史数据.csv→数据.csv

数据采集

历史数据.csv

账户管理

用户登录等操作

**数据采集、处理**

从获取的所有历史数据中筛选出需要的数据

**模型构建**

根据已有的数选择合适的模型进行拟合并实现预测

**账户管理**

用户的注册、登录、修改密码、修改用户名等

**前后端连接**

实现前端网页、服务器与数据层的连接

**可视化**

将数据层给出的预测数据绘制成可视化图表

1.3组员分工

尚达： 用户数据库的建立，虚拟机集群的配置，数据的清洗

何文龙：服务器和客户端的交互实现，代码的整合

flask框架的建立（讨论之后未使用）

施泽宇：fbprophet模型的构建，数据的处理和json文件建立

java读取json文件并进行处理

knn模型的建立（未使用）

徐帅：前端网页界面的实现，多个界面的跳转和显示功能实现

王贺盟：SARIMA模型的建立，json接口实现（比较之后未使用）

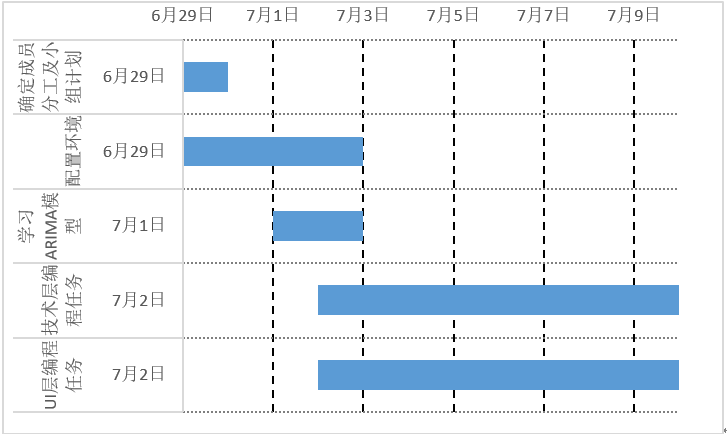
数据的收集

进行部分功能的测试和调试

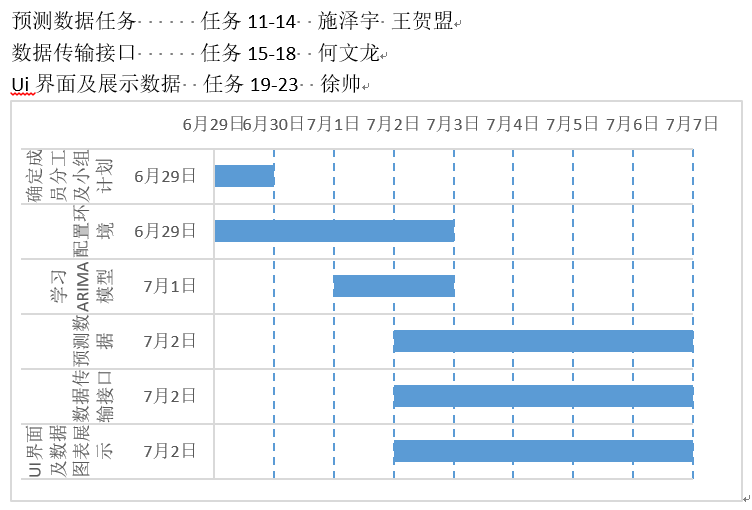
文档的编写

**2.项目设计与实现**

2.1项目计划



最初的项目实施计划如上图，但是在中期之后我们发现了一些问题，如进度缓慢，成员分配不合理，资源利用不充分等，因此对计划做出了如下修改



2.2项目实施

**1.模型构建**

我们寻找了很多模型，如ARMA、KNN、Prophet等，期间也探索过相对于ARMA更为严谨的SARIMAX模型，但是由于个人能力问题，在尝试了很久之后也没能使模型达到足够精确的标准，所以最后选用了Prophet模型。

Prophet是facebook开源的一个时间序列预测工具,使用了时间序列分解与机器学习拟合的方法Facebook 所提供的 prophet 算法不仅可以处理时间序列存在一些异常值的情况，也可以处理部分缺失值的情形，还能够几乎全自动地预测时间序列未来的走势。从论文上的描述来看，fbprophet 算法是基于时间序列分解和机器学习的拟合来做的，其中在拟合模型的时候使用了 pyStan 这个开源工具，因此能够在较快的时间内得到需要预测的结果。

Prophet模型实现

在时间序列分析领域，有一种常见的分析方法叫做时间序列的分解（Decomposition of Time Series），它把时间序列分成几个部分，分别是季节项，趋势项，剩余项。也就是说对所有的，都有除了加法的形式，还有乘法的形式，也就是以上式子等价于 。所以，有的时候在预测模型的时候，会先取对数，然后再进行时间序列的分解，就能得到乘法的形式。在 fbprophet 算法中，作者们基于这种方法进行了必要的改进和优化。一般来说，在实际生活和生产环节中，除了季节项，趋势项，剩余项之外，通常还有节假日的效应。所以，在 prophet 算法里面，作者同时考虑了以上四项，也就是：

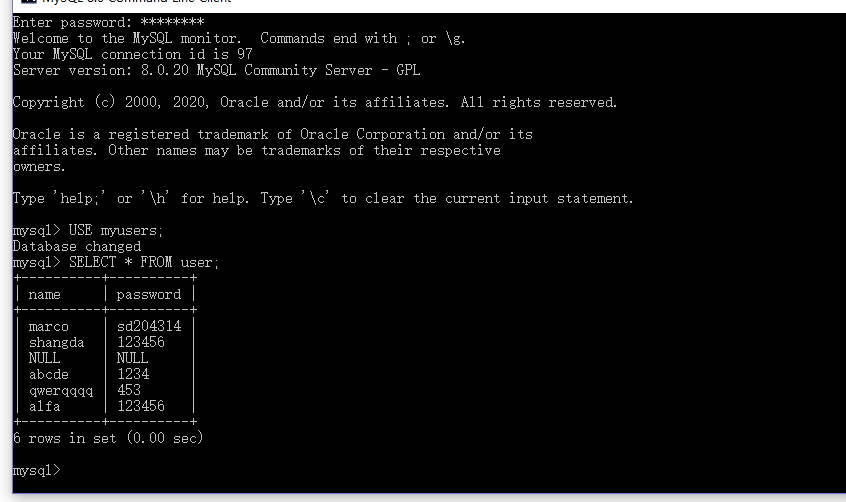
其中表示趋势项，它表示时间序列在非周期上面的变化趋势；表示周期项，或者称为季节项，一般来说是以周或者年为单位；表示节假日项，表示在当天是否存在节假日；表示误差项或者称为剩余项。Prophet 算法就是通过拟合这几项，然后最后把它们累加起来就得到了时间序列的预测值。

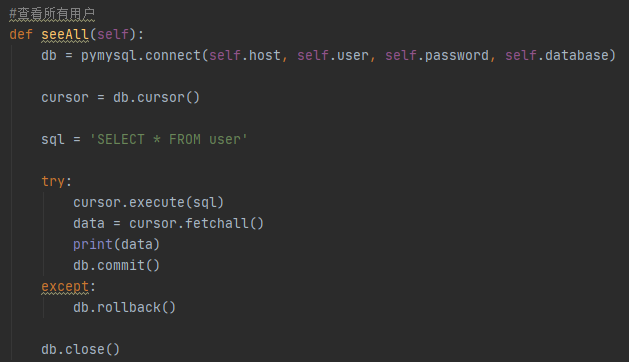
完善工作

根据fbprophet模型，进行对天气数据的预测运算调试参数，使数据结果更加符合预期同时预测出预测数据的波动区间即，实际的数据如果不完全吻合预测数据，也会落在一个温度区间之内由于fbprophet的部分函数的封装，无法直接对输出结果进行处理因此将结果直接写入txt文件之中，再次通过这个txt文件进行数据处理，将数据处理为json文件并保存在前端的java当中，编写了关于读取json文件，处理相应的数据的程序。

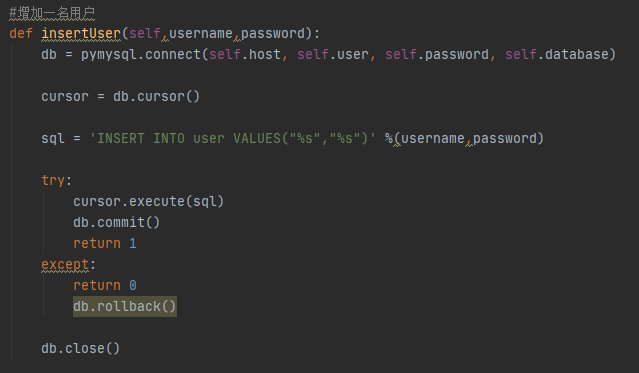
**2.数据库构建**

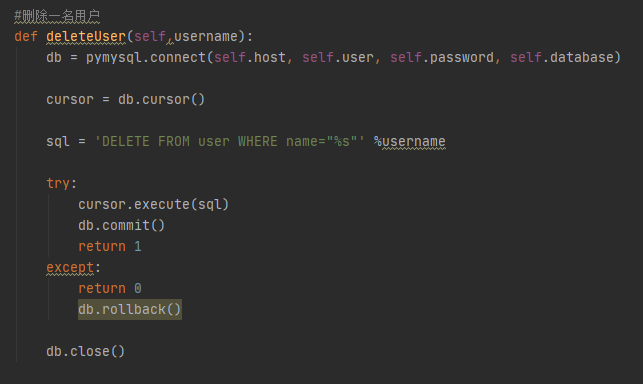
学习mysql数据库的搭建及基本操作，并成功建立一个数据表用于测试学习

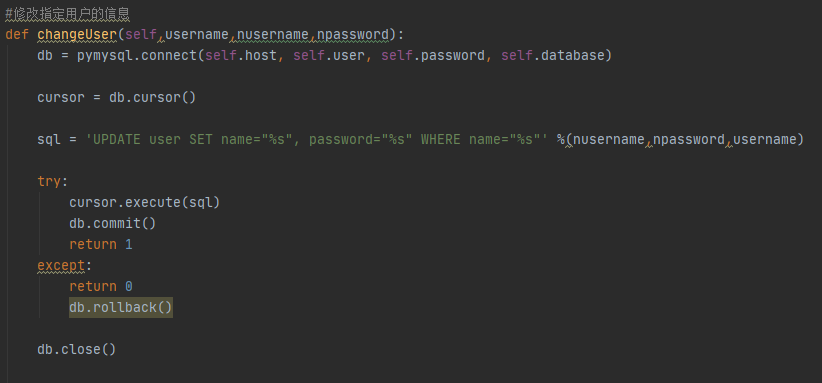


浏览所有用户

判断用户是否存在

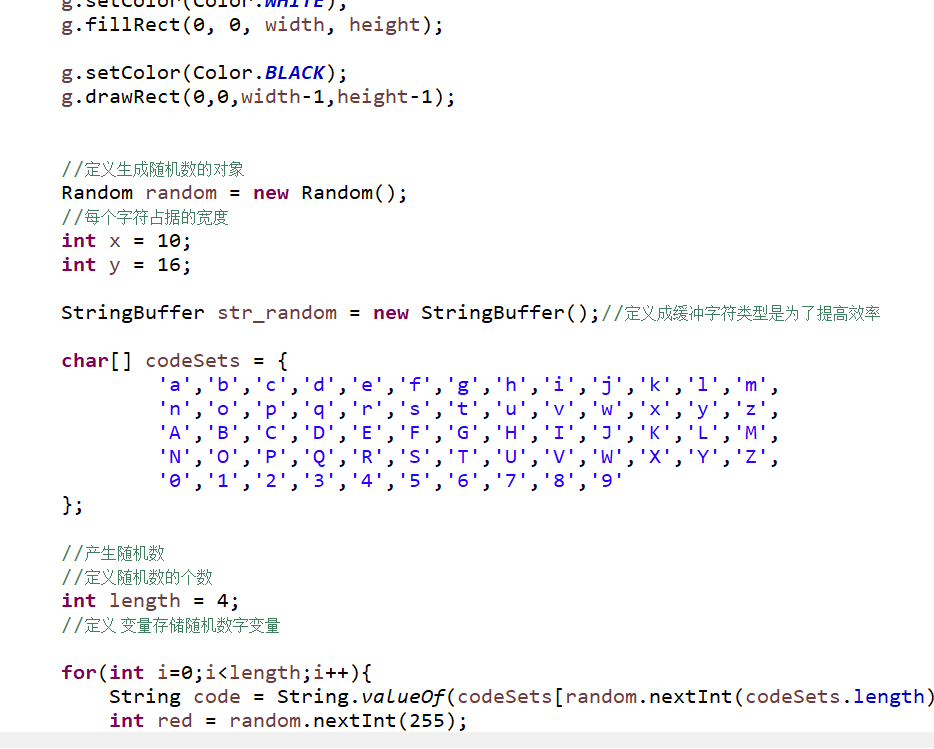
增加一名用户

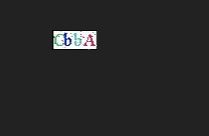
删除一名用户

修改用户信息

**3.前端验证码部分**

根据前端网页的要求编写VerifyCodeServlet类，在其中生成验证码图片，并通过resbonse类来传递到网页端，但由于网页端并未使用resbonse类，导致生成验证码到网页上成了大问题，最终通过合力解决完成了验证码功能。





**4.后台服务器实现**

后台服务器实现：

1. 实现模型的接合和服务器开始连接

①数据库支持函数

在主函数中使用数据库

def \_\_init\_\_(self):

初始化变量，连接数据库

def seeAll(self):

查看所有数据库信息

def isExist(self,username,password):

查看用户是否存在于该数据库中

def insertUser(self,username,password):

插入一名新用户

def deleteUser(self,username):

删除一名新用户

def changeUser(self,username,nusername,npassword):

修改一名用户信息

②服务器接口

def other()

def init()



实现数据库的连接

mysql = MyUserSql()

接收一个新client并开辟线程

thread = Thread(target=acceptclient)  
thread.setDaemon(True)  
thread.start()

def acceptclient()

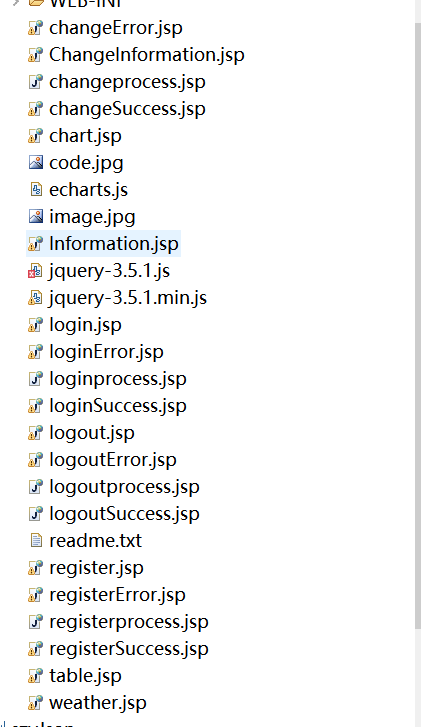
后端向前端传递信息

def message\_handle(client):

def message\_handle(client):  
 while True:  
 szBuf = client.recv(1024)  
 buffer=str(szBuf, **'gbk'**)  
 str1,str2,str3=buffer.split(**"\_"**)  
 print(str1+**" "**+str2+**" "**+str3)  
 if str1==**'register'**:  
 if mysql.isExist(str2,str3):  
 client.send(b'no\n')  
 print(**"failed1"**)  
 else:  
 if mysql.insertUser(str2, str3):  
 client.send(b"yes\n")  
 print(**"success"**)  
 else:  
 client.send(b"no\n")  
 print(**'failed2'**)  
  
 if str1==**'sign'**:  
 if mysql.isExist(str2,str3):  
 client.send(b"yes\n")  
 else:  
 client.send(b"no\n")  
 if str1==**'delete'**:  
 if mysql.isExist(str2,str3):  
 if mysql.deleteUser(str2):  
 client.send(b"yes\n")  
 else:  
 client.send(b"no\n")  
 if str1==**'sendfile'**:  
 filename = **r"D:\pythonfiles\datas\data1.json"** myfile = open(filename, **'rb'**)  
 for readline in myfile:  
 client.send(readline)  
 if str1==**'change'**:  
 if mysql.changeUser(str2,str2,str3):  
 client.send(b'yes\n')  
 else:  
 client.send(b'no\n')  
  
 client.close()  
 print(**"end of servive"**)

前端系统实现：

主要使用tomcat对jsp网页进行部署，所制作的网页文件如下



与后端服务器进行交互的类为SocketClient：

**public** **boolean** Registe(String name,String Password) **throws** IOException前端注册网页所需要代码，在register网页被调用

**public** **boolean** Sign(String name,String Password) **throws** IOException

前端登录网页所需外部支持代码，在login网页被调用

**public** **void** closeall() **throws** IOException

关闭所有输入输出流以及和和服务器的socket连接

在logout网页被调用

**public** **void** readjson() **throws** IOException

读取json数据，用来实现后期echart图表和table的实现

**public** **void** sendfile() **throws** IOException

接收前端传递过来的数据文件

**public** **void** run()

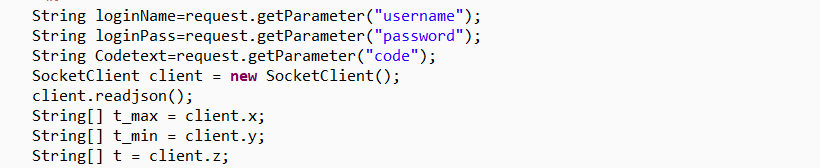
为传输文件单独开辟一个线程

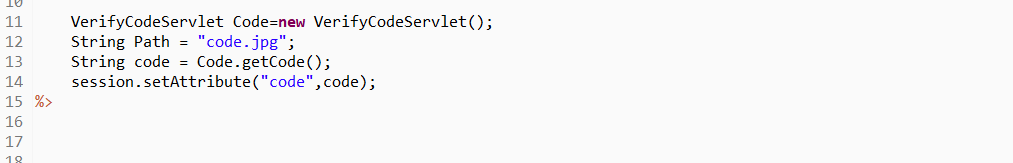
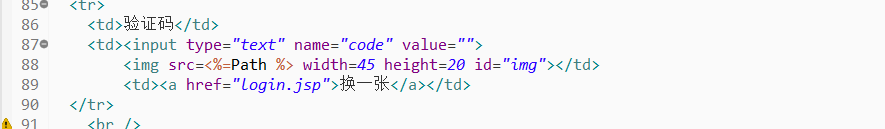
验证码实现的类为VerifyCodeServlet：

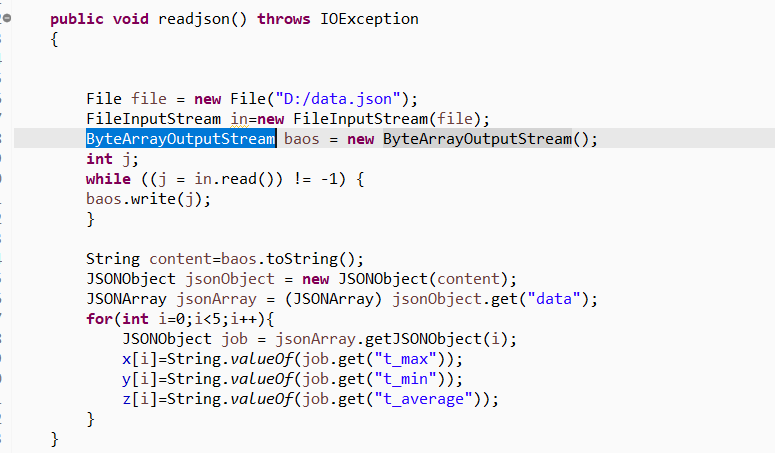
VerifyCodeServlet实现验证码生成

**5.前端网页实现**

前端页面主要有jsp文件实现，主要使用javascript语言。最终的实现方法是在jsp网页通过导入Java客户端的类来实例化对象来和Python服务器部分实现对接，完成数据传输和返回指令。以进行下一步的网页跳转。

登录，注册，注销，修改用户信息主要通过“submit’提交”form“表单将用户输入的信息传递到Python服务器端进行操作。

验证码的实现主要依靠自定义的类VerifyCodeServlet实现。实现过程为现在此类的构造函数中生成一张图片，然后在jsp网页实例化，并将图片的路径赋值个img的src属性。最终对比验证码输入栏和验证码图片上的字符串判定能否登录。主要实现代码：

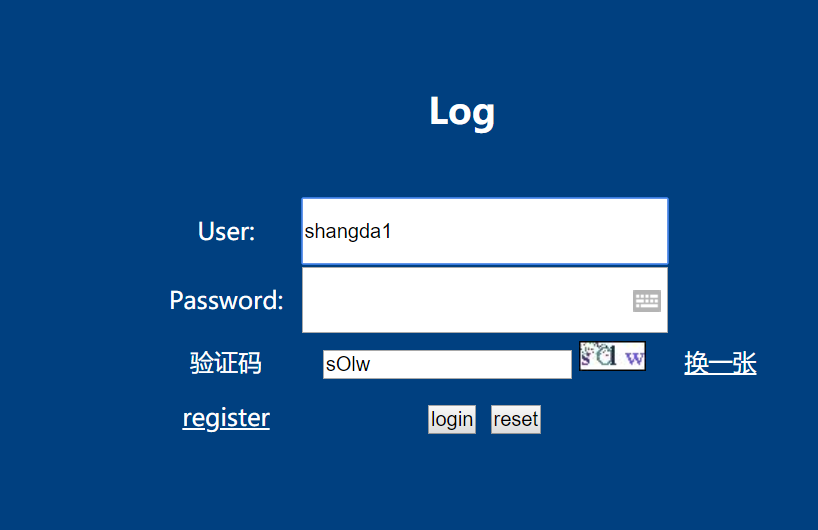
实现Python服务端传输的json格式文件的读取：主要先通过FileInputStream和ByteArrayOutputStream类将文件以字节流的方式读取出来，再将字节流转化为String类型，最后通过类JSONObject将String转化成json格式。在通过JSONArray类将json数据中的每一种属性单独读取出来。主要实现代码：

Table表格的制作，table表格主要通过jsp页面中的“table实现“，先定义表头，再将json格式的变量对应输出在相应的列之下。主要实现代码：

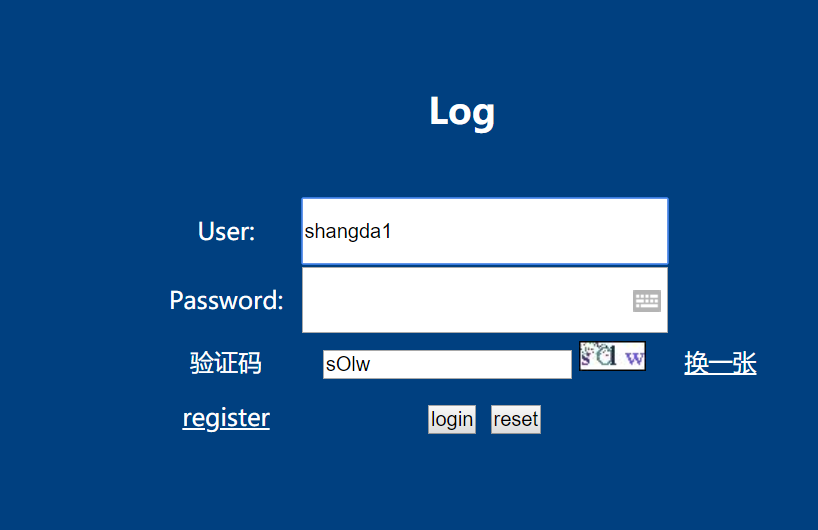
Echarts折线图的展示，主要通过对echarts官网上的模板进行学习，以及网上关于jsp页面制作echarts图的教程。导入了制作echarts图必须的js脚本。成功将气温做成图表形式。主要实现代码：

**3.使用手册**

1. 登录：如下图所示界面，用户可以输入用户名，密码和验证码。如果都正确。点击“login“按钮可以跳转至下一界面进行操作。点击”Reset“按钮，可以清空用户名输入栏和密码输入栏。如果验证码看不清，可以点击验证码图片左侧的按钮刷新图片。若用户还未完成注册，可以点击左下角的”register“链接，跳转至注册界面，进行注册。



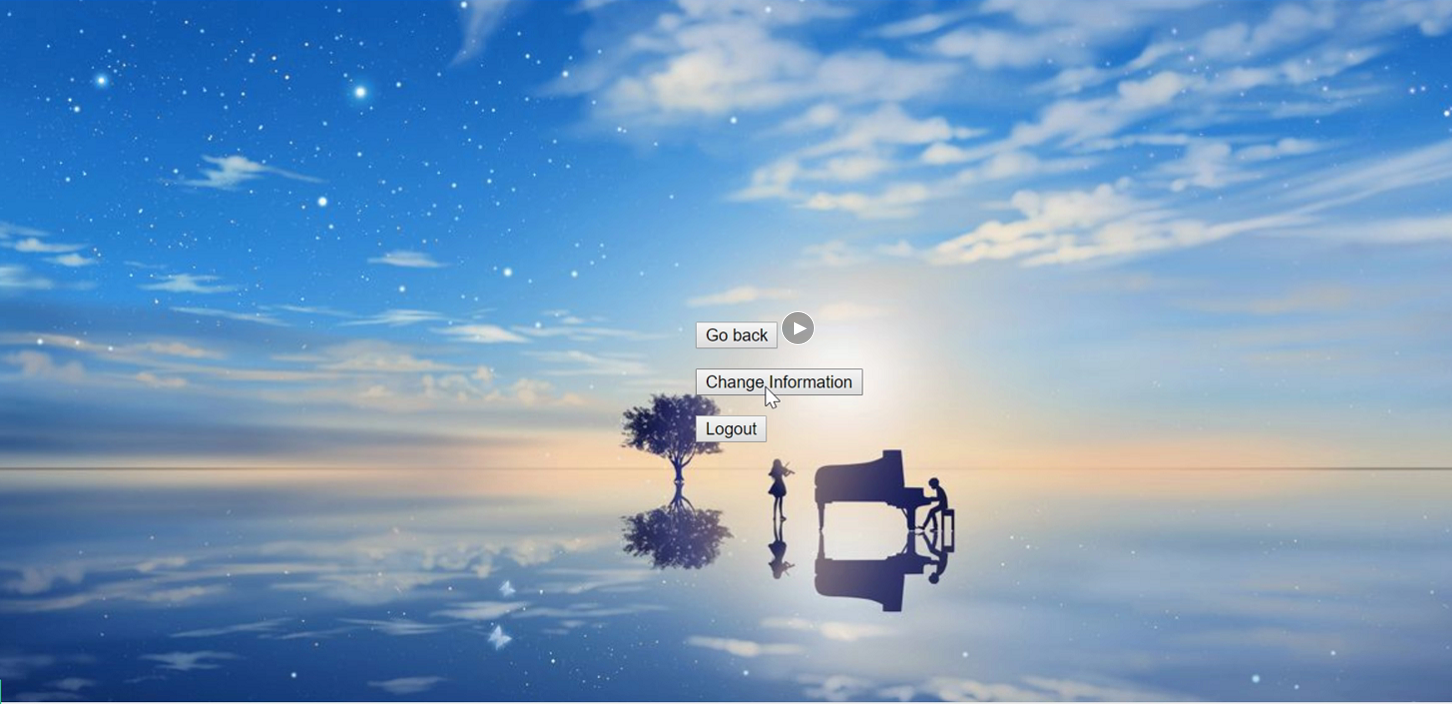
1. 注册：如下图所示界面，点击登录界面的“register“链接，既可进入注册界面，可根据界面输入用户名，密码，及验证码。如若信息正确且用户名和服务器数据库中已存在用户不冲突。这可进入注册成功界面。注册成功界面有一”Login“按钮，点击可返回登录界面进行登录。否则将进入注册失败界面，注册失败界面有一”register again“，点击课返回注册界面重新注册。若想从注册界面直接返回登录界面，可点击左下角”login“链接返回登录。



1. 登录成功：如下图所示界面，可点击“Information“按钮，实现对自己账号信息的操作。点击”weather“按钮，既可进入天气信息查询界面。点击”Go back“按钮，既可返回上一级界面—登录界面。



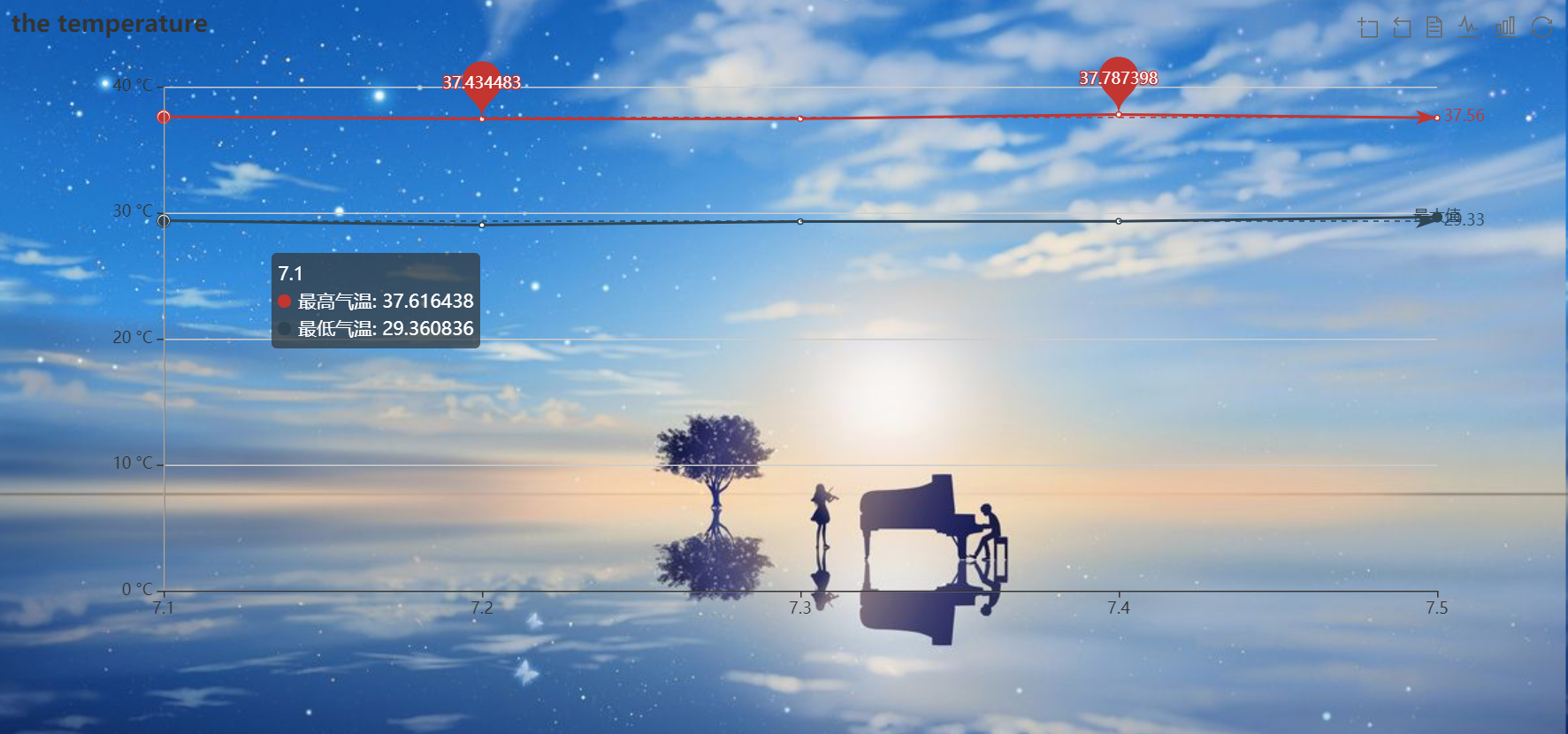
1. 账号信息操作：如下图所示界面，点击“change information：按钮，既可进入修改密码界面。点击“Log out”按钮既可进入注销账户界面。点击“Go back’按钮，既可返回上一级界面。



1. 修改密码：如下图所示界面，根据界面输入新密码，确认新密码。并点击“repair“按钮。若两次输入密码相同，则可改变密码，进入修改成功界面，修改成功界面有一提示按钮”Log in again“提示返回登录界面重新登录。否则，进入修改失败界面，修改失败界面有提示按钮”change again“点击返回修改密码界面重新修改。若点击提示按钮”give up change“则返回登录成功界面。



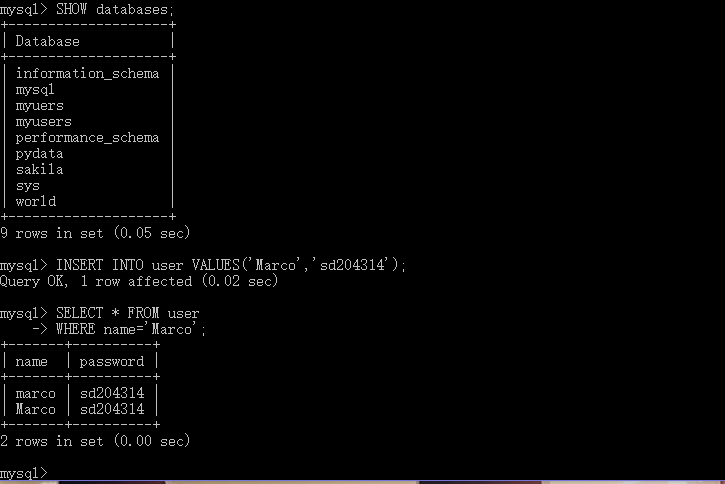
1. 注销：如下图所示界面，根据界面输入用户名，密码。若此时输入的用户名的密码与登录时的用户名，密码相同，则进入注销成功界面。注销成功界面有一提示按钮“login“点击则返回登录界面重新登录。否则，进入注销失败界面。注销失败界面有两个提示按钮。点击“Log out again”，返回注销界面重新注销。点击“give up Logout”，返回登录成功界面。
2. 查看天气信息：如下图所示界面，点击“table View”按钮，既可看到表格形式的天气报表。点击“chart View”，既可看到折线图形式的天气报表。点击“Go back”按钮，既可返回上一级，登录成功界面。



**4.项目测试**

**数据库测试：**

建立本地数据库并进行操作



**服务器测试**

测试程序如下：

import java.io.BufferedInputStream;

import java.io.BufferedReader;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.io.DataInputStream;

import java.io.RandomAccessFile;

import java.net.Socket;

import java.net.UnknownHostException;

public class Thelastsocket {

public static void main(String args[])throws Exception {

try {

Socket socket = new Socket("localhost",8001);

//获取输出流，向服务器端发送信息

OutputStream os=socket.getOutputStream();//字节输出流

PrintWriter pw=new PrintWriter(os);//将输出流包装为打印流

pw.write("我是Java服务器");

pw.flush();

InputStream is=socket.getInputStream();

BufferedReader in = new BufferedReader(new InputStreamReader(is));

String info=in.readLine();

System.out.println(info);

DataInputStream din=new DataInputStream(new BufferedInputStream(is));

File f=new File("D:/test/6.jpg");

RandomAccessFile fw=new RandomAccessFile(f,"rw");

byte[] b=new byte[1024];

int num=din.read(b);

while(num!=-1){

fw.write(b,0,num);

fw.skipBytes(num);

num=din.read(b);

}

pw.write("没毛病老铁");

pw.flush();

File f2=new File("D:/test/7.jpg");

RandomAccessFile fw2=new RandomAccessFile(f2,"rw");

while(num!=-1){

fw2.write(b,0,num);

fw2.skipBytes(num);

num=din.read(b);

}

info=in.readLine();

System.err.println(info);

System.out.println("asfd");

din.close();

is.close();

in.close();

socket.close();

} catch (UnknownHostException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

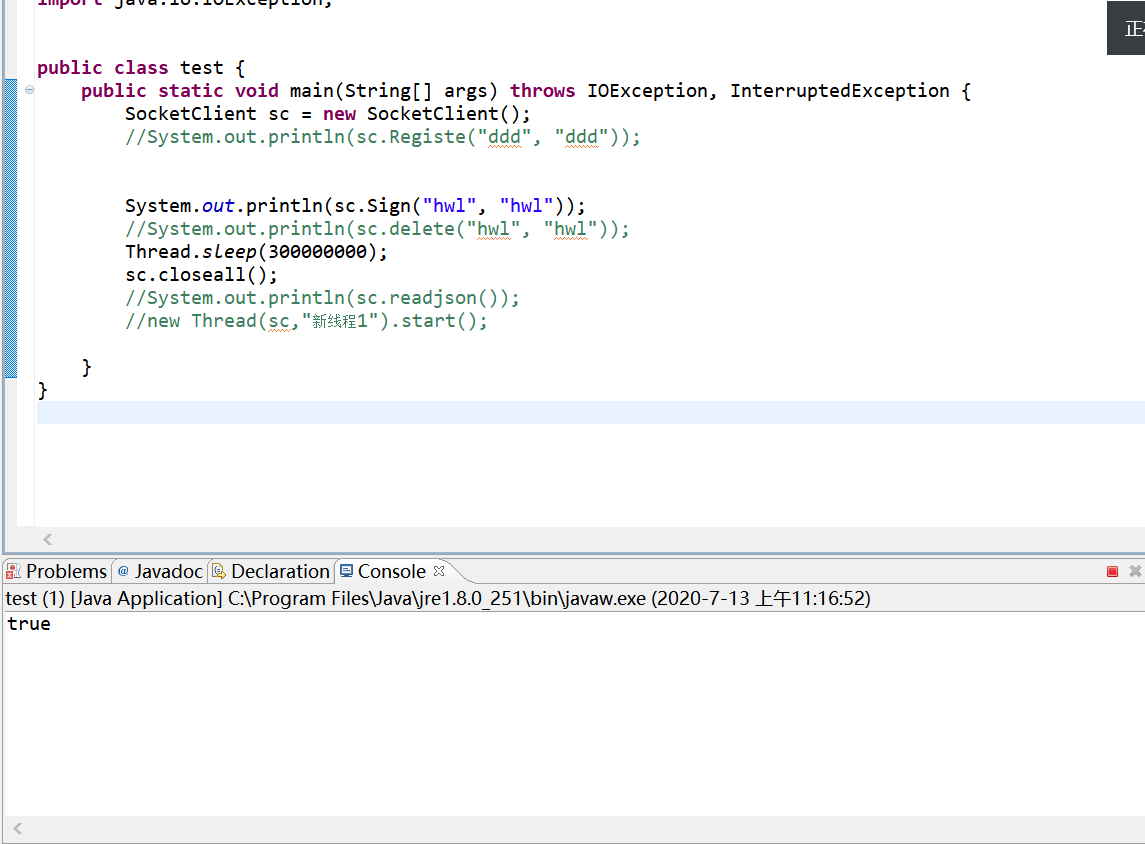
}

服务器和客户端交互主要问题是在于数据传输的流畅性，所以测试程序也主要是这方面的测试

Python端测试程序  
*#!/usr/bin/env python*import socket  
sock=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  
sock.bind((**'localhost'**,7556))  
sock.listen(5)  
while True:  
 connection,address = sock.accept()  
 print( **"client ip is "**)  
 print( address)  
 try:  
 connection.settimeout(5)  
 buf = connection.recv(1024)  
 if buf == **'1'**:  
 connection.send(**'welcome to python server!'**)  
 else:  
 connection.send(**'please go out!'**)  
 except socket.timeout:  
 print (**'time out'**)  
 connection.close()

测试截图：





**前端网页测试**

由于前端网页需要和后端进行数据交互。所以进行测试时只能在前端网页内置参数实现各网页的功能实现和网页跳转逻辑。起初为实现页面跳转，尝试使用继承HTTPservlet类重写doPost方法实现。并更改了项目部署文件web.xml的URL路径。但不知是jdk版本问题还是tomcat问题。最终没能通过此方法实现页面跳转。于是转用在jsp网页内导入后端Java类。只实现在网页之间交换参数。并调用Java类的方法实现与Python服务器后端的数据传输和指令返回。后期整合时，再和后端进行对接，调用函数对网页程序进行微调。

**5.项目总结**

这一次的小学期实训课程让我们收获了很多。这次实训课的很大一部分知识对我们小组的成员来说使完全陌生的，老师给出的资料也很有限，在绝大多数时候是靠我们自学完成。在查阅资料学习的时候我们也走了很多弯路，因为查到的资料可能对于我们的项目来说并不适用，而且资料本身的正确性也很难保证。网络上的资料五花八门，对于我们来说怎么正确寻找自己真正想要了解的，怎么辨别资料的正确性也是一种锻炼。所以一个人独自研究某个方面的时候经常出现在一个问题卡住，一两天没有结果。这样一路下来走了不少的弯路遇到了不少的坑，也很好的提高了我们的自学能力、资料查询的能力、知识的即时应用能力。

这一次实训也让我们明白了，在一定的时间内，领导一个小组合力快速高质量的完成一个项目并没有看上去那么简单。我们一开始就是因为轻视了这个问题，前期分工就没有能够快速适应节奏快速调整，在中期虽然进度勉强跟上，但是到了后期沟通整合的时候，这个严重的问题就暴露了出来，以至于最后整体进度完成度并不理想。这个经历和老师的话让我们明白了，在小组工作的时候，及时的沟通、对互相之间进度的掌握、合理的分工，这些都是非常重要的。相信这次的教训会让我们在以后的学习工作中都重视这一点。