**1（首页）**

各位老师大家好，我是本次的答辩人xxx，

本次毕业设计的题目是：融创富文本音视频集合的文档社区平台，

下面开始我的答辩。

**2（目录）**

本次答辩内容主要分为以下三个部分：

项目简介、功能演示和详细设计，

其中详细设计包括用户端和管理端。

**3（项目简介）**

这是项目的主要功能模块，分为用户和管理端。旁边是论文里面的插图。

管理员和用户的功能基本是相互对应的，所以重复的功能就不再单独介绍。

在“新增发布”这一个页面中，实现了多个功能，包括分类、上传、富文本编辑。

**4（项目简介）**

本次设计的主要内容就是，实现一个功能完备、界面友好的文档社区平台。

系统的主要功能包括这些。

**5（功能演示）**

下面我先进行演示，然后再介绍实现的过程。

**6（详细设计——整体介绍：技术框架）**

整个系统是一个基于“Vue+SpringBoot”的前后端分离项目。

前端框架是“Vue+ElementUI”，后端是“SpringBoot+Mybatis”，数据库采用MySQL

前端主要用于构建页面，与用户交互，绑定数据、绑定事件、和发送请求。

后端用于接收前端请求、编写方法和SQL语句。

数据库的作用就是存储数据和操作数据。

**7（详细设计——整体介绍：功能实现）**

根据项目特点，可以把系统的主要功能，分为“基础CRUD”，和其他部分，

因为几乎每个模块都是在增、删、改、查的基础上，附加其他功能实现的。

用户操作界面时，前端发送请求，调用后端接口，

后端通过MyBatis操作数据库，并将数据结果返回给前端。

在前端发送请求，和后端返回结果之前，都要进行数据封装：

前端通过Request封装请求消息；后端通过Params封装，用于接收前端参数；通过Result封装，用于返回最终结果。

在后端，将按照Controller--->Service--->Dao/Mapper--->Entity…的顺序依次调用。

下面我会依次介绍这些功能的实现。

**8-9（详细设计——登录注册）**

登录注册部分。

用户输入用户名、密码和验证码，点击登录。

登录时先验证验证码，再验证用户名和密码。

注册时用户名不能为空，也不能重复，密码和身份可以为空，

如果为空，就使用默认的用户名和密码。

登录注册的实现过程是这样的。

首先要在后端配置好图形验证码和JWT拦截器。

图形验证码是在pom.xml文件里面引入的一个工具依赖，

每当用户点击图片或者刷新页面的时候，前端就会生成一个随机数，生成一个不一样的验证码。

JWT的作用是，配置Web拦截器，不允许未登录用户也就是非法用户，

访问系统的其他页面。当一个用户登录以后，系统就会根据用户名和密码，

生成一个token，用户每次登录都会生成一个token，每个token都是唯一的，

而且有有效期，当一个用户登录以后，在有效期内，就可以不用重复登录，访问系统的其他页面。

用户登录后的token保存在浏览器的 local Storage 里，可以直接访问local Storage获取当前登录用户的信息。

**10-11（详细设计——日志记录）**

日志记录部分。每当用户进行了某一操作时，比如登录，新增或删除，

系统就会自动记录下来。

日志记录部分使用了面向切面编程的原理，也就是这里的AOP拦截机制。

首先在后端需要创建一个自定义注解，它里面有一个value属性，

就是用于记录日志的内容。

比如需要记录登录日志，就在登录方法上面加上自定义的注解。

下面这个 Around 环绕注解，就是用于监听 auto Log 注解的，

当auto Log 注解下面的方法被调用以后，程序就会切换到 Around 注解，

先执行 Around 注解里面的方法。通过auto Log . value () 获取操作内容，

获取操作用户、操作IP、和操作时间，然后保存到数据库。

经过前端调用，返回给用户。

**12-13（详细设计——文件接口）**

文件接口部分，在“新增发布”的文件上传下载，和“个人中心”的头像上传部分，会用到文件接口。

前端通过el-upload 标签进行文件上传操作，

其中有一个action属性，用于调用后端的文件处理接口，

当文件上传后，会调用 on-success 绑定的回调方法，将上传信息保存到数据库。

文件的上传信息，是根据文件的上传时间和文件名组合而成的。

在前端，可以通过时间戳和后端配置的路径，调用文件接口，进行文件上传或下载。

当文件上传以后，需要在前端进行预览。

前端需要对得到的文件名进行解析，判断文件的格式，

根据不同文件类型选择不同的标签进行展示。

**14-15（详细设计——富文本编辑）**

富文本编辑部分。

富文本编辑器引入了 wangeditor ，这是一个开源组件，

通过 npm 直接安装，在前端通过 import 引入使用。

需要注意的是，一个页面上只能创建一个编辑器实例，

所以当用户每次编辑不同区域的内容的时候，

需要先销毁旧的编辑器实例，再重写创建新的编辑器实例。

每次打开编辑器的时候，都要查询数据库，重新赋值编辑器的内容。

如果一个编辑器没有内容，则不能调用editor.txt.html() 获取编辑器的内容。

**16-17（详细设计——协同编辑）**

协同编辑部分，就是当两个或多个用户同时编辑一个文本区的时候，

他们编辑的内容不会发生覆盖冲突，和合并冲突，而是完整保留两者的内容。

协同编辑是基于 WebSocket 实现的，WebSocket 是一个全双工通信协议，

能够编辑操作的实时同步、和跨域通信。当一个用户在文本框中输入内容时，

WebSocket 连接会将内容发送到整个服务器，

每个登录用户都能获取到当前的编辑信息。

首先需要在后端编写好配置方法，

前端在 mounted 方法里编写创建方法，每次刷新页面时就能建立 Socket 通信。

前端通过 textarea 标签绑定用户输入，当用户编辑文本区时，就向后端发送消息，

连接器的消息是全局共享的，为了区分不同文本区编辑的内容，

可以通过设置一个ID字段加以区分。

每当用户编辑文本区，前端就创建WebSocket 连接，把消息传递到后端，并保存到数据库，再通过onmessage 获取连接器的消息，split处理得到的数据，获取 content 字段，绑定到文本区中。