南京大学

计算机科学与技术系

软件工程实验报告

实验名称：软件功能实现

学　　号：171240501

姓　　名：匡舒磊

指导教师：张天

实验地点：基础实验楼 208

实验时间：2019.11.8

一、 实验名称

软件功能实现

二、 实验目的

1. 熟悉 Spring Web 开发框架，了解框架的开发特性并尝试进行 Web 开发；
2. 针对工业 App 分级系统，根据分配到的功能进行初步实现；

三、 实验要求

1. 配置实验环境，并了解 Spring Web 开发框架的使用；
2. 实现“工业 App 分级系统”的相应功能。
3. 完成实验报告。

四、 实验环境

1. 软件：IntelliJ IDEA, DataGrip
2. 硬件：电脑
3. 项目名称：工业 APP 的实现（企业版）

五、 实验内容

1. 项目环境搭建
    a) 参考项目中 tutorial.md 进行项目导入和环境搭建，运行 web 应用
2. 实现 Web 应用功能
    a) 实现帐户管理
    b) 工业 App 分级申请（企业）

六、 实验结果与说明

实现功能：

1）用户的登录、注册、修改个人资料、登出

2）提交审核材料

3）查看已提交材料，查看审核结果

4）删除已提交材料

下对各功能点进行说明

1. 初始界面

登入初始界面后，显示如下：

用户可以点击登录/现在开始界面进入登录界面，或者点击注册按钮进入注册页面。

2. 账户管理

1） 前端页面截图

登录界面：

注册界面

修改个人资料界面

登出选项：

2） 数据库表结构截图

```
CREATE TABLE user (
    StringId VARCHAR(100) NOT NULL,
    user_name VARCHAR(100),
    password VARCHAR(20) NOT NULL,
    PRIMARY KEY (StringId)
)DEFAULT CHARSET=utf8;
```

初始内容如下：

	StringId	user_name	password
1	001	鸡肉工厂	123456
2	002	假手机工厂	111

3) 后端与数据库交互 Mapper 代码截图

```
@Mapper
public interface HelloMapper {
    @Select("select * from user ")
    @Results({
            @Result(property = "id", column = "stringId"),
            @Result(property = "name", column = "user_name"),
            @Result(property = "password", column = "password")
    })
    List<HelloUser> findAll();

    @Insert("insert into user(stringId,user_name,password) values(#{id},#{name},#{password})")
    void insert(HelloUser helloUser);


    @Select("select * from user where stringId = #{id}")
    @Results({
            @Result(property = "id",column = "stringId"),
            @Result(property = "name",column = "user_name"),
            @Result(property = "password", column = "password")
    })
    HelloUser getOne(String id);

    @Update("update user set user_name = #{name}, password = #{password} where StringId = #{id}")
    void updateByID(HelloUser helloUser);//UPDATE 表名称 SET 列名称 = 新值 WHERE 列名称 = 某值

    @Delete("delete from user where StringId = #{id}")
    void deleteByID(String id);//DELETE FROM 表名称 WHERE 列名称 = 值
}
```

4) 请求处理实现代码截图

登录请求实现代码：

```java
@RequestMapping(value = "/login", method = RequestMethod.GET) // 获取用户输入数据
public String hello(Model model) { return "login"; }
@RequestMapping(value = "/login", method = RequestMethod.POST)
public String login(HelloUser user, Model model, HttpSession session) {
    String userId = user.getId();
    String password = user.getPassword();
    logger.info("login  your id and password is :" + userId + " " + password);
    HelloUser one = helloService.getOne(userId);    // 判断对象是否为空
    if (one.getId() == null) {
        model.addAttribute( s: "loginInfo",  o: "该用户不存在");
        return "login";
    } else if (one.getPassword().equals(password)) {
        logger.info("you've logged!!");
        //String name=one.getName();
        model.addAttribute( s: "loginInfo",  o: "登录成功");
        session.setAttribute( s: "usrName",one.getName());
        session.setAttribute( s: "usrID",userId);
        session.setAttribute( s: "usrPassword",password);
        loginUsrId = userId;
        return "redirect:dashboard";
    } else {
        model.addAttribute( s: "loginInfo",  o: "密码错误");
        return "login";
    }
}
```

注册实现代码：

```java
@RequestMapping(value = "/signup", method = RequestMethod.GET) // 获取用户输入数据
public String trySignUp(Model model) { return "signup"; }
@RequestMapping(value = "/signup", method = RequestMethod.POST)
public String signup(HelloUser user, Model model, HttpSession session) {
    String userId = user.getId();
    String usrName = user.getName();
    String password = user.getPassword();
    logger.info("try to signup ID "+userId);
    HelloUser one = helloService.getOne(userId);     // 判断对象是否为空
    if (one.getId() != null) {
        model.addAttribute( s: "signupInfo",  o: "该用户已存在");
        return "signup";
    } else {
        logger.info("sign up new id");
        model.addAttribute( s: "signupInfo",  o: "注册成功");
        Map<String, String> newinfo = new HashMap<~>();
        newinfo.put("id",userId);
        newinfo.put("name",usrName);
        newinfo.put("password",password);
        helloService.InsertUser(newinfo);
        loginUsrId = userId;
        session.setAttribute( s: "usrName",usrName);
        session.setAttribute( s: "usrID",userId);
        session.setAttribute( s: "usrPassword",password);
        loginUsrId = userId;
        return "redirect:dashboard";
    }
}
```

修改个人资料请求代码：

```
@RequestMapping(value = "/usrinfo", method = RequestMethod.GET)
public String usrinfo(){
    if(loginUsrId==null)
        return "redirect:login";
    logger.info("change usrinfo");
    return "usrinfo";
}
@RequestMapping(value = "/usrinfo", method = RequestMethod.POST)
public String changeinfo(HelloUser user, Model model, HttpSession session){
    logger.info("change usrinfo");
    HelloUser one = helloService.getOne(loginUsrId);
    String userName = user.getName();
    String password = user.getPassword();
    Map<String, String> newinfo = new HashMap<~>();
    newinfo.put("id",loginUsrId);
    if(userName!=null){
        newinfo.put("name",userName);
        session.setAttribute( s: "usrName",userName);
    } else {
        newinfo.put("name",one.getName());
    }
    if(password!=null){
        newinfo.put("password",password);
        session.setAttribute( s: "usrPassword",password);
    }else {
        newinfo.put("password",one.getPassword());
    }
    helloService.UpdateByID(newinfo);
    return "redirect:dashboard";
}
```

登出请求代码：

```
@RequestMapping("/logout")
public String logout(HelloUser user, Model model, HttpSession session){
    logger.info("log out now");
    session.setAttribute( s: "usrID", o: null);
    session.setAttribute( s: "usrName", o: null);
    return "redirect:login";
}
```

5) 简单说明

对于登录页面，通过 html 的 input 获取输出的值，然后根据用户输入的用户 ID（在数据库中，id 被设置为 Key 值，不会产生重复）进行搜索，若为空，则利用 model 与 thymeleaf 中的 th 表达式在页面提示账户不存在，若存在，则利用 getPassword 函数获取其数据库中的密码，得到密码后利用 equal 值进行比较，若相等则进入主页面“dashboard”，并且利用 session 对登入的 ID 进行记录，在后续的操作中用于保持用户的登录状态，若密码错误则进行提醒。

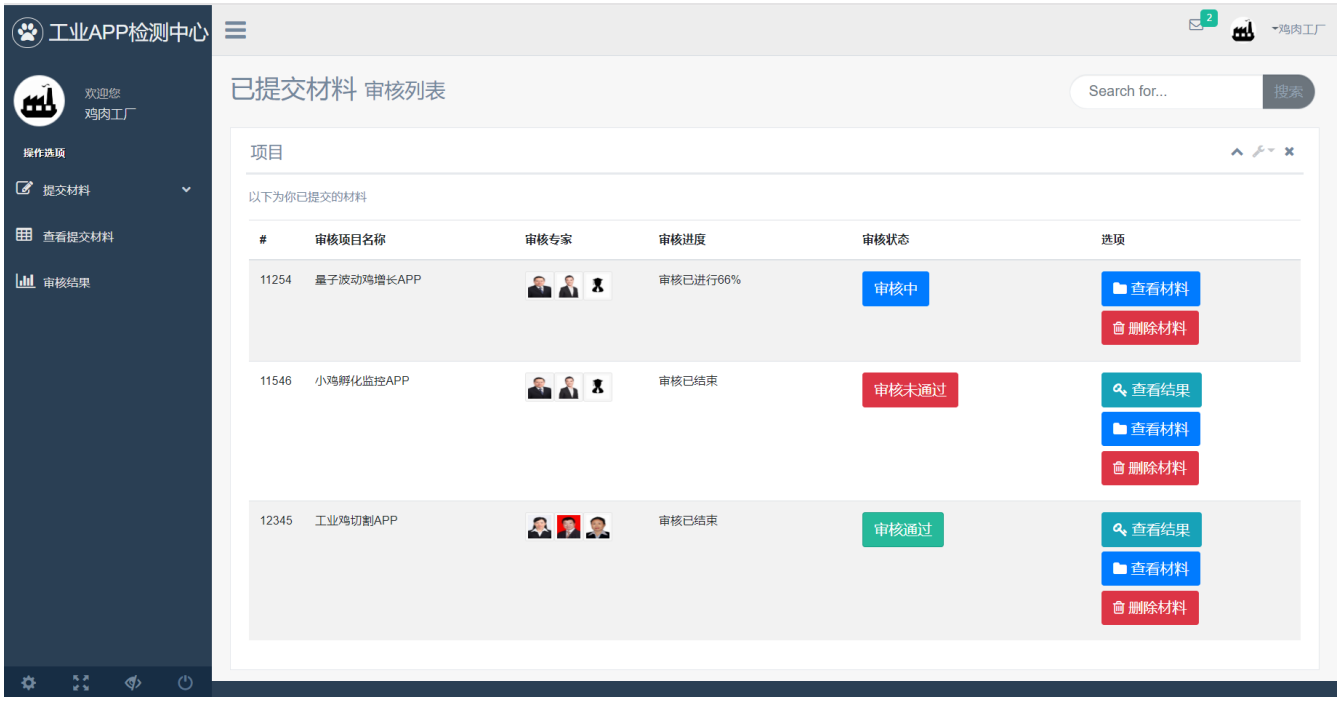
对于注册界面，先对用户提交的 ID 进行搜索，若在数据库中存在此 ID，

则提示已存在此账户，若不存在此 ID，则利用 Insert 函数对其进行插入数据库处理，并用 session 对登入的 ID 进行记录以便在之后保持登录状态。

对于修改个人信息页面，先检查是否已经是登录状态，若未登录则重定向至登录界面，否则的话则根据已经登陆的 ID，利用 updateByID 函数对数据库中的信息进行更新。

对于登出界面，只要将存储的登陆信息进行清空，并且重定向至登陆界面即可。

3. 主界面



数据库与 Mapper 函数见后，先简要说明一下后端响应代码：

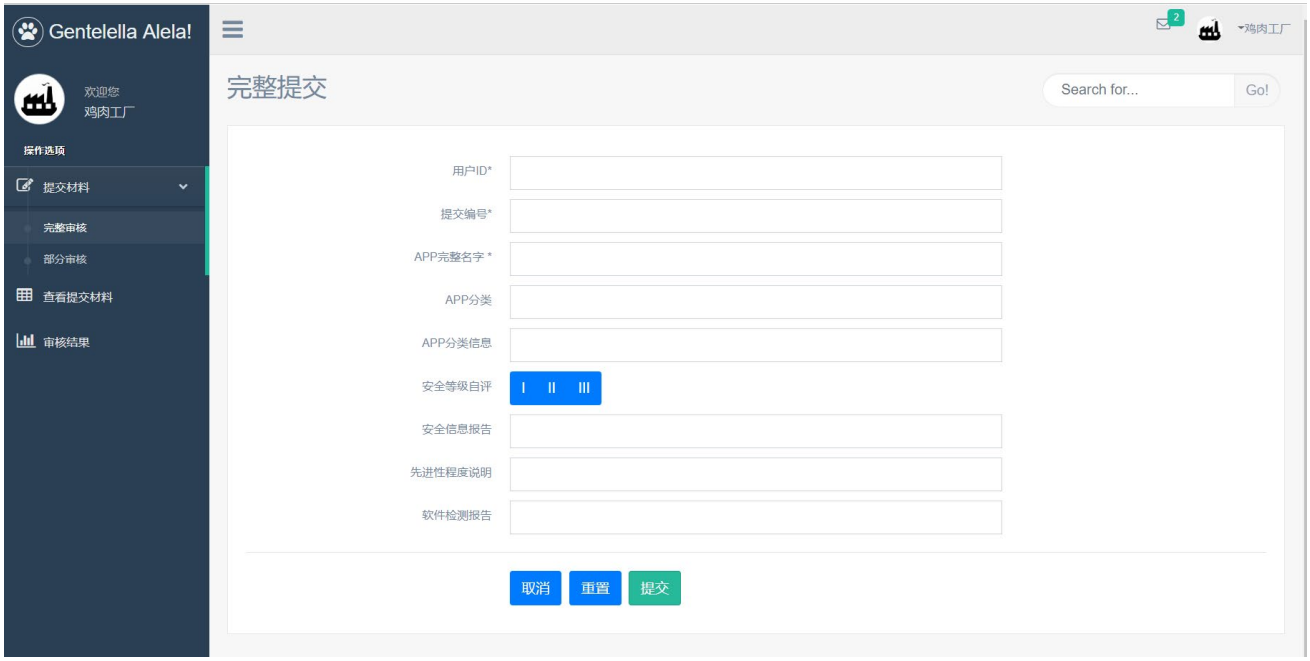
```
@RequestMapping("/dashboard")
public String dashboard(Appcheck app, Model model,HttpSession session){
    String usrId = (String) session.getAttribute( s: "usrID");
    if(usrId==null)
        return "redirect:login";
    List<Appcheck> list = appcheckService.getAppByUsr(usrId);
    model.addAttribute( s: "list", list);
    return "dashboard";
}
```

进入主页面后，先判断是否处于登陆状态，如果未登录则自动进入登录界面要求客户进行登录。若已登录，则在数据库中利用 getAppByUsr 函数(在产品数据库中每个都与一个用户 ID 进行了绑定)，然后利用 model 对这个用户的 id 绑定的产品列表进

行绑定，在 dashboard 的 html 文件中利用 thymeleaf 的 th 表达式，对用户提交过的材料进行排列。其中排列时会利用 th：if 语句对不同的审核进度进行不同的展示，如在未审核完成前不会显示查看结果的选项。

4. 提交材料

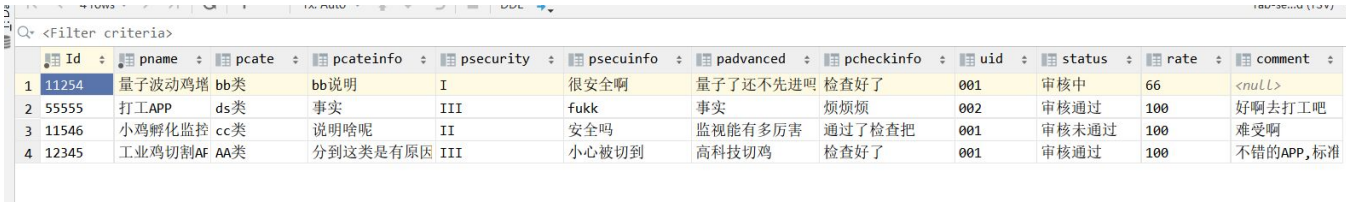
1） 前端页面截图



2） 数据库表结构截图

```
CREATE TABLE app(
    Id VARCHAR(10) NOT NULL ,
    pname VARCHAR(100) NOT NULL ,
    pcate VARCHAR(100),
    pcateinfo VARCHAR(500),
    psecurity VARCHAR(100),
    psecuinfo VARCHAR(500),
    padvanced VARCHAR(400),
    pcheckinfo VARCHAR(200),
    uid VARCHAR(100),
    status VARCHAR(50),
    rate VARCHAR(10),
    comment VARCHAR(200)
);
```

初始数据如下：



<Filter criteria>

	Id	pname	pcate	pcateinfo	psecurity	psecuinfo	padvanced	pcheckinfo	uid	status	rate	comment
1	11254	量子波动鸡增	bb类	bb说明	I	很安全啊	量子了还不先进吗	检查好了	001	审核中	66	<null>
2	55555	打工APP	ds类	事实	III	fukk	事实	烦烦烦	002	审核通过	100	好啊去打工吧
3	11546	小鸡孵化监控	cc类	说明啥呢	II	安全吗	监视能有多厉害	通过了检查把	001	审核未通过	100	难受啊
4	12345	工业鸡切割AF	AA类	分到这类是有原因	III	小心被切到	高科技切鸡	检查好了	001	审核通过	100	不错的APP,标准

3) 后端与数据库交互 Mapper 代码截图

```
@Mapper
public interface AppcheckMapper {
    @Select("select * from app ")
    @Results({
            @Result(property = "pid", column = "Id"),
            @Result(property = "pname", column = "pname"),
            @Result(property = "pcate", column = "pcate"),
            @Result(property = "pcateinfo",column = "pcateinfo"),
            @Result(property = "psecurity",column = "psecurity"),
            @Result(property = "psecuinfo",column = "psecuinfo"),
            @Result(property = "padvanced",column = "padvanced"),
            @Result(property = "pcheckinfo",column = "pcheckinfo"),
            @Result(property = "uid",column = "uid"),
            @Result(property = "status",column = "status"),
            @Result(property = "rate",column = "rate")
    })
    List<Appcheck> findAllApp();

    @Insert("insert into app(Id,pname,pcate,pcateinfo,psecurity,psecuinfo,padvanced,pcheckinfo,uid,status,rate,comment) values(#{pid},#{pname},#{pcate},#{pc
    void insertApp(Appcheck appcheck);

    @Select("select * from app where Id=#{pid}")
    @Results({
            @Result(property = "pid", column = "Id"),
            @Result(property = "pname", column = "pname"),
            @Result(property = "pcate", column = "pcate"),
            @Result(property = "pcateinfo",column = "pcateinfo"),
            @Result(property = "psecurity",column = "psecurity"),
            @Result(property = "psecuinfo",column = "psecuinfo"),
            @Result(property = "padvanced",column = "padvanced"),
            @Result(property = "pcheckinfo",column = "pcheckinfo"),
            @Result(property = "uid",column = "uid"),
            @Result(property = "status",column = "status"),
            @Result(property = "rate",column = "rate")
    })
    Appcheck getOneApp(String pid);

    @Select("select * from app where uid=#{uid}")
    @Results({
            @Result(property = "pid", column = "Id"),
            @Result(property = "pname", column = "pname"),
            @Result(property = "pcate", column = "pcate"),
            @Result(property = "pcateinfo",column = "pcateinfo"),
            @Result(property = "psecurity",column = "psecurity"),
            @Result(property = "psecuinfo",column = "psecuinfo"),
            @Result(property = "padvanced",column = "padvanced"),
            @Result(property = "pcheckinfo",column = "pcheckinfo"),
            @Result(property = "uid",column = "uid"),
            @Result(property = "status",column = "status"),
            @Result(property = "rate",column = "rate")
    })
    List<Appcheck> getAppByUsr(String uid);

    @Update("update app set pname = #{pname},pcate = #{pcate},pcateinfo = #{pcateinfo},psecurity = #{psecurity},psecuinfo = #{psecuinfo},padvanced = #{padva
    void updateAppById(Appcheck appcheck);

    @Delete("delete from app where Id=#{pid}")
    void deleteApp(String pid);
}
```

4) Service 代码

```java
@Service
public class AppcheckService {
    @Resource
    private AppcheckMapper appcheckMapper;

    public List<Appcheck> getAppList(){
        List<Appcheck> list = appcheckMapper.findAllApp();
        return list;
    }

    public void InsertApp(Map<String, String> params){
        ObjectMapper objectMapper = new ObjectMapper();
        Appcheck appcheck = objectMapper.convertValue(params, Appcheck.class
        appcheckMapper.insertApp(appcheck);
    }

    public Appcheck getOneApp(String pid){
        Appcheck result = appcheckMapper.getOneApp(pid);
        System.out.println("getOneApp:"+result);
        if (result==null)
        {
            result=new Appcheck();
        }
        System.out.println(result.toString());
        return result;
    }

    public List<Appcheck> getAppByUsr(String uid){
        List<Appcheck> list = appcheckMapper.getAppByUsr(uid);
        return list;
    }


    public void UpdateByID(Map<String, String> params){
        String pid = params.get("pid");
        Appcheck temp = appcheckMapper.getOneApp(pid);
        if(params.get("pname")!=null)
            temp.setPname(params.get("pname"));
        if(params.get("pcate")!=null)
            temp.setPcate(params.get("pcate"));
        if(params.get("pcateinfo")!=null)
            temp.setPcateinfo(params.get("pcateinfo"));
        if(params.get("psecurity")!=null)
            temp.setPsecurity(params.get("psecurity"));
        if(params.get("psecuinfo")!=null)
            temp.setPsecuinfo(params.get("psecuinfo"));
        if(params.get("padvanced")!=null)
            temp.setPadvanced(params.get("padvanced"));
        if(params.get("pcheckinfo")!=null)
            temp.setPcheckinfo(params.get("pcheckinfo"));
        if(params.get("uid")!=null)
            temp.setUid(params.get("uid"));
        if(params.get("status")!=null)
            temp.setStatus(params.get("status"));
        if(params.get("rate")!=null)
            temp.setRate(params.get("rate"));
        appcheckMapper.updateAppById(temp);
    }

    public void DeleteByID(String pid){
        appcheckMapper.deleteApp(pid);;
        System.out.println("AfterDelete:"+appcheckMapper.getOneApp(pid));
    }
}
```

5） 后端请求处理代码

```java
@RequestMapping(value = "addApp", method = RequestMethod.GET)
public String ApptoAdd(){
    logger.info("createNewAppForm");
    return "addApp";
}
@RequestMapping(value = "addApp", method = RequestMethod.POST)
public String addApp(Appcheck app, Model model, HttpSession session){
    String pid = app.getPid();
    Appcheck apptest = appcheckService.getOneApp(pid);
    System.out.println("get pid is "+pid);
    if(apptest.getPname()!=null){
        model.addAttribute( s: "addAppInfo", o: "此编号已被占用，请选择其他编号");
        return "addApp";
    }
    String uid=app.getUid();
    String pname=app.getPname();
    String pcate=app.getPcate();
    String pcateinfo=app.getPcateinfo();
    String psecurity=app.getPsecurity();
    String psecuinfo=app.getPsecuinfo();
    String padvanced=app.getPadvanced();
    String pcheckinfo=app.getPcheckinfo();
    Map<String, String> newapp = new HashMap<~>();
    newapp.put("pid",pid);
    newapp.put("pname",pname);
    newapp.put("pcate",pcate);
    newapp.put("pcateinfo",pcateinfo);
    newapp.put("psecurity",psecurity);
    newapp.put("psecuinfo",psecuinfo);
    newapp.put("padvanced",padvanced);
    newapp.put("pcheckinfo",pcheckinfo);
    newapp.put("uid",uid);
    newapp.put("status","待审核");
    newapp.put("rate","0");
    newapp.put("comment",null);
    appcheckService.InsertApp(newapp);
    return "redirect:dashboard";
}
```

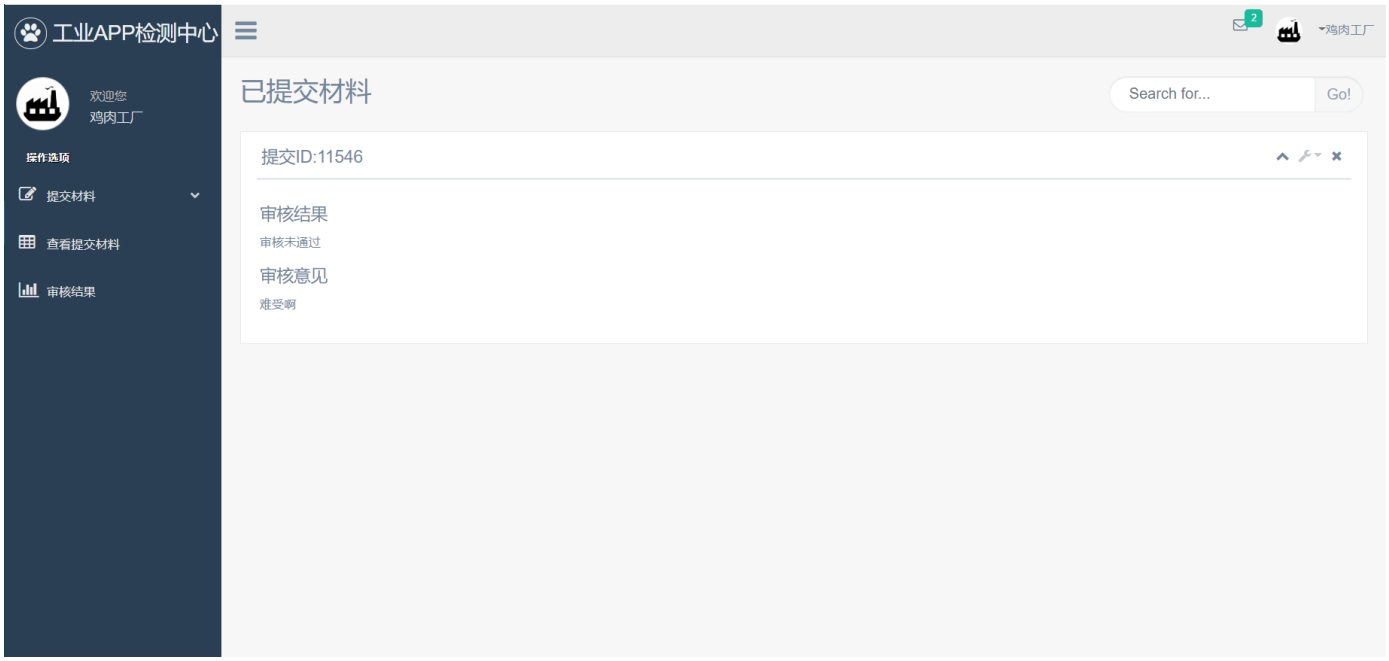
6) 简单说明。

企业提交材料的数据库由以下几个部分构成：企业用户 id，材料 id，提交的相关材料与说明，审核进度，审核状态，审核意见，其中审核意见在进入不为 100 的情况下，审核意见为 null。

企业在进行材料审核提交的时候，需要保证提交材料编号不与数据库中的任何一个编号重复，若重复则提醒用户更换编号，之后利用 Map 对插入的提交材料进行设置，在材料后的进度设置为待审核，进入设置为 0，意见为 null.

5. 已提交材料的材料查看与结果查看，删除材料

1) 前端界面



2）后端处理代码

```
@RequestMapping("/viewM/{pid}")
public String viewma(@PathVariable("pid") Integer pid,Model model,HttpSession session){
    String id = toString().valueOf(pid);
    Appcheck app = appcheckService.getOneApp(id);
    session.setAttribute( s: "appToShow",app);
    return "redirect:../material";
}

@RequestMapping("/viewA/{pid}")
public String viewAns(@PathVariable("pid") Integer pid,Model model,HttpSession session){
    String id = toString().valueOf(pid);
    Appcheck app = appcheckService.getOneApp(id);
    session.setAttribute( s: "appToShow",app);
    return "redirect:../checkResult";
}

@RequestMapping("/deleteM/{pid}")
public String deleteM(@PathVariable("pid") Integer pid,Model model,HttpSession session){
    String id = toString().valueOf(pid);
    appcheckService.DeleteByID(id);
    return "redirect:../dashboard";
}

@RequestMapping(value="/material",method = RequestMethod.GET)
public String getMaterial() { return "material"; }

@RequestMapping(value="/checkResult",method = RequestMethod.GET)
public String getResult() { return "checkResult"; }
```

在前端的 html 文件中，在 dashboard 界面中（见上），每一个材料后跟了两个或者三个选项，其中包括查看已提交材料(后端调用 viewM/{pid}，pid 为材料提交编号)，查看审核意见（后端调用 viewA/{pid}），删除材料（后端调用 deleteM/{pid}），然后前两者在获取 pid 后将 pid 对应的材料利用 session 绑定在展示属性中，之后重定向至相对应的展示界面，在界面的 html 中利用 thymeleaf 函数对其进行展示。而若为删除材料则直接利用传递过来的 pid 利用 DeleteByID 函数进行删除。

七、 结论

本次实验花费时间较长，大约花了 20 多个小时的时间，实验报告的撰写花了一小时左右，最主要的耗时在于对相关语言的不熟悉，我采用的是先考虑前端 ui，再考虑后端进行相应与数据库进行交互的内容，然后在前端上由于我之前没有接触过相关的内容，网路上也是各种相关的内容都有，在找到合适的框架上花费了很多的时间（现在似乎大多数比较流行的框架更注重于移动层次）。之后利用了一个基于 Bootstrap 的框架。此处鸣谢（https://github.com/ColorlibHQ/gentelella）开源框架 gentelella。之后的话主要是在相关的根据不同的用户展示不同的界面卡住了，然后在同学的提示下学习了 Thymeleaf 的 th 标签，同时也感谢助教老师相关的帮助。

实验评分： _____

指导教师签字： _____

年　　月　　日