目 录

1. 需求分析

随着国家政策的推广和社会环境的需要，垃圾分类越来越多的出现在人们大的视野之中，但随之而来的是在垃圾分类过程中垃圾种类的繁多和各个地方对垃圾分类不同的类别的不同，对普通人的生活造成了不小的烦恼，错误的垃圾分类也会对周围环境造成非常大的污染。并且对与一些垃圾来讲，其实换一种说法对它们称呼为旧物更加合适，它们有非常多的潜在的利用价值值得人们去挖掘，比如对于一些废纸箱和废纸盒经过一些加工和美化完全可以制成一些笔盒等其他日常生活中可以使用的小物件。所以我和我们的小组成员试想能不能使用互联网和人工智能的技术去结合起来，能够辅助甚至是帮助我们去实现这些垃圾分类的任务，并且提供一个平台让每个人可以去分享和学习各在日常生活中对于垃圾重复利用的奇思妙想和垃圾分类相关的知识。所以创建一个网站—垃圾分类知识社区并在其中有机的结合图像分类的人工智能技术俨然成为解决这一任务最好的实现方法。

SSM 框架集由 Spring、MyBatis 两个开源框架整合而成（SpringMVC 是 Spring 中的部分内容），常作为数据源较简单的 web 项目的框架。根据其低耦合的特性非常适合去应用在本次的网站开发之中，并且其在本次的网站项目中主要作用为采用异步通信的模式和 JS 中的 Ajax 模块一起去通过 json 文件的传递去渲染前端的所有页面。

Flask 是一个轻量级的可定制框架，使用 Python 语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合 MVC 模式进行开发，另外，Flask 还有很强的定制性，用户可以根据自己的需求来添加相应的功能，在保持核心功能简单的同时实现功能的丰富与扩展，其强大的插件库可以让用户实现个性化的网站定制，开发出功能强大的网站。由于其使用 python 语言编写所以非常适合和人工智能模型进行有机的结合实现连调，再由其和 SSM 编写的 Tomcat 服务器之间实现通信，去实现网站调用人工智能模型进行垃圾图像分类预测等相关的功能。

图像分类，根据各自在图像信息中所反映的不同特征，把不同类别的目标区分开来的图像处理方法。它利用计算机对图像进行定量分析，把图像或图像中的

每个像元或区域划归为若干个类别中的某一种，以代替人的视觉判读。而采用深度学习中的卷积神经网络则是图像分类最好的实现方法之一。在卷积神经网络的众多典型模型之中，MobileNetV3 模型以模型参数少，训练速度快，可扩展性强，还可以进行一些嵌入式开发，比较符合垃圾分类这一应用场景。

本文就着重讨论该如何去运用 SSM 框架集和 Flask 实现在垃圾分类的应用场景下传统网络社区和人工智能技术实现有机的结合。对于如何应对此工程中的数据库建立、网站各个应用模块的实现等难点进行实验和研究。

2. 概要设计

2.1 项目功能分析

首先，提供一个可以分享生活中垃圾分类和回收利用经验的平台大致需要一下几点功能：（1）使用户可以上传和观看关于垃圾分类知识或者垃圾如何回收利用的视频（2）使用户可以上传和浏览关于垃圾分类知识或者垃圾如何回收利用的帖子（3）用户上传垃圾图像然后对垃圾进行智能分类（4）在地图上标出附近垃圾桶的位置供用户使用。

其次，对于网站本身和用户本身使用而言需要的功能有：（1）用户的登录注册功能（2）用户对自我信息的修改和完善（3）发帖子（4）上传视频（5）对帖子或视频进行评论（6）对帖子和视频进行点赞和收藏。

最后，考虑为了垃圾分类识别此功能和垃圾分类社区之间的关联程度更高，可以设计一个可视化统计图表对用户的不知道如何分类的垃圾进行统计和展示，网站管理员就可以根据此统计数据去统一公布一些公告，让网站的所有用户都可以了解这些大多数人不知道但又很重要的垃圾分类知识。

2.2 数据库设计

根据以上的功能可以分析出一共需要实体和表名分别为：user（用户）、trash(垃圾)、trashcan(垃圾桶)、video(视频)、posts(帖子)、notice(公告)、headpicture(头像)、comment(评论)、goodv(视频的点赞)、collectionv(视频的收藏)、goodp(帖子的点赞)、collectionp(帖子的收藏)十二张表。数据库的

ER 图如下：

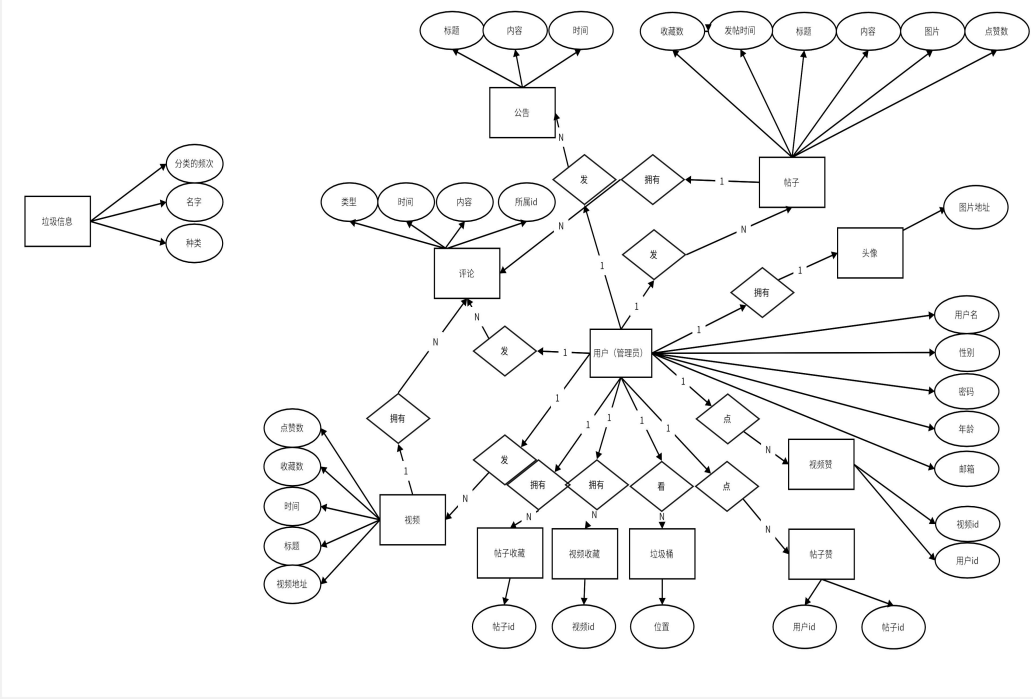


图-1 数据库 ER 图

user(用户)的表结构：

```
CREATE TABLE `user` (
`id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID',
`username` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '用户名',
`password` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '密码',
`sex` varchar(1) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '性别',
`age` int(11) DEFAULT NULL COMMENT '年龄',
`email` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '邮箱',
`type` int(1) NOT NULL COMMENT '是否管理员 1 是 0 否',
PRIMARY KEY (`id`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

trash(垃圾)的表结构：

```
CREATE TABLE `trash` (
  `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '垃圾名字',
  `type` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '类型',
  `frequency` int(20) NOT NULL COMMENT '频率',
  PRIMARY KEY (`name`,`type`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

trashcan(垃圾桶)的表结构：

```
CREATE TABLE `trashcan` (
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  `x` int(11) NOT NULL COMMENT 'x 坐标',
  `y` int(11) NOT NULL COMMENT 'y 坐标',
  PRIMARY KEY (`userid`,`x`,`y`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

video(视频)的表结构：

```
CREATE TABLE `video` (
 `fileId` int(11) NOT NULL AUTO_INCREMENT,
  `cover` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `titleOrig` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `titleAlter` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `numgood` int(11) DEFAULT NULL,
  `numcollection` int(11) DEFAULT NULL,
  `size` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `type` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `path` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
  `uploadtime` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,

  `userid` int(11) NOT NULL,
  PRIMARY KEY (`fileId`,`userid`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

posts(帖子)的表结构：

```
CREATE TABLE `posts` (
`id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `time` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '时间',
  `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '标题',
  `content` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '内容',
  `numgood` int(11) DEFAULT NULL COMMENT '点赞数',
  `numcollection` int(11) DEFAULT NULL COMMENT '收藏数',
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  PRIMARY KEY (`id`,`userid`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

notice(公告)的表结构：

```
CREATE TABLE `notice` (
  `time` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '时间',
  `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '标题',
  `content` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL COMMENT '内容',
  PRIMARY KEY (`time`,`title`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

headpicture(头像)的表结构：

```
CREATE TABLE `headpicture` (
`userid` int(11) NOT NULL COMMENT '用户 ID',
  `padress` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '头像地址',
  PRIMARY KEY (`userid`,`padress`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

comment(评论)的表结构：

```
CREATE TABLE `comment` (
  `time` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '时间',
```

```
  `content` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '内容',
  `type` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL COMMENT '类型',
  `itid` int(11) NOT NULL COMMENT '视频或帖子的 ID',
  `userid` int(11) NOT NULL COMMENT '用户 id',
  PRIMARY KEY (`itid`,`userid`,`type`,`time`,`content`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

goodv(视频的点赞)的表结构：

```
CREATE TABLE `goodv` (
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  `videoid` int(11) NOT NULL COMMENT '视频 ID',
  PRIMARY KEY (`userid`,`videoid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

collectionv(视频的收藏)的表结构：

```
CREATE TABLE `collectionv` (
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  `videoid` int(11) NOT NULL COMMENT '视频 ID',
  PRIMARY KEY (`userid`,`videoid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

goodp(帖子的点赞)的表结构：

```
CREATE TABLE `goodp` (
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  `postsid` int(11) NOT NULL COMMENT '帖子 ID',
  PRIMARY KEY (`userid`,`postsid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;IC;
```

collectionp(帖子的收藏)的表结构：

```
CREATE TABLE `collectionp` (
  `userid` int(11) NOT NULL COMMENT '用户 ID',
  `postsid` int(11) NOT NULL COMMENT '帖子 ID',
  PRIMARY KEY (`userid`,`postsid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin ROW_FORMAT=DYNAMIC;
```

2.3 SSM 项目框架设计

首先，创建一个 maven 项目导入项目所需的依赖，调整好开发 SSM 的项目结构，在项目文件夹相应位置下分别建立 controller、mapper、domain、service、util 五个包,其中 mapper 包下的 mapper 接口类调用相应的 xml 文件通过 MyBatis 对数据库进行增删改查，service 包内主要有 service 接口和接口实现类，在实现类中去通过 IOC 注入 mapper 类后去调用 mapper 中的方法。其次在 domain 下建立我们这次项目所需要的十二个类，其中类的属性应该和数据库中对应表的属性保持一致。最后，为了保持项目的规范性，此项目的结构如下：

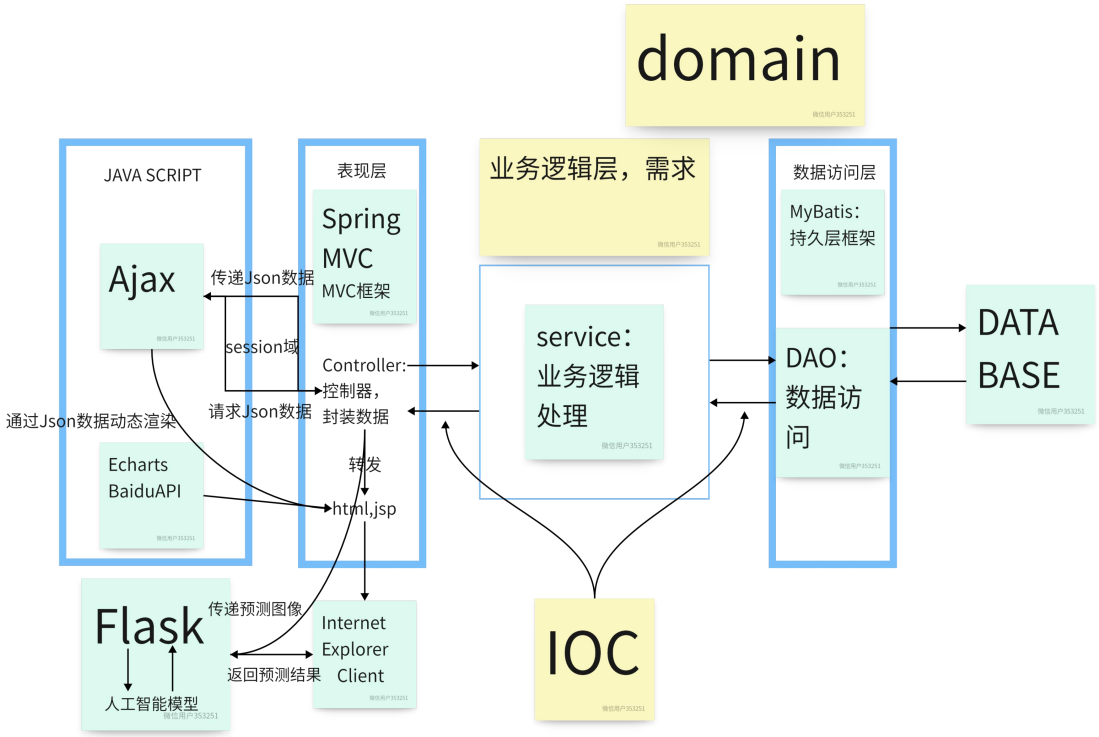


图-2 各个模块之间的调用关系

3. 开发工具和编程语言

操作系统：windows11（64 位）

硬件：处理器 AMDRyzen9 6900HX with Radeon Graphics 标准频率 3.30GHz

机带 RAM：32.0 GB

独立显卡型号：NVDIA GEFORCE RTX 3070TI

编程语言：JAVA 64-bit JDK 版本 1.8、python 64-bit 版本 3.8.8、

开发工具：IntelliJ IDEA2022.2.3、Pycharm2022.2.2

4. 详细设计

4.1 视频的上传和删除

对于解决视频的上传问题我们要考虑我们要存入那些和视频相关的信息进入数据库，并且考虑如何根据数据库中的信息能够将其渲染到前端的页面上。（数据库中视频的表结构如上）所以可以非常简单地得出我们需要把视频的地址在视频上传的同时传入数据库中，等在前端需要渲染这个视频时，直接把视频的地址传入即可。理清楚思路后我们首先需要编写在 domain 中视频类并重写 seter、geter 方法和 tostring 方法。在传入文件过程中最需要注意的几点是，前端的 form 表单一定要加入 enctype="multipart/form-data" 属性，并且采用 POST 方式去访问相应的网址，并且要在 pom.xml 文件中引入相关依赖。

其次,添加视频时我们对其所有的属性都进行数据库中的插入操作，并通过其主键 id 对视频进行删除，以下为编写 mapper 接口类和其相关的 mapper.xml 实现对数据库的增删改查：

FileMapper.xml

```
1.  <?xml version="1.0" encoding="UTF-8" ?>
2.  <!DOCTYPE mapper
3.   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4.   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5.  <mapper namespace="com.jzlw.itc.mapper.Filemapper">
6.    <resultMap id="videomap" type="com.jzlw.itc.domain.FileEntity">
7.      <result column="fileId" property="fileId"/>
8.      <result column="userid" property="userid"/>
9.      <association property="user" javaType="com.jzlw.itc.domain.User"
10.                  select="com.jzlw.itc.mapper.Usermapper.selectauser" column="
    userid">
11.     </association>
12.   </resultMap>
13.   <update id="updatecover">
14.     update video set title=#{title},cover=#{cover}
15.     where titleAlter = #{titleAlter}
16.   </update>
17.   <select id="findByid" parameterType="java.lang.Long" resultMap="videomap">

18.     select * from video where fileId=#{id}
19.   </select>
```

```
  <insert id="saveFile" parameterType="com.jzlw.itc.domain.FileEntity">
    insert into video
    (titleOrig,titleAlter,`size`,`type`,`path`,`uploadTime`,`userid`,`cover`,numgood,numcollection)
    VALUES
    (#{titleOrig}, #{titleAlter}, #{size}, #{type}, #{path},#{uploadTime},#{userid},#{cover},0,0)
  </insert>
  <select id="selectvideo" resultMap="videomap">
    select * from video
  </select>
  <delete id="deletevideo">
    delete from video where userid = #{userid} and fileId =#{id}
  </delete>
</mapper>
```

FileMapper.java

```
public interface Filemapper {
//保存视频
    void saveFile(FileEntity entity);
    //根据视频的 id 查询
    FileEntity findByid(long id);
    //更新视频的封面
    void updatecover(@Param("title") String title,@Param("cover") String cover,@Param("titleAlter") String titleAlter);
    //删除视频
    void deletevideo(@Param("id") long id,@Param("userid") Integer userid);
}
```

然后实现规范的流程，编写 service 接口类和实现类去对 mapper 类进行调用，然后 controller 类再对注入 service 类进行调用最后实现对数据库增删查改的目的。

IFileService

```
public interface Filemapper {
   //保存视频
    void saveFile(FileEntity entity);
    //根据视频的 id 查询
    FileEntity findByid(long id);
    //更新视频的封面
    void updatecover(@Param("title") String title,@Param("cover") String cover,@Param("titleAlter") String titleAlter);
```

```
    //返回一些视频供主页面展示
    List<FileEntity> selectvideo();
    //删除视频
    void deletevideo(@Param("id") long id,@Param("userid") Integer userid);
}
```

FileServiceimpl

```
@Service
public class FileServiceimpl implements IFileService {
    @Autowired
    private Filemapper filemapper;
    public void saveFile(FileEntity entity) {
        filemapper.saveFile(entity);
    }
    public FileEntity findByid(long id) {
        return filemapper.findByid(id);
    }
    @Override
    public void updatecover(String title,String cover,String titleAlter) {
        filemapper.updatecover(title,cover, titleAlter);
    }
    @Override
    public List<FileEntity> selectvideo() {
        return filemapper.selectvideo();
    }
    public void deletevideo(long id, Integer userid) {
        filemapper.deletevideo(id,userid);
    }
}
```

4.2 视频的展示

在本次项目中视频的展示是一个比较困难的一点。我们想要实现对每一个用户发布的视频进行显示，首先需要再前端界面展示此视频的封面，并且使用 a 标签配合 Ajax 动态的将此视频的 id 作为参数渲染到 a 标签 href 属性里，实现点击此封面就会访问相应的 controller，此网址下的相应的 controller 将接收到的视频 id 存放到 seesion 域中并且转发出将要展示视频的 html，然后展示视

频的 html 再通过 Ajax 去访问相应网址下的 controller，此 controller 去 sessionz 域中去取出此视频的 id 并且通过数据库查询操作将此视频的所有信息通过传递 json 的形式返回给将要展视频的页面，页面再根据此 json 文件的信息去动态渲染该页面。源代码和流程如下：

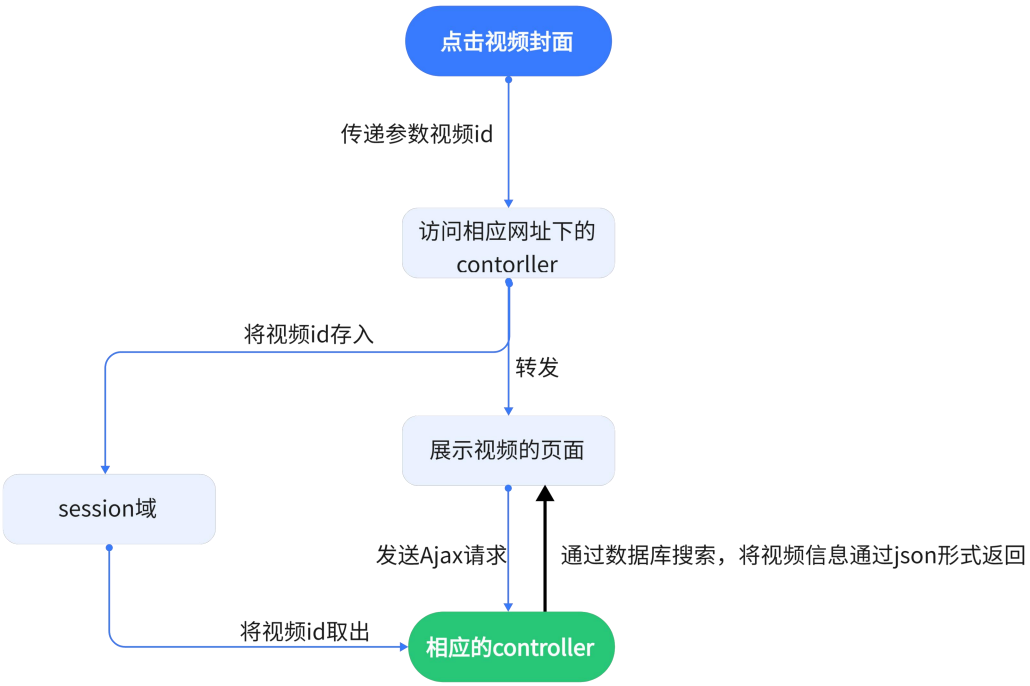


图-3 视频展示在前端页面流程

Videocontroller;

```
@Controller
@SessionAttributes({"id","fileId","uuid"})
public class Videocontroller {
    @Autowired
    private FileServiceimpl service;
    @Autowired
    private UserServiceimpl userservice;
    @Autowired
    private GoodvServiceimpl gvservice;
    @Autowired
    private CollectionvServiceimpl cvservice;
    @Autowired
    private CommentServiceimpl commentservice;

    @RequestMapping(value = "/uploadVideo")
```

```
16.     public String upload(@RequestParam(value = "file", required = false) Mult
    ipartFile multipartFile,
17.                          HttpServletRequest request, @ModelAttribute("id")Int
    eger id,Model model) {
18.         FileEntity entity = new FileEntity();
19.         String logoPathDir = request.getParameter("shipin");
20.         FileUploadTool fileUploadTool = new FileUploadTool();
21.         try {
22.             entity = fileUploadTool.createFile(logoPathDir, multipartFile, re
    quest,id);
23.             if (entity != null) {
24.                 service.saveFile(entity);
25.                 model.addAttribute("uuid",entity.getTitleAlter());
26.                 System.out.println("上传成功");
27.             } else {
28.                 System.out.println("上传失败");
29.             }
30.         } catch (Exception e) {
31.             e.printStackTrace();
32.         }
33.         return "victory";
34.     }
35.     @GetMapping(value = "/addvideo")
36.     public String addvideo(){
37.         return "forward:/WEB-INF/views/my/addvideo.html";
38.     }
39.     @PostMapping(value = "/addcover",params = {"title"})
40.     public String oneFileUpload(
41.             @RequestParam("file") CommonsMultipartFile file,
42.             HttpServletRequest request, ModelMap model, @ModelAttribute("uuid
    ") String uuid,String title) throws IOException {
43.         Timestamp timestamp = new Timestamp(System.currentTimeMillis());
44.         // 获得原始文件名
45.         String fileName = file.getOriginalFilename();
46.         System.out.println("原始文件名:" + fileName);
47.         // 新文件名
48.         String newFileName = uuid + ".jpg";
49.         String path = "C:/Users/jing/Desktop/java-project/itc/target/itc-1.0-
    SNAPSHOT/static/video/cover";
50.         File f = new File(path);
51.         if (!f.exists())
52.             f.mkdirs();
53.         if (!file.isEmpty()) {
```

```
            try {
                FileOutputStream fos = new FileOutputStream(path + '/' + newFileName);
                InputStream in = file.getInputStream();
                int b = 0;
                while ((b = in.read()) != -1) {
                    fos.write(b);
                }
                fos.close();
                in.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        path = "/static/video/cover/" + newFileName;
        System.out.println("上传图片到:" + path + '/' + newFileName);
        service.updatecover(title,path,uuid);
        return "forward:/WEB-INF/views/my/addvideo.html";
    }
    //用于主页面的显示返回一些视频
    @PostMapping("/displayvideo")
    @ResponseBody
    public List<FileEntity> displayvideo(){
        return service.selectvideo();
    }
    //接收参数，并且把视频 id 保留到 session 中,并且渲染播放页面
    @GetMapping(value = "/loadid",params = {"fileId"})
    public String loadid(String fileId, Model model){
        model.addAttribute("fileId",fileId);
        return "forward:/WEB-INF/views/playvideo.html";
    }
    //从 session 中得到刚才的 id 并且返回一个视频 json
    @PostMapping(value = "/getvideo")
    @ResponseBody
    public FileEntity getvideo(@ModelAttribute("fileId") String fileId){
        FileEntity video = service.findByid(Long.parseLong(fileId));
        return video;
    }
    //删除帖子,从网页段前来的 id 是视频的 id
    @PostMapping(path = "/deletevideo",params = {"id","filename"})
    public String deletevideo(Long id,String filename,@ModelAttribute("id")Integer userid){
        service.deletevideo(id,userid);
```

```
95.         //删除和帖子相关的所有的收藏和点赞
96.         gvservice.degoodvforv(id.intValue());
97.         cvservice.decollvforv(id.intValue());
98.         //删除帖子所有的评论
99.         commentservice.deleteacommmentfordeleted("video",id.intValue());
100.         filename = "C:/Users/jing/Desktop/itc/target/itc-1.0-SNAPSHOT/static
    " + filename;
101.         File file = new File(filename);// 根据指定的文件名创建 File 对象
102.         if (!file.exists()) { // 要删除的文件不存在
103.             System.out.println("文件" + filename + "不存在，删除失败！");
104.         } else { // 要删除的文件存在
105.             if (file.isFile()) { // 如果目标文件是文件，判断是文件
106.                 file.delete();
107.                 System.out.println("已经删除");//删除文件
108.             }
109.         }
110.         return "forward:/WEB-INF/views/my/myvideo.html";
111.     }
112. }
```

展示视频的前端 Ajax:

```
1.  <script>
2.  var posts = ""
3.      $.ajax({
4.          url:"/getposts",
5.          type:"post",
6.          dataType:"json",
7.          success:function (data){
8.              posts = data
9.              var thehtml = "<h1 class='displayh1'>标
    题:  " + posts.title +"</h1>"
10.
11.             str = "<div class='sty1'>"
12.                     +"<p>内容:  "+ posts.content +"</p>"
13.                 +"</div>"
14.
15.             thehtmlthehtml = thehtml + str
16.             $('.display').html(thehtml)
17.         },
18.         error:function (xhr,status){
19.             console.log("错误了")
```

```
            console.log(status)
        }
    })
    var user = ""
    $.ajax({
        url:"/transmitinfo",
        type:"post",
        dataType:"json",
        success:function (data){
            user = data
            var result = 0
            var thehtml = "<div class='sty2'>"
                                +"<h1 class='sty2h1'>帖子的点赞和收藏</h1>"
                                +"<br><h2>用户简介:</h2><br>"
                                +"<img src= '" + posts.user.headpicture.padress + "'>"
                                +"<h4>发布者:"+posts.user.username+"</h4>"
                                +"<br><h4>发布时间:"+ posts.time+"</h4><br><br>"
                                +"<br><h3>点赞数:"+posts.numgood+"</h3>"
                                +"<br><h3>收藏数:"+posts.numcollection+"</h3><br>"
                            +"</div>"
            try {
                var str = ""
                user.goodps.forEach((item,index) => {
                    if(item.postsid === posts.id){
                        str = "<a title='取消点赞' href='/deletegoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>    "
                        thehtmlthehtml = thehtml + str
                        result++
                        throw new Error('找到了')
                    }
                })
            }catch (e) {
                if (e.message !== "找到了") throw e
            }
            //如果本来就没有点赞直接显示可以点赞
            if (user.goodps.length === 0){
                str = "<a title='点赞' href='/addgoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-i
```

```
con\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>
    "
                thehtmlthehtml = thehtml + str
            }
            //没有进入 if 就可以点赞
            else if (result === 0){
                str = "<a title='点赞
' href='/addgoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-i
con\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>
    "
                thehtmlthehtml = thehtml + str
            }
            //收藏
            result = 0
            try {
                var str = ""
                user.collectionps.forEach((item,index) => {
                    if(item.postsid === posts.id){
                        str = "<a title='取消收藏
' href='/deletecollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=
\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">
  </i></a>"
                        thehtmlthehtml = thehtml + str
                        result++
                        throw new Error('找到了')
                    }
                })
            }catch (e) {
                if (e.message !== "找到了") throw e
            }
            //如果本来就没有点赞直接显示可以点赞
            if (user.collectionps.length === 0){
                str = "<a title='收藏
' href='/addcollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"l
ayui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </
i></a>"
                thehtmlthehtml = thehtml + str
            }
            //没有进入 if 就可以点赞
            else if (result === 0){
                str = "<a title='收藏
' href='/addcollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"l
```

```
ayui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>"
                thehtmlthehtml = thehtml + str
            }
            $('.gandc').html(thehtml)
            //渲染评论框
            $("#userid").attr('value',user.id)
            $("#itid").attr('value',posts.id)
        },

        error:function (xhr,status){
            console.log("错误了")
            console.log(status)
        }
    })
    $.ajax({
         url:"/getpostscomment",
         type:"post",
         dataType:"json",
         success:function (data){
             var comment = data
             var thehtml = "<h1>" + "评论区" +"</h1>"
             comment.forEach((item,index) => {
                 str = "<div class='star2'>"
                       +"<span style='color: #243b65; font-family: 等线'>用户" + item.user.username + "评论:</span><hr>"
                       +"<br><h3 style='color: #243b65; font-family: 等线'>" +item.content + "</h3>"
                       +"</div>"
                thehtmlthehtml = thehtml + str
             })
             $('.comment').html(thehtml)
         },
         error:function (xhr,status){
             console.log("错误了")
             console.log(status)
         }
    })
</script>
```

4.3 帖子的上传和删除

帖子的上传删除除了完全是对数据库中的数据进行操作其他的和视频的上传和删除同理，在此就不再赘述。源代码如下：

Postsmapper:

```
@Repository
public interface Postsmapper {
    //写帖子
    void insertaposts(Posts posts);
    //删除帖子
    void deleteaposts(@Param("id")Integer id,@Param("userid")Integer userid);

    //查询一些帖子用于主页展示
    List<Posts> selectposts();
    //通过帖子的 id 查询
    Posts findByid(Integer id);
```

Postsmapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.jzlw.itc.mapper.Postsmapper">
    <resultMap id="postsmap" type="com.jzlw.itc.domain.Posts">
        <id column="id" property="id"/>
        <result column="userid" property="userid"/>
        <association property="user" javaType="com.jzlw.itc.domain.User"
                     select="com.jzlw.itc.mapper.Usermapper.selectauser" colu
mn="userid">
        </association>
    </resultMap>
    <insert id="insertaposts">
        insert into posts
            (`time`,title,content,userid,numgood,numcollection)
        values
            (#{time},#{title},#{content},#{userid},0,0)
    </insert>
    <delete id="deleteaposts">
        delete from posts where `id` = #{id} and userid = #{userid}
    </delete>
    <select id="selectposts" resultMap="postsmap">
```

```
        select * from posts
    </select>
    <select id="findByid" resultMap="postsmap">
        select * from posts where id = #{id}
</mapper>
```

IPostsService:

```
public interface IPostsService {
     //写帖子
    void insertaposts(Posts posts);
    //删除帖子
    void deleteaposts(Integer id,Integer userid);
    List<Posts> selectposts();
    Posts findByid(Integer id);
}
```

PostsServiceimpl:

```
@Service
public class PostsServiceimpl implements IPostsService {
    @Autowired
    private Postsmapper mapper;
    public void insertaposts(Posts posts) {
        mapper.insertaposts(posts);
    }
    public void deleteaposts(Integer id, Integer userid) {
        mapper.deleteaposts(id,userid);
    }
    public List<Posts> selectposts() {
        return mapper.selectposts();
    }
    public Posts findByid(Integer id) {
        return mapper.findByid(id);
    }
```

4.4 帖子的展示

帖子的展示除了完全是对数据库中的数据进行操作其他的和视频的上传和删除同理，在此就不再赘述。源代码如下：

Postscontroller:

```
1.  //写入帖子
2.  @GetMapping("/writeposts")
3.      public String writeposts(){
4.          return "forward:/WEB-INF/views/my/writeposts.html";
5.      }
6.      @PostMapping(path = "/writeposts",params = {"title","content"})
7.      public String writeposts(String title,String content,@ModelAttribute("id")
    Integer userid){
8.          Timestamp timestamp = new Timestamp(System.currentTimeMillis());
9.          Posts posts = new Posts(timestamp.toString(),title,content,userid);
10.         service.insertaposts(posts);
11.         return "forward:/WEB-INF/views/my/myposts.html";
12.     }
13.     //删除帖子,从网页段前来的 id 是帖子的 id
14.     @PostMapping(path = "/deleteposts",params = {"id"})
15.     public String deleteposts(Integer id,@ModelAttribute("id")Integer userid)
    {
16.         service.deleteaposts(id,userid);
17.         //删除和帖子相关的所有的收藏和点赞
18.         gpservice.degoodpforp(id);
19.         cpservice.decollpforp(id);
20.         //删除帖子所有的评论
21.         commentservice.deleteacommmentfordeleted("posts",id);
22.         return "forward:/WEB-INF/views/my/myposts.html";
23.     }
24.     //用于主页展示一些帖子
25.     @PostMapping("/displayposts")
26.     @ResponseBody
27.     public List<Posts> displayvideo(){
28.         return service.selectposts();
29.     }
30.     //接收参数，并且把帖子 id 保留到 session 中,并且渲染播放页面
31.     @GetMapping(value = "/loadpostsid",params = {"postsid"})
32.     public String loadid(String postsid, Model model){
33.         model.addAttribute("postsid",postsid);
34.         return "forward:/WEB-INF/views/displayposts.html";
35.     }
36.     //从 session 中得到刚才的 id 并且返回一个视频 json
37.     @PostMapping(value = "/getposts")
38.     @ResponseBody
39.     public Posts getposts(@ModelAttribute("postsid") String postsid){
40.         Posts posts = service.findByid(Integer.parseInt(postsid));
41.         return posts;}
```

展示帖子的前端 Ajax：

```
<script>
     var posts = ""
    $.ajax({
        url:"/getposts",
        type:"post",
        dataType:"json",
        success:function (data){
            posts = data
            var thehtml = "<h1 class='displayh1'>标题:  " + posts.title +"</h1>"

            str = "<div class='sty1'>"
                      +"<p>内容:  "+ posts.content +"</p>"
                  +"</div>"

            thehtml = thehtml + str
            $('.display').html(thehtml)
        },
        error:function (xhr,status){
            console.log("错误了")
            console.log(status)
        }
    })
    var user = ""
    $.ajax({
        url:"/transmitinfo",
        type:"post",
        dataType:"json",
        success:function (data){
            user = data
            var result = 0
            var thehtml = "<div class='sty2'>"
                                +"<h1 class='sty2h1'>帖子的点赞和收藏</h1>"
                                +"<br><h2>用户简介:</h2><br>"
                                +"<img src= '" + posts.user.headpicture.padress + "'>"
                                +"<h4>发布者:"+posts.user.username+"</h4>"
                                +"<br><h4>发布时间:"+ posts.time+"</h4><br><br>"
                                +"<br><h3>点赞数:"+posts.numgood+"</h3>"
```

```
                                +"<br><h3>收藏数:"+posts.numcollection+"</h3><br>"
                                +"</div>"
            try {
                var str = ""
                user.goodps.forEach((item,index) => {
                    if(item.postsid === posts.id){
                        str = "<a title='取消点赞' href='/deletegoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>    "
                        thehtml = thehtml + str
                        result++
                        throw new Error('找到了')
                    }
                })
            }catch (e) {
                if (e.message !== "找到了") throw e
            }
            //如果本来就没有点赞直接显示可以点赞
            if (user.goodps.length === 0){
                str = "<a title='点赞' href='/addgoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>    "
                thehtml = thehtml + str
            }
            //没有进入 if 就可以点赞
            else if (result === 0){
                str = "<a title='点赞' href='/addgoodp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </i></a>    "
                thehtml = thehtml + str
            }
            //收藏
            result = 0
            try {
                var str = ""
                user.collectionps.forEach((item,index) => {
                    if(item.postsid === posts.id){
                        str = "<a title='取消收藏' href='/deletecollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=
```

```
\"layui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">
  </i></a>"
                        thehtml = thehtml + str
                        result++
                        throw new Error('找到了')
                    }
                })
            }catch (e) {
                if (e.message !== "找到了") throw e
            }
            //如果本来就没有点赞直接显示可以点赞
            if (user.collectionps.length === 0){
                str = "<a title='收藏
' href='/addcollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"l
ayui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </
i></a>"
                thehtml = thehtml + str
            }
            //没有进入 if 就可以点赞
            else if (result === 0){
                str = "<a title='收藏
' href='/addcollectionp?userid="+user.id+"&postsid="+posts.id+"'><i class=\"l
ayui-icon\" style=\"font-size: 30px;display: inline; color: #df0707a8;\">  </
i></a>"
                thehtml = thehtml + str
            }
            $('.gandc').html(thehtml)
            //渲染评论框
            $("#userid").attr('value',user.id)
            $("#itid").attr('value',posts.id)
        },

        error:function (xhr,status){
            console.log("错误了")
            console.log(status)
        }
    })
    $.ajax({
         url:"/getpostscomment",
         type:"post",
         dataType:"json",
         success:function (data){
             var comment = data
```

```
            var thehtml = "<h1>" + "评论区" +"</h1>"
            comment.forEach((item,index) => {
                str = "<div class='star2'>"
                      +"<span style='color: #243b65; font-family: 等线'>用户
" + item.user.username + "评论:</span><hr>"
                      +"<br><h3 style='color: #243b65; font-family: 等线
'>" +item.content + "</h3>"
                      +"</div>"
                thehtml = thehtml + str
            })
            $('.comment').html(thehtml)
        },
        error:function (xhr,status){
            console.log("错误了")
            console.log(status)
        }
    })
</script>
```

4.5 Flask 的搭建

对 Flask 服务器的基础设置如下：

```
import pymysql
from flask import Flask, request, render_template, abort
from flask_cors import CORS
from flask_sqlalchemy import SQLAlchemy
import inference
app = Flask(__name__,template_folder='../templates',static_folder='../static')

CORS(app, resources=r'/*')  # 注册 CORS, "/*" 允许访问所有 api
#解决跨域
CORS(app, supports_credentials=True)
#配置数据库地址
app.config['SQLALCHEMY_DATABASE_URI'] ='mysql://root:hbn@127.0.0.1:3306/itc'
#设置第二个数据库
SQLALCHEMY_BINDS = {
    'sqlserve': 'mssql://dsjxy:dsjxy@47.104.89.208:14333/GYDX_HBN_DSJXY?drive
r=SQL Server',
}
app.config['SQLALCHEMY_BINDS'] = SQLALCHEMY_BINDS
# 跟踪数据库的修改，不建议开启
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

```
db = SQLAlchemy(app)
```

4.6 Flask 对人工智能模型的调用

首先使用 pytorch 编写调用已经训练过的模型的权重，对传入的图像进行预测的一个 a 函数，然后再在 Flask 中调用此方法，所以在 Flask 路由中写的函数方法应为首先接受来自前端图像并将其存入到 Flask 相应的文件夹下，再将其图像传入 a 函数中进行预测，得出的结果插入到数据库中后，再将结果返回到前端进行显示。

```
@app.route('/trashclassficion',methods=['POST'])
def trashclassficion2():
    img = request.files.get('photo')
    path = "../static/photo/"
    file_path = path + img.filename
    img.save(file_path)
    detector = inference.Detector('large', num_classes=12)
    num = str(detector.detect('./weights/best.pkl',file_path))
    result = ""
    if num == "0":
        result = "电池:有害垃圾"
    elif num == "1":
        result = "食物:厨余垃圾"
    elif num == "2":
        result = "棕色玻璃:可回收垃圾"
    elif num == "3":
        result = "纸箱:可回收垃圾"
    elif num == "4":
        result = "衣物:可回收垃圾"
    elif num == "5":
        result = "绿玻璃:可回收垃圾"
    elif num == "6":
        result = "金属:可回收垃圾"
    elif num == "7":
        result = "纸张:可回收垃圾"
    elif num == "8":
        result = "塑料:可回收垃圾"
    elif num == "9":
        result = "鞋子:可回收垃圾"
    elif num == "10":
        result = "口罩:其他垃圾"
```

```
    elif num == "11":
        result = "透明玻璃:可回收垃圾"
    tehstr = ":"
    name = result[:result.index(tehstr)][:]
    thetype = result[result.index(tehstr):][1:]
    thedb = pymysql.connect(host="localhost", user="root",
                            password="hbn", db="itc", port=3306, charset='utf
8')
    cur = thedb.cursor()
    sql1 = "select * from trash where name = %s and type = %s"
    if cur.execute(sql1, [name, thetype]) == 0:
        sql2 = "insert into trash(name,type,frequency) values(%s,%s,1)"
        cur.execute(sql2, [name, thetype])
        thedb.commit()
        thedb.close()
    else:
        sql3 = "update trash set frequency = frequency+1 where name = %s and
type = %s"
        cur.execute(sql3, [name, thetype])
        thedb.commit()
        thedb.close()
return render_template("trashclassficion.html",result = result,file_path = fi
le_path)
```

4.7 Flask 与 Tomcat 之间的通信

SSM 的主要作用转发出相应的网页并在浏览器中展示，在垃圾分类这个功能模块中，传入垃圾图片由 SSM 进行转发，将 form 表单 action 属性改为 Flask 设置的访问接口，Flask 接受到图片并且计算出结果后渲染出页面，并转发出去。我们前端的首页采用的是 iframe 这样就可以将结果页面直接显示出来。

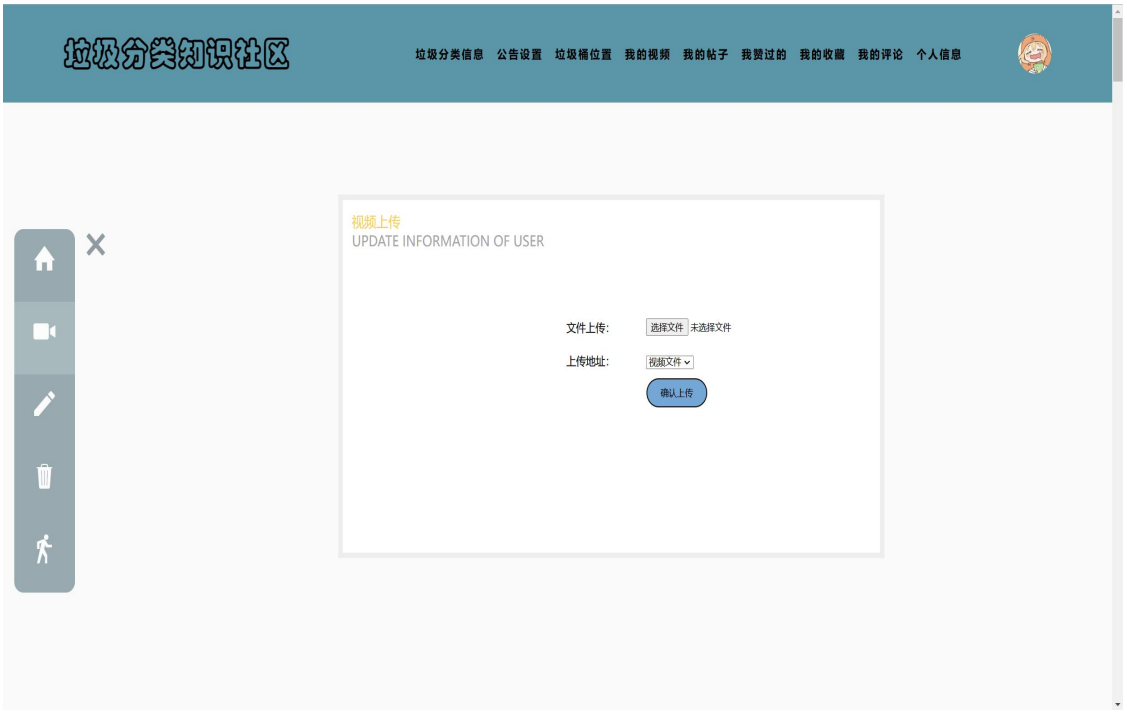
Flask 渲染的页面：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>垃圾分类</title>
</head>
<body>
<div class="rg_layout" style="margin-top: 6%">
    <div class="rg_left">
```

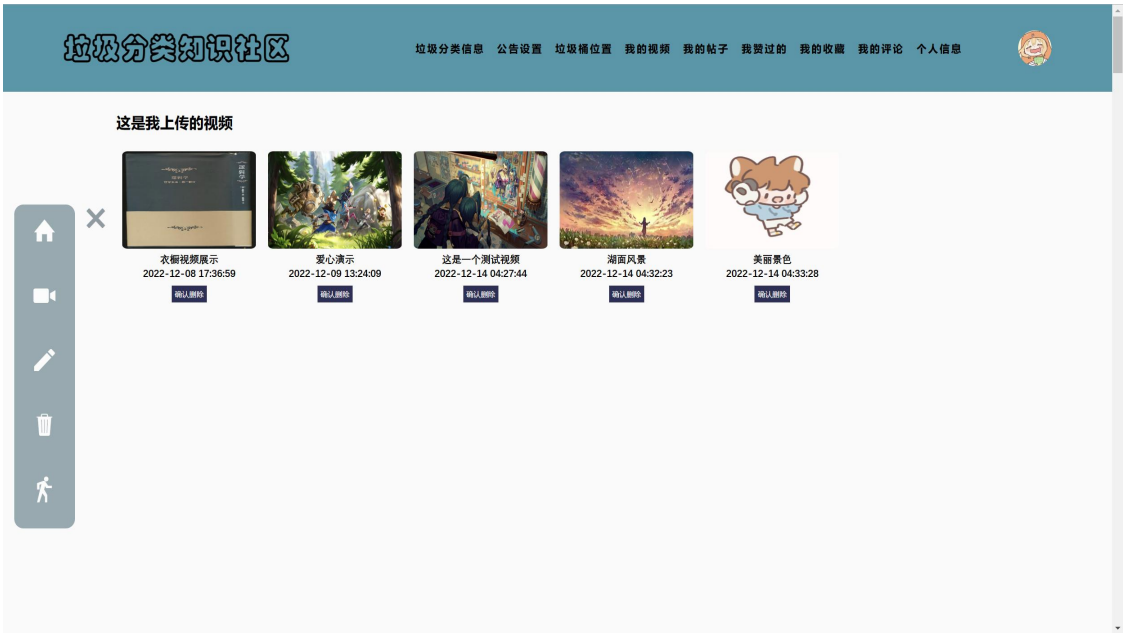
```
        <p>垃圾分类结果</p>
        <p>TRASH CLASSIFICATIONRESULT</p>
    </div>
    <div class="rg_center">
        <div class="rg_form">
            <img src="{{ file_path }}" style="border-radius: 10px" height="330px" width="250px">
            <h4 style="font-family: '等线 Light';color: #01AAED;text-align: center">图片的预测结果分类是{{ result }}</h4>
        </div>
    </div>
</div>
</body>
</html>
```

5. 运行结果

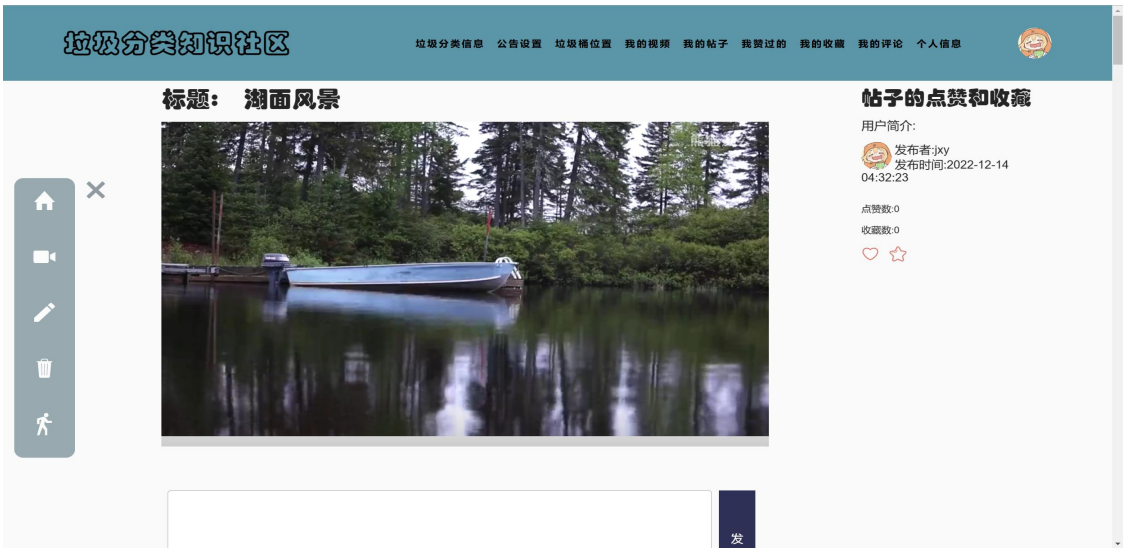
视频上传：



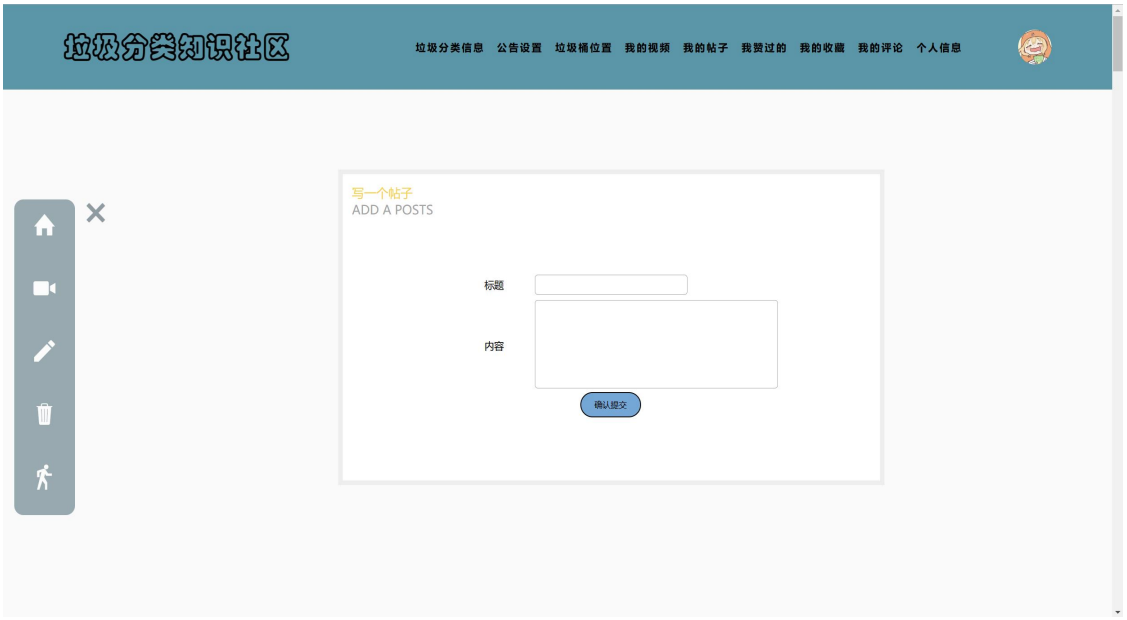
视频删除：



视频展示：



帖子上传：

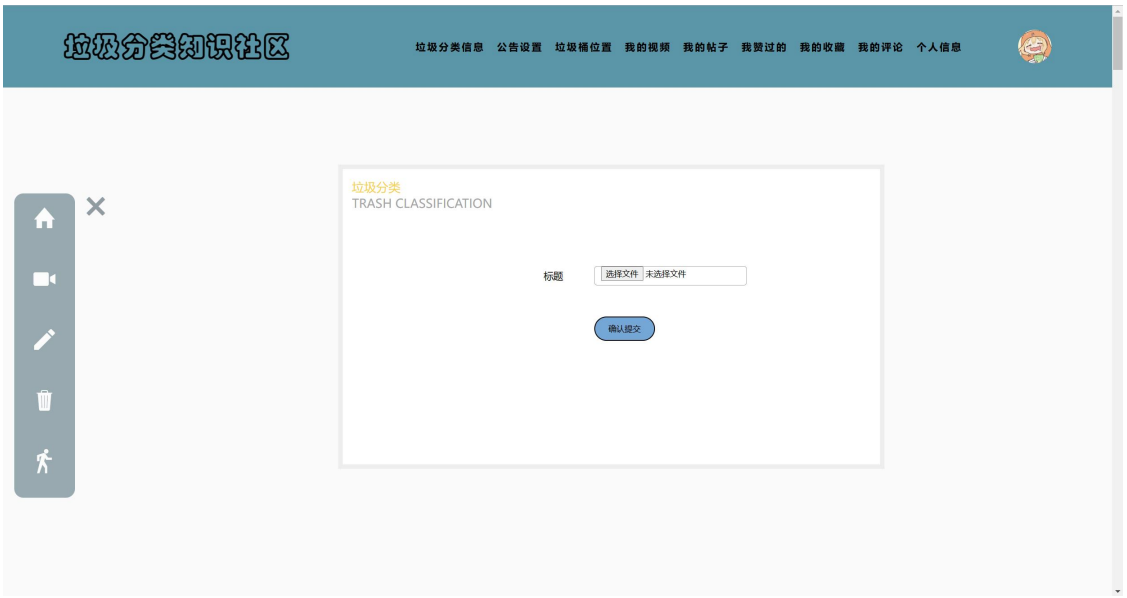


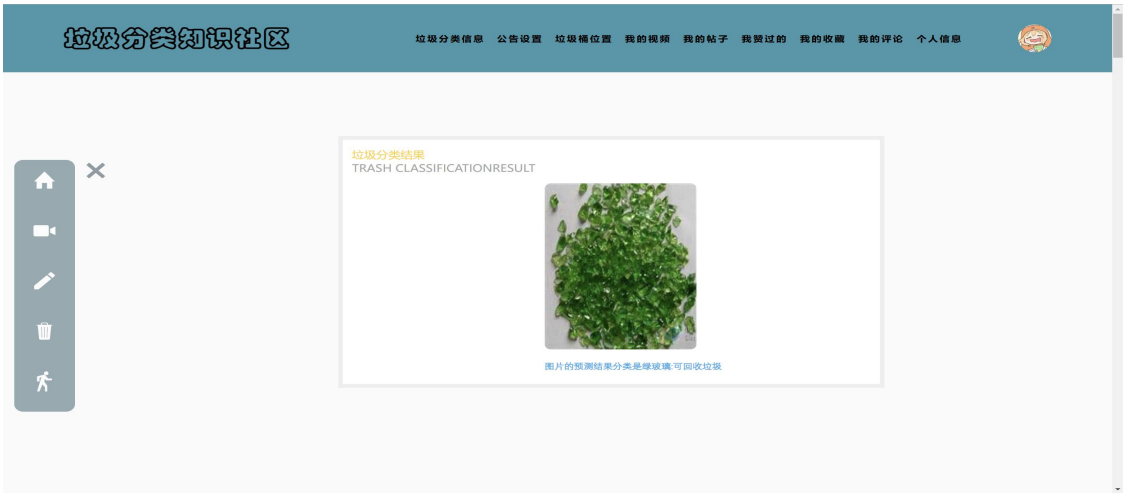
帖子删除：

帖子展示：


垃圾分类知识社区
垃圾分类信息 公告设置 垃圾桶位置 我的视频 我的帖子 我赞过的 我的收藏 我的评论 个人信息
标题： 我们要保护环境
内容: 保护环境从你我做起
发
帖子的点赞和收藏
用户简介:
发布者:jxy
发布时间:2022-12-09 16:06:42.685
点赞数:1
收藏数:1

垃圾分类：


垃圾分类知识社区
垃圾分类信息 公告设置 垃圾桶位置 我的视频 我的帖子 我赞过的 我的收藏 我的评论 个人信息
垃圾分类
TRASH CLASSIFICATION
标题
选择文件 未选择文件
确认提交


垃圾分类知识社区
垃圾分类信息 公告设置 垃圾桶位置 我的视频 我的帖子 我赞过的 我的收藏 我的评论 个人信息
垃圾分类结果
TRASH CLASSIFICATIONRESULT
图片的预测结果分类是绿玻璃:可回收垃圾

6. 调试分析

在本次项目中我负责的视频和帖子的上传和删除展示模块中，最困难的是视频的上传和展示，需要考虑的因素非常的多，要非常了解开发中每一步的步骤，并且一步一个脚印的写代码，不要一口气把一个功能的代码全部写完，这样出现bug 也不知道到底是哪处。在传入视频的过程中我们要注意 web.xml 下的视频上传的配置，上传视频最大内存，超过多少内存才会上传到本地等参数。对于视频也必须进行转码或者限定上传视频的格式。在视频展示的过程中，要按流程将每一步的代码完善完整，这样才能使最后的结果和预期的一致。

7. 总结

想到到“垃圾分类”这个主题之后，先是查阅了相关资料和文献，然后查看了一些他人做的已经成熟的机器学习的模型和相关的有机结合的网站。了解到网站制作是什么样的流程以及怎么样去操作。弄清楚这个概念之后，我又详细理解了老师所讲的设计要求和注意事项，开始思考、分析问题，然后确定了总体的设计思路，提出了问题的解决方案，以及系统设计方案和框架，接下来我就着手开始编程。

在编程的过程中，我先是根据系统所要求，找出所需要的我在 Java、python 和智能系统设计的课程中学习过的知识并将其一一运用。在编程的过程中，出现了许许多多的 bug，我又进行了一次又一次的调试，找出了其中的错误，并将其一一纠正，并且修改了其中不太完善的部分，力求做到实用并且精确。在编程过程中最难的就是 service 和 mapper 的编写，繁琐又容易出错，所以我在阅读他人文献时了解到必须要符合项目的规范这，使用以后极大的提高了实验的效率。通过这次课程设计使我清楚的认识到了，课本上的理论知识固然重要，但也要将其付诸于实践，要养成多动手的好习惯，所以我以后应该要更加注重实践，要多动手，多练习，养成良好的编程习惯，独立思考，从多种角度考虑，寻找最优方案和最优代码。要将理论和实践相结合，提高自己对知识的理解程度。

8.参考文献

[1]基于 MobileNetV3 公共垃圾分类系统 梅书枰 [J]武汉纺织大学 2020-06-01 硕士论文

[2]张宇，王映辉，张翔南．基于 Spring 的 MVC 框架设计与实现［J］计算机工程，2010，36(4):59－62

[3]刘中兵 开发者突击:Java Web 主流框架整合开发:J2EE+ Struts+ Hibernate + Spring［M］北京: 电子工业出版社，2008

[4]贾昆，甘仞初，高慧颖．数据访问对象模式在企业应用集 成中的应用[J].计算机工程与设计，2006，27(3):373 － 375

[5]王坤 基于 J2EE 平台 Spring MVC 框架开发的 MIS 系统设计与实现［D］．上海:华东师范大学，2008