一、课程设计的内容

课程设计题目：国内疫情实时大数据可视化

2020 年初，一场突如其来的新冠肺炎疫情肆虐武汉，并蔓延至中国大地，扰乱了全国人民的正常工作与生活，在党中央的领导下，国内疫情得到了有效控制，为全世界争取了时间，但由于许多国家的不作为，疫情在世界范围内全面散播开来，中国也很难独善其身，国内疫情稍有反复，但总体情势乐观，本站旨在监控国内疫情的情况，为持续抗疫奉献力量。

基本介绍：本项目基于 requests 第三方库获取腾讯提供的疫情实时数据，包括历史数据和详细到市的具体数据，为了进行实时更新，我将爬虫程序封装成了 exe 可执行文件，并在云服务器中定时执行，接着，我还将获取到的数据存入到 mysql 数据库中，通过 python 和第三方库 flask 进行数据处理和创建数据接口。在前端部分，我使用 echarts 可交互图表，用 ajax 调用接口获取数据，并将其渲染到图中，在第一排的第二个框中使用了 vue 框架，将数据直接渲染至页面。

开发技术：requests flask mysql echarts vue nginx

预览地址：http://121.5.177.147:5230/

二、功能描述

一、网络爬虫

作为一名大学生，我们能够获取的疫情信息主要来自于互联网，通过网络爬虫技术将需要的数据抓取到本地文档中，并进行一系列后续的操作。

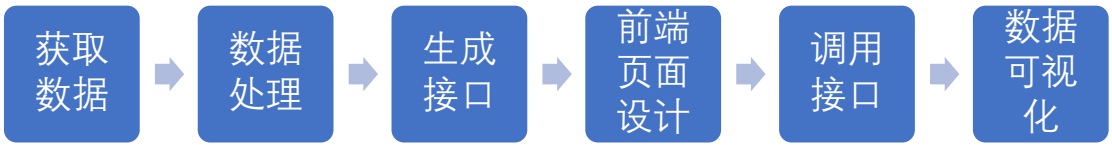
二、实时数据可视化

在数据可视化的过程中，我将网络爬虫获取到的数据进行一连串的数据分析，并将其封装成接口，供图表获取数据，最后以可交互图表的形式展现给用户，给用户以良好的使用体验。

三、及时预警

用户在访问页面查看数据的时候，可以清晰地查看到国内疫情的数据，在出现异常增长值的时候可以做出相应的措施，尽量减少出门次数、佩戴口罩等。

三、概要设计



① 在获取数据模块，项目采用python编写爬虫程序，运用第三方urllib库、requests库、bs4库抓取腾讯新闻里的疫情数据，存入本地mysql数据库中，并将其封装成exe可执行文件，定时执行任务，确保数据的实时性。

② 在数据处理模块，项目采用python的第三方库pandas和pymysql，从mysql数据中调取已经收集好的疫情数据，对其进行预处理和分析。

③ 在生成接口模块，项目采用python的第三方库flask进行接口搭建，将处理好的出局按图标类别分好，方便前端调用。

④ 在前端页面设计模块，项目在GitHub中找了一个模板，并对其进行前端三大件的修改，为后续内容提供完整的布局。

⑤ 在调用接口模块，项目使用原生js的ajax请求来调用接口获取数据。

⑥ 在数据可视化模块，项目使用基于js的开源可视化库Echarts，能够展现可视化图表，由于不了解DOM树的基本操作，项目还使用到了vue框架，直接渲染文档到页面。

四、详细设计

（一）获取数据

① 在发送request的时候需要指定headers，headers是解决requests请求反爬的方法之一，相当于我们进去这个网页的服务器本身，假装自己本身在爬取数据。

② 获取到的数据为json格式，需要用json库解析。

③ 连接数据库后要关闭，进行sql语句操作之后要commit才会存到本地的mysql表中。

④ 爬虫项目中需要使用try-except语句保证代码安全性。

```python
def get_conn():
    # 链接数据库
    conn=pymysql.connect(host='localhost',
                         user='root',
                         password='123456',
                         db='cov',
                         charset='utf8')
    cursor=conn.cursor()
    return conn,cursor

def close_conn(conn,cursor):
    # 关闭数据库
    if cursor:
        cursor.close()
    if conn:
        conn.close()
```

```python
def update_details(data):
    # 更新详细数据
    conn, cursor = None, None
    try:
        conn,cursor=get_conn()
        sql = "insert into details(update_time,province,city,confirm,confirm_add,heal,dead)values(%s,%s,
        sql_query="select %s=(select update_time from details order by id desc limit 1)"
        cursor.execute(sql_query,data[0][0])
        if not cursor.fetchone()[0]:
            print("{}开始更新数据".format(time.asctime()))
            for item in data:
                cursor.execute(sql,item)
            conn.commit()
            print("{}更新数据完毕".format(time.asctime()))
        else:
            print("{}已经是最新数据".format(time.asctime()))
    except Exception as e:
        print(e)
    finally:
        close_conn(conn,cursor)
```

（二）数据处理

① 数据处理的目的在于将数据库中的数据提取出来，进行一定的处理，中间用到了列表和字典的基本操作，最后目的在于封装成一个json格式的数据。

② 从数据库中调取数据时需要用到SQL语句，可以先在MYSQL WorkBench中确定SQL语句的正确性，再写入python处理中。

```python
# 解析数据
for i in data:
    confirm = i[0]
    suspect = i[1]
    heal = i[2]
    dead = i[3]
return jsonify({"confirm": confirm, "suspect": suspect, "heal": heal, "dead": dead})
```

（三）生成接口

① 在这个部分使用到了python的flask，可以便捷地构建api。

② 由于前后端分离可能存在的跨域问题，需要完善。

```python
@app.route("/api/l2")
def get_data_l2():
    date, confirm_add = [], []

    # 从数据库中读取数据
    sql = "SELECT ds,confirm_add FROM cov.history ORDER BY ds desc limit 14;"
    conn, cursor = get_conn()
    cursor.execute(sql)
    data = cursor.fetchall()
    close_conn(conn, cursor)

    # 解析数据
    for i in data:
        date.append(i[0].strftime("%m-%d"))
        confirm_add.append(i[1])

    return jsonify({"date": date, "confirm_add":confirm_add})
```

```python
app = Flask(__name__)


@app.after_request
def cors(environ):
    environ.headers['Access-Control-Allow-Origin'] = '*'
    return environ
```

（四）前端页面设计

① 运用可视化大屏模板，根据需求做出修改。

② 在右上角加入实时更新的时间，编写函数即可实现。

（五）调用接口

① 运用ajax请求flask生成的api，获取数据。

② 为保证数据的更新，需要编写定时函数，实时从api中获得数据。

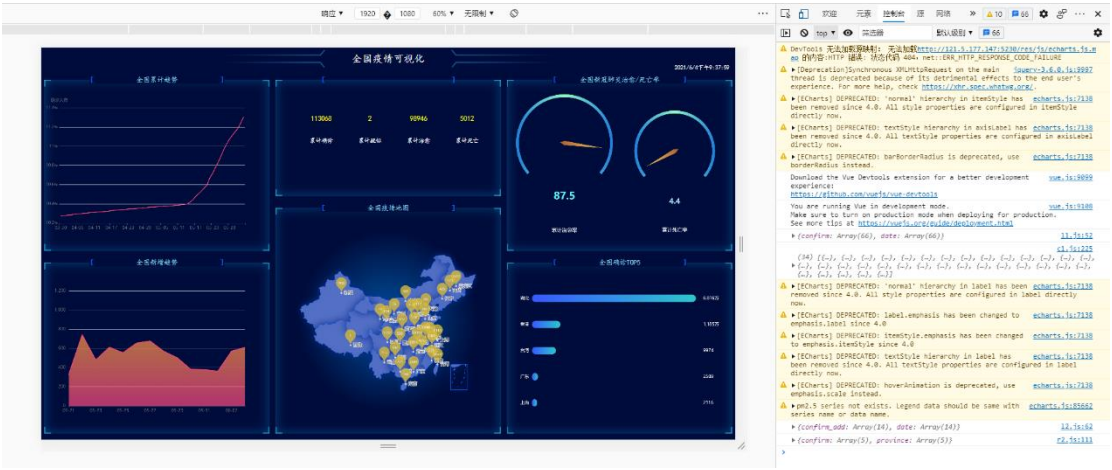
```
function get_data_l1() {
  $.ajax({
    url: "http://121.5.177.147:8090/api/l1",
    success: function (data) {
      console.log(data);
      option_1.xAxis.data = data.date.reverse();
      option_1.series[0].data = data.confirm.reverse();
      myChart_1.setOption(option_1, true);
      return get_data_l1
    }
  });
}

self.setInterval(get_data_l1(), 10000000);
```

（六）数据可视化

① 从echarts官网中复制模板，若有第三方js库则需要导入。

② 将ajax获取的数据放入模板当中，最后生成可交互的图表。

五、测试结果



六、总结

本次项目采用前后端分离，基于python爬虫、flask和echarts等技术完成疫情数据的可视化，在实战过程中，老师带领我们层层递进，先从获取数据开始，利用网络爬虫爬取疫情的历史和详细数据，接着运用pandas和简单的flask创建api，以供后续步骤使用，接着，完成前端页面框架的构建，接着倒入echarts图表，调用api，将数据放入echarts图表中，此外，我还在腾讯云服务器中采用nginx部署项目，使其能够在公网上访问，进而完成整个项目。

在本次项目中，我完整地完成了一次可视化BI大屏的开发，对大数据的处理有了更加深刻的认识，前后端分离的方法步骤也给了我很大的启发，还有云服务器的部署，在过程中我遇到了许多问题，例如云服务器外网访问、数据处理、等问题，在老师的指导和网络的帮助下，我将问题逐个解决。

项目仍存在部分不足尚未改进，例如，在获取疫情总数据的部分未编写定时函数，若长时间停留页面会导致资源浪费等问题，在定时爬虫部分，没有深入观察腾讯新闻疫情数据更新的时间，在下午两点多执行数据，有时会导致最新数据有两天的时间差，在选择图标部分，没有深入研究数学模型，只做了表意非常简单的图表，针对这些不足，项目后期我会进一步改善。