综合实践能力创新实训（1） 课程设计报告

**专业班级：** .

**姓 名：** .

**学 号：** .

**指导教师：** .

**时 间**： .

**地 点**： .

|  |
| --- |
| **指导教师评语：**  **成绩： 等级：**  **签名：**  **年 月 日** |

1. **实验题目**

图书管理系统 & 仿QQWeb即时聊天系统  
**二、实验目的**

1. 复习和巩固软件设计知识，加深对软件设计方法、技术和思想的理解。
2. 运用所学软件设计知识和WEB工程技术进行综合软件设计，通过本课程设计让学生进行简单WEB应用软件系统的开发。
3. 掌握软件设计的方法和面向对象程序设计的基本技术，提高在软件开发方面的综合应用能力。
4. 仿QQWeb即时聊天系统要实现Web点对点即时的文本消息聊天功能，实现Web表情的发送、接收、显示功能，实现Web的图片的发送、接收、显示功能，实现本地消息的存储，在离线的时候也能加载和查看历史消息，要求使用WebSocket。
5. 图书管理系统要求实现学生信息的输入、图书信息的输入、图书信息的查询、学生信息的查询、实现图书的借出和归还功能。

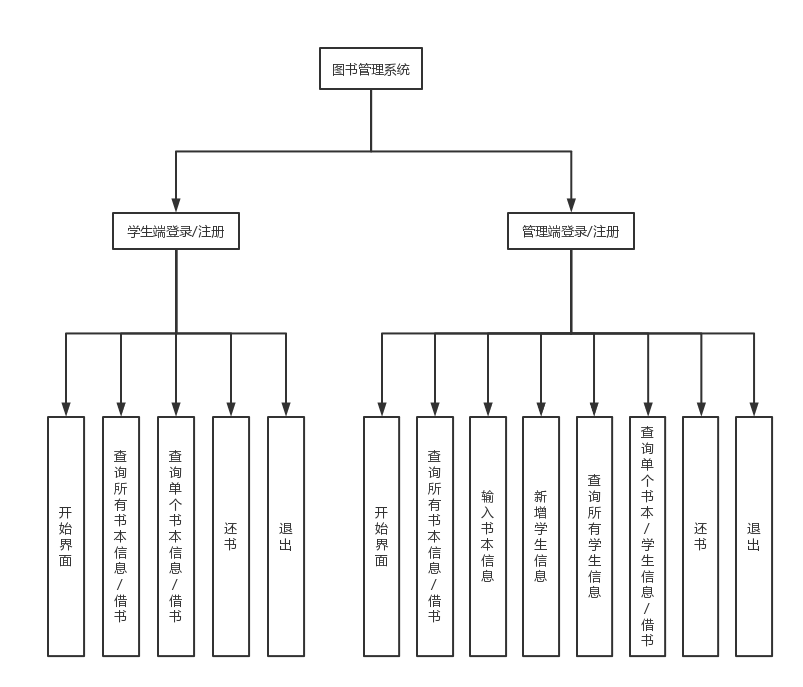
**三、总体设计**

* **背景知识：**

1. 知识细究：
2. 在Java中，Model是一个接口，它代表着数据模型。Model可以通过addAttribute()方法来向View层传递数据，其中addAttribute()方法的实现取决于你使用的具体框架或技术。在Spring框架中，Model接口主要被用于将数据传递给View层。当你调用model.addAttribute("student", student)时，这个方法会将student对象存储为一个名为"student"的属性，并将其添加到Model中。在视图渲染过程中，可以通过这个属性名来获取该属性的值，并将其展示在页面上。在Spring MVC中，Model接口通常是由DispatcherServlet创建并传递给Controller层，在执行完Controller代码后，DispatcherServlet将Model中的属性传递给View层进行渲染。底层原理就是将属性值以键值对的形式存储在Model中，这些键值对可以在View层中进行访问和展示。
3. @RequestMapping("/AdminShowOne2")是Spring框架中用于处理HTTP请求映射的注解之一。它可以将一个特定的HTTP请求映射到一个特定的方法上，并在该方法被调用时执行相应的业务逻辑。具体来说，当客户端向服务器发送一个HTTP请求时，Spring框架会根据@RequestMapping注解配置的URL路径和请求方法（例如GET、POST、PUT等）来确定需要调用哪个方法来处理该请求。如果请求的URL路径与@RequestMapping注解中配置的路径匹配，则相应的方法将被调用，并且该方法可以访问该请求的参数、请求头信息等内容。在上述例子中，@RequestMapping("/AdminShowOne2")表示将URL路径为"/AdminShowOne2"的HTTP请求映射到相应的方法上，从而实现对该请求的处理。这个方法可能在后台进行一些数据查询或者其他操作，然后返回一个结果给客户端。
4. @Controller 是 Spring MVC 框架中的一个注解，用于标识一个类为控制器，用于处理 HTTP 请求。其底层原理涉及到 Spring MVC 的请求处理流程，可以简单理解为以下过程：当客户端发送请求时，DispatcherServlet 接收并处理该请求。DispatcherServlet 根据请求 URL 和 HandlerMapping（又称为映射处理器）找到对应的 Controller，其中 @Controller 注解的类会被扫描并注册到 Spring 的应用上下文中。一旦找到对应的 Controller，DispatcherServlet 根据请求方法（GET、POST 等）和请求参数调用对应的方法，该方法使用 @RequestMapping 注解标识。方法返回一个 Model 对象（或者其他对象），其中包含响应中需要的数据。DispatcherServlet 使用 ViewResolver（视图解析器）解析出视图（View）对象，该对象负责将模型数据渲染成最终的 HTML 页面。最终，DispatcherServlet 通过 HttpServletResponse 将响应发送给客户端。在这个过程中，@Controller 注解实际上是一个特殊的 @Component 注解，并且它被作为一个组件注册到 Spring 应用上下文中。当 DispatcherServlet 扫描到带有 @Controller 注解的类时，它会自动创建一个 Bean 实例并将其注册到 Spring 应用上下文中，并将该实例用于处理请求。因此，@Controller 注解的类实际上是一个 Spring Bean，可以通过依赖注入等方式使用其他 Spring 功能。
5. SSM框架是目前比较流行的一种Java Web开发框架，其包含Spring、SpringMVC和MyBatis三个框架，可以提供全面的企业级Web应用程序开发解决方案。使用SSM框架可以使得开发人员更加专注于业务逻辑的实现，同时也可以提高项目的开发效率和可维护性。
6. MySQL是一种关系型数据库管理系统，它是一种开源软件，常用于支持Web应用程序。在图书管理系统中，MySQL作为底层数据库可以存储各种图书、期刊、论文等相关信息，并支持查询、更新、删除等操作。
7. 本次课设还采用了前端技术，主要包括 HTML、CSS 和 JavaScript 等。HTML 用于构建页面结构，CSS 用于样式布局，JavaScript 用于实现动态交互效果。通过前端技术，可以实现用户登录界面、好友列表展示界面、聊天界面等功能。
8. Spring Boot 是一个基于 Spring 框架的快速开发 Web 应用程序的框架，它通过简化 Spring 的配置和部署过程，使得开发者可以更加专注于业务逻辑的实现。在本系统中，使用 Spring Boot 框架可以快速搭建出后台服务，并实现用户登录、好友列表展示、消息发送与接收等功能。
9. WebSocket 是一种在单个 TCP 连接上进行全双工通信的协议。它使得客户端和服务器之间能够实时地进行双向通信，从而支持实时聊天等应用场景。在本系统中，使用 WebSocket 技术可以实现用户之间的实时消息发送和接收。

* **基本原理：**

1. 需求分析：
2. 实现学生信息的输入功能
3. 实现图书信息的输入功能
4. 实现图书信息的查询功能
5. 实现学生信息的查询功能
6. 实现实现图书的借出和归还功能
7. 系统模块：
8. 登录注册模块
9. 图书信息模块
10. 管理员管理模块
11. 学生管理模块
12. 查询书本模块
13. 查询学生信息模块
14. 借书模块
15. 还书模块
16. 设计思路：
17. 总体设计：



1. 数据库设计：

数据库采用的是的mysql数据库，系统主要需要维护4张数据表，student表、admin表、havebooks表、books表。

* **模块介绍：**

1. 登录注册模块：

当用学生输入账号和密码后，进行验证，如果账号和密码其中有一个不正确，则不能登录，并有相应的提示。如果学生没有账号，可以在登录页面进行注册，如果所注册的账号和之前学生或管理员注册的账号重复，则会提示账号重复灯信息，注册失败，需要重新注册。

当用管理员输入账号和密码后，进行验证，如果账号和密码其中有一个不正确，则不能登录，并有相应的提示。如果管理员没有账号，可以在登录页面进行注册，如果所注册的账号和之前学生或管理员注册的账号重复，则会提示账号重复灯信息，注册失败，需要重新注册。

1. 管理员管理模块

管理员进入系统后，可以进行所有学生信息的查询，新增学生信息，也可以根据学生账号查询单个的学生信息。管理员还可以对所有图书信息进行查询，也可以对单个图书信息进行查询。管理员也可以借阅或归还图书，最后面还可以退出系统。

1. 学生管理模块

学生进入系统后，可以进行图书资料的查询，还可以借阅和退还图书，所借书本信息也可以全部显示出来。学生还可以选择退出系统。

1. 查询书本模块

学生或者管理员进入系统后，可以根据书名查询该书的详细信息，也可以查询所有书本的信息，还可以查询所借所有书本的详细信息。

1. 查询学生信息模块

管理员进入系统后，可以根据学生账号查询学生信息，也可以查询所有学生信息。

1. 借书模块

学生或者管理员可以在查看所有书本信息进行借书操作，也可以在查询单个书本信息时进行借书。

1. 还书模块

学生或者管理员在查阅所借所有书本信息的时候进行还书操作。

* **设计步骤：**

1. 系统需求分析

在本次课设中，需要开发一个图书管理系统。该系统应当能够实现学生信息和图书信息的输入，以及图书和学生信息的查询功能，同时还需要实现图书的借出和归还功能。

1. 数据库设计

基于以上需求，我们可以设计出如下的数据库结构：

1. 学生表（student）：用于存储学生的基本信息，包括账号、学号、姓名、性别、年龄等字段。
2. 管理员表（admin）：用于存储管理员的基本信息，包括账号、姓名、性别、年龄等字段。
3. 图书表（book）：用于存储图书的基本信息，包括书名、作者、书本数目等字段
4. 借书表（havebooks）：用于借书记录，包括借书时间、所借书本名称、借书人账号等字段。
5. 界面设计

为了满足用户的使用需求，我们需要设计出易用的界面。具体来说，采用以下设计方案。

1. 登陆注册界面：学生、管理员输入账号密码，或者注册新的账号密码后登录，可以进入系统。
2. 主界面：学生、管理员进入系统后，可以看到欢迎界面，包含学生信息、图书信息、查询和借还功能的侧边栏按钮。
3. 学生信息输入界面：用于输入新的学生信息。
4. 图书信息输入界面：用于输入新的图书信息。
5. 查询界面：提供多种查询方式，包括按学号查询学生信息、按书名查询图书信息等。
6. 借还界面：用于借出和归还图书。
7. 编码实现

在完成上述设计后，需要开始进行编码实现。这一过程中，可以采用面向对象编程技术，将系统分为多个模块，每个模块负责一个功能。

1. 学生管理模块：负责学生信息的输入和查询等。
2. 管理员模块：负责管理员界面的相关跳转，以及学生信息的展示等。
3. 登录注册模块：负责学生、管理员的登录和注册。
4. 图书管理模块：负责图书信息的输入和查询等。
5. 测试与优化

在完成编码实现后，进行全面的测试和优化。测试应该涵盖各个功能模块，以确保系统的稳定性和可靠性。如果发现任何问题，应该及时解决并进行优化。

1. 实验报告

最后，需要将整个开发过程进行文档化。文档内容应该包括需求分析、数据库设计、界面设计、编码实现、测试报告等，以方便后续的维护和升级工作。 **四、详细设计**

1. **课设所用数据库表的详细信息：**

* **book表：**

这个数据表名为book，包含四个字段：bookName、bookPrice、author和number。bookName是一个varchar类型的字段，用于存储书籍名称。bookPrice是一个int类型的字段，用于存储书籍价格。author是一个varchar类型的字段，用于存储书籍作者。number是一个int类型的字段，用于存储书籍数量。

| 名 | 类型 |
| --- | --- |
| bookName | varchar |
| bookPrice | int |
| author | varchar |
| number | int |

* **admin表：**

这个数据表名为admin，包含六个字段：adminName、password、age、phoneNumber、account和sex。adminName是一个varchar类型的字段，用于存储管理员名称。password是一个varchar类型的字段，用于存储管理员密码。age是一个int类型的字段，用于存储管理员年龄。phoneNumber是一个varchar类型的字段，用于存储管理员电话号码。account是一个varchar类型的字段，用于存储管理员账户信息。sex是一个varchar类型的字段，用于存储管理员性别。

| 名 | 类型 |
| --- | --- |
| adminName | varchar |
| password | varchar |
| age | int |
| phoneNumber | varchar |
| account | varchar |
| sex | varchar |

* **havebooks表：**

这是一个借书信息存储表，这个数据表名为havebooks，包含六个字段：bookName、bookPrice、author、number、time和account。bookName是一个varchar类型的字段，用于存储书籍名称。bookPrice是一个int类型的字段，用于存储书籍价格。author是一个varchar类型的字段，用于存储书籍作者。number是一个int类型的字段，用于存储书籍数量。time是一个varchar类型的字段，用于存储借书相关时间信息。account是一个varchar类型的字段，用于存储用户或管理员账户信息。该字段通常用于记录借阅者账户信息。

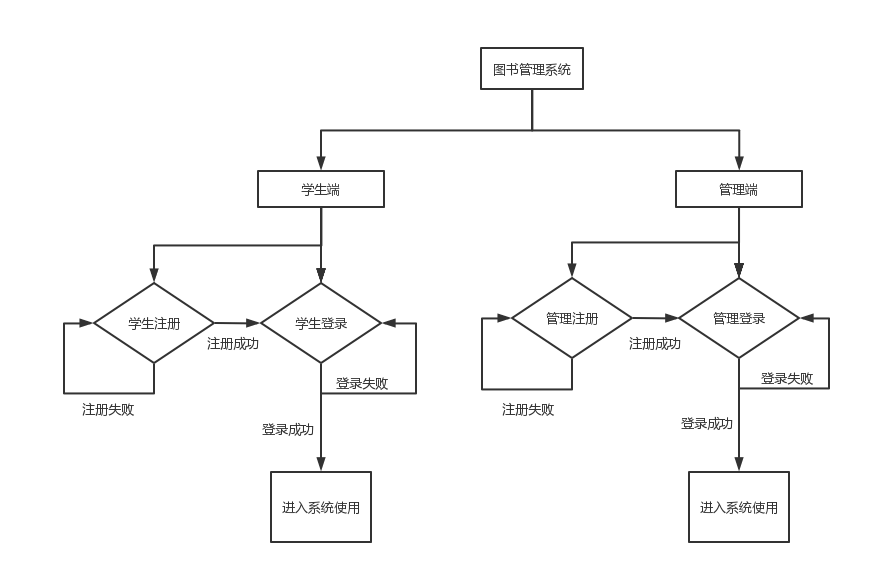
|  |  |
| --- | --- |
| 名 | 类型 |
| bookName | varchar |
| bookPrice | int |
| author | varchar |
| number | int |
| time | varchar |
| account | varchar |

* **student表：**

这个数据表名为student，包含了学生的信息。studentName：这一列表示学生的姓名，数据类型为varchar。studentNumber：这一列表示学生的学号，数据类型为varchar。age：这一列表示学生的年龄，数据类型为int。password：这一列表示学生的账号密码，数据类型为varchar。sex：这一列表示学生的性别，数据类型为varchar。major：这一列表示学生所在的专业，数据类型为varchar。account：这一列表示学生的账号，数据类型为varchar。通过这个表，可以记录学生的基本信息，并且可以根据学生的学号、姓名、专业等信息进行查询和管理。

|  |  |
| --- | --- |
| 名 | 类型 |
| studentName | varchar |
| studentNumber | varchar |
| age | int |
| password | varchar |
| sex | varchar |
| major | varchar |
| account | varchar |

1. **程序流程图：**



1. **关键代码：**

* **登录：**

在登录界面的后端代码中传入参数为账号和密码，返回值为整数类型的结果，其中0表示验证失败，1表示验证成功。在方法中，首先定义了一个默认值为0的result变量表示验证失败。然后通过调用studentMapper对象的selectStudentByAccountAndPassword方法，查询数据库中是否存在对应账号和密码的学生信息。如果查询结果为null，则说明验证失败，将result赋值为0，并输出"验证失败"。否则，验证成功，将result赋值为1，并输出"验证成功"。最后，无论验证成功还是失败，都会将result作为方法的返回值返回给调用者。如果查询语句出现错误，则会抛出异常并打印堆栈信息。

public Integer getStudent(String account, String password) {  
 int result = 0;*//默认验证失败* try{  
 Student student = studentMapper.selectStudentByAccountAndPassword(account,password);  
 if (student == null) {  
 System.*out*.println("验证失败");  
 result = 0;  
 } else {  
 System.*out*.println("验证成功");  
 result = 1;  
 }  
 }catch (Exception e){  
 e.printStackTrace();*//查询语句错误* }  
 return result;  
}

* **注册：**

传入参数为一个Student对象，返回值为整数类型的结果，其中0表示注册失败，1表示注册成功。在方法中，首先定义了一个默认值为0的result变量表示注册失败。然后判断传入的Student对象是否完整，如果有任何属性值为null，则输出"注册失败，请填写信息完整"，将result赋值为0。否则，说明传入的学生信息完整，接下来会查询数据库中是否存在相同的账号信息。首先通过调用adminMapper的selectAll方法，获取所有管理员的信息列表，然后通过调用studentMapper的showAll方法，获取所有学生的信息列表。然后，遍历学生信息列表和管理员信息列表，检查当前传入的学生信息中的账号信息是否与列表中的账号信息重复。如果重复，将result赋值为0。最后，根据result的值确定是否可以将该学生信息添加到数据库中。如果result值为1，则调用studentMapper的addStudent方法添加该学生信息到数据库中，否则输出"注册失败，账号重复"。无论注册成功或失败，都会将result作为方法的返回值返回给调用者。如果查询语句出现错误，则会抛出异常并打印堆栈信息。

public Integer addStudent(Student student) {  
 int result = 0;  
 try{  
 if(student.getStudentName() == null||student.getStudentNumber() == null || student.getAccount()==null || student.getAge()==null ||student.getMajor()==null ||student.getSex()==null||student.getPassword()==null  
 ){  
 System.*out*.println("注册失败，请填写信息完整");  
 result=0;  
 }else{  
 result = 1;  
 List<Admin> admins=adminMapper.selectAll();  
 List<Student> students=studentMapper.showAll();  
 for(Student student1:students){

if(student.getAccount().equals(student1.getAccount())){  
 result=0;  
 }  
 }  
 for(Admin admin1:admins){  
 if(student.getAccount().equals(admin1.getAccount())){  
 result=0;  
 }  
 }  
 if(result==1)  
 studentMapper.addStudent(student);  
 else  
 System.*out*.println("注册失败，账号重复");  
 }  
 }catch (Exception e){  
 e.printStackTrace();*//查询语句错误* }  
 return result;  
}

* **展示所有书本信息：**

该方法使用了Spring MVC框架的Model对象和HttpServletRequest对象。首先，通过调用bookService对象的showAll方法，获取数据库中所有书本的信息列表，保存在一个名为books的List<Book>对象中。然后，将books对象添加到model中，将其命名为"books"。model对象可以将数据传递到前端页面进行展示。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面展示数据。在这里，返回的是"book/selectAllBooks"，表示需要跳转到名为selectAllBooks的前端页面展示所有书本信息。在前端页面中，我们可以通过访问model中的"books"属性来获取所有书本的信息列表，并进行展示。

public String selectAll(Model model, HttpServletRequest request){  
 List<Book> books = bookService.showAll();  
 model.addAttribute("books", books);  
 return "book/selectAllBooks";  
}

* **展示所有学生信息：**

这是一个用于展示所有学生信息的Java方法。该方法使用了Spring MVC框架的Model对象和HttpServletRequest对象。首先，通过调用studentService对象的showAll方法，获取数据库中所有学生的信息列表，保存在一个名为students的List<Student>对象中。然后，将students对象添加到model中，将其命名为"students"。model对象可以将数据传递到前端页面进行展示。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面展示数据。在这里，返回的是"student/selectAllStudents"，表示需要跳转到名为selectAllStudents的前端页面展示所有学生信息。在前端页面中，我们可以通过访问model中的"students"属性来获取所有学生的信息列表，并进行展示。

@RequestMapping("/selectAllStudents")  
public String selectAll(Model model,HttpServletRequest request){  
 List<Student> students = studentService.showAll();  
 model.addAttribute("students", students);  
 return "student/selectAllStudents";  
}

* **展示单个学生信息：**

这是一个用于展示单个学生信息的Java方法。该方法使用了Spring MVC框架的Model对象和HttpServletRequest对象。首先，接收一个名为account1的字符串类型参数，表示需要查询的学生账号。然后通过调用studentService对象的showOne方法，查询指定账号的学生信息。将查询结果保存在一个Student对象中，命名为"student"，并添加到model中。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面展示数据。在这里，返回的是"admin/selectOne"，表示需要跳转到名为selectOne的前端页面展示单个学生信息。在前端页面中，我们可以通过访问model中的"student"属性来获取查询到的单个学生信息，并进行展示。

@RequestMapping("/AdminShowOne")  
public String showOne1(String account1,HttpServletRequest request,Model model){  
 Student student=studentService.showOne(account1);  
 model.addAttribute("student",student);  
 return "admin/selectOne";  
}

* **展示单个书本信息：**

这是一个用于展示单个书本信息的Java方法。该方法使用了Spring MVC框架的Model对象和HttpServletRequest对象。首先，接收一个名为bookName的字符串类型参数，表示需要查询的书本名称。然后通过调用bookMapper对象的showOne方法，查询指定名称的书本信息。将查询结果保存在一个Book对象中，命名为"book"，并添加到model中。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面展示数据。在这里，返回的是"admin/selectOne"，表示需要跳转到名为selectOne的前端页面展示单个书本信息。在前端页面中，我们可以通过访问model中的"book"属性来获取查询到的单个书本信息，并进行展示。

@RequestMapping("/AdminShowOne2")  
public String showOne2(String bookName,Model model,HttpServletRequest request){  
 Book book=bookMapper.showOne(bookName);  
 model.addAttribute("book",book);  
 return "admin/selectOne";  
}

* **借书：**

这是一个用于学生借书的Java方法。该方法使用了Spring MVC框架的HttpServletRequest对象和Model对象。首先，通过HttpServletRequest对象获取从前端页面传来的图书信息参数，包括书名、价格、作者和数量。然后，根据这些参数创建一个新的Book对象，并将书名、价格和作者设置为相应的参数值。接着，判断图书数量是否大于0，如果大于零，则将图书数量减1，并调用bookService对象的updateBook方法更新图书信息。同时，获取当前时间并格式化为指定格式的字符串；再将相关信息填充到HaveBooks对象中，包括时间、书名、价格、作者、数量以及当前登录用户的账号，并调用haveBooksService对象的addBook方法，将借阅信息添加到数据库中。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面。在这里，如果成功借阅，则返回"success/student/borrowBookSuccess"页面，表示借阅成功。否则，返回"success/student/cantBorrow"页面，表示无法借阅。在前端页面中，我们可以通过访问model中的属性来展示借书操作的结果。

@RequestMapping("/borrowBook2")  
public String borrowBook2(HttpServletRequest request,Model model){  
 String bookName = request.getParameter("bookName");  
 int bookPrice = Integer.*parseInt*(request.getParameter("bookPrice"));  
 String author = request.getParameter("author");  
 int number = Integer.*parseInt*(request.getParameter("number"));  
 Book book = new Book();  
 book.setBookName(bookName);  
 book.setBookPrice(bookPrice);  
 book.setAuthor(author);  
 if(number!=0){  
 book.setNumber(number-1);  
 bookService.updateBook(book);  
 LocalDateTime now = LocalDateTime.*now*();  
 DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd--HH：mm：ss");  
 String formattedDateTime = now.format(formatter);  
 HaveBooks haveBooks=new HaveBooks();  
 haveBooks.setTime(formattedDateTime);  
 haveBooks.setBookName(bookName);  
 haveBooks.setBookPrice(bookPrice);  
 haveBooks.setAuthor(author);  
 haveBooks.setNumber(book.getNumber());  
 haveBooks.setAccount(account);  
 haveBooksService.addBook(haveBooks);  
 return "success/student/borrowBookSuccess";  
}  
 book.setNumber(number);  
 return "success/student/cantBorrow";  
}

* **还书：**

这是一个用于学生还书的Java方法。该方法使用了Spring MVC框架的HttpServletRequest对象和Model对象。首先，通过HttpServletRequest对象获取从前端页面传来的图书信息参数，包括书名、价格、作者和数量。然后，根据这些参数创建一个新的Book对象，并将书名、价格、作者和数量设置为相应的参数值。接着，将数量加1，并调用bookService对象的updateBook方法更新图书信息。同时，调用haveBooksService对象的deleteBook方法，删除数据库中对应的借阅信息。最后，方法返回一个字符串类型的结果，表示需要跳转到哪个前端页面。在这里，返回"success/student/returnBookSuccess"页面，表示还书成功。在前端页面中，我们可以通过访问model中的属性来展示还书操作的结果。

@RequestMapping("/returnBook2")  
public String returnBook2(HttpServletRequest request,Model model){  
 String bookName = request.getParameter("bookName");  
 int bookPrice = Integer.*parseInt*(request.getParameter("bookPrice"));  
 String author = request.getParameter("author");  
 int number = Integer.*parseInt*(request.getParameter("number"));  
 Book book = new Book();  
 book.setBookName(bookName);  
 book.setBookPrice(bookPrice);  
 book.setAuthor(author);  
 book.setNumber(number);  
 book.setNumber(number+1);  
 bookService.updateBook(book);  
 haveBooksService.deleteBook(bookName);  
 return "success/student/returnBookSuccess";  
}

1. **实验结果与分析**
2. **图书管理系统**

**学生登录、注册界面：**

在登录注册模块中，当学生输入账号和密码后，系统会进行验证。如果账号和密码其中有一个不正确，则不能登录，并有相应的提示。同时，如果学生没有账号，可以在登录页面进行注册，如果所注册的账号和之前学生或管理员注册的账号重复，则会提示账号重复信息，注册失败，需要重新注册。同时，在注册时，系统可以检测到账号的重复，避免了数据冲突的出现。



**管理登录、注册界面：**

在登录注册模块中，当管理员输入账号和密码后，系统会进行验证。如果账号和密码其中有一个不正确，则不能登录，并有相应的提示。同时，如果学生没有账号，可以在登录页面进行注册，如果所注册的账号和之前学生或管理员注册的账号重复，则会提示账号重复信息，注册失败，需要重新注册。同时，在注册时，系统可以检测到账号的重复，避免了数据冲突的出现。



**学生端系统界面：**

这是一个学生端的主窗口界面，使用了layui框架搭建。整个界面由三个部分组成：头部、左侧导航栏和主体内容区域。头部包含了系统logo、操作选择菜单以及用户信息展示，左侧导航栏包含了多个功能链接，主体内容区域是界面的核心，用于显示各种业务内容。这个界面采用了响应式设计，可以在不同大小的屏幕上自适应地显示。同时，它还使用了一些动画效果和图标，使得界面更加生动。在内容主体区域，欢迎语和一个卡片板块呈现，整个页面看起来简洁明了。左侧导航栏提供了四个链接，分别是开始窗口、查询所有书本信息/借书、还书和查询单个书本信息/借书，方便学生进行借阅操作。在底部固定区域，没有放置任何内容。页面中还使用了一些JavaScript代码，实现了元素渲染、弹出层、时间计算等功能。

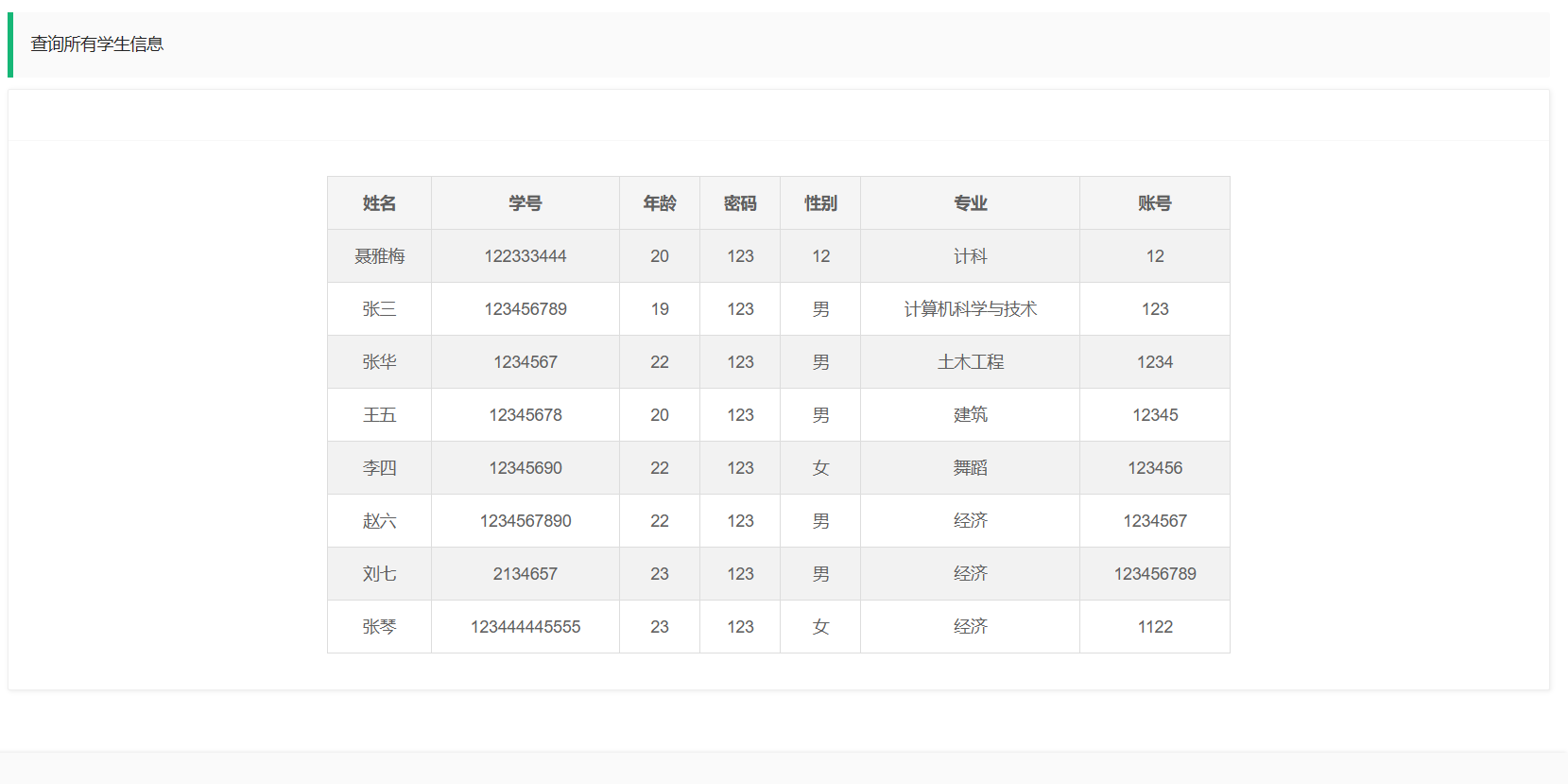


**管理端系统界面：**

这是一个管理端的主窗口界面，使用了layui框架搭建。整个界面由三个部分组成：头部、左侧导航栏和主体内容区域。头部包含了系统logo、操作选择菜单以及用户信息展示，左侧导航栏包含了多个功能链接，主体内容区域是界面的核心，用于显示各种业务内容。这个界面采用了响应式设计，可以在不同大小的屏幕上自适应地显示。同时，它还使用了一些动画效果和图标，使得界面更加生动。在内容主体区域，欢迎语和一个卡片板块呈现，整个页面看起来简洁明了。左侧导航栏提供了六个链接，分别是开始窗口、查询所有书本信息/借书、输入书本信息、新增学生信息、查询所有学生信息和查询单个学生/图书信息，方便管理员进行图书管理和学生信息管理操作。其中，"开始窗口"链接是一个默认的欢迎页，"查询所有书本信息/借书"链接用于查询图书和借阅操作，"输入书本信息"和"新增学生信息"链接分别用于添加新的书本和学生信息，"查询所有学生信息"链接用于查询全部学生信息，"查询单个学生/图书信息"链接用于查询单个学生或图书信息。最后一个链接"还书"用于管理员归还学生的借阅书籍。



**学生信息展示界面：**

这是一个查询所有学生信息的界面，使用了LayUI框架进行页面布局和样式设计。整个界面由头部、左侧导航栏、主体内容区域和底部固定区域四个部分组成。头部包括系统logo、操作选择菜单以及用户信息展示。左侧导航栏提供了多个功能链接，包括开始窗口、查询所有书本信息/借书、输入书本信息、新增学生信息、查询所有学生信息和查询单个学生/图书信息。主体内容区域显示查询结果，展示所有学生的信息，表格呈现方式为列表形式。底部固定区域为空。在具体实现上，该页面使用了JavaServer Pages（JSP）技术和JSTL标签库来动态生成HTML内容。同时，还使用了JavaScript代码和LayUI的相关组件来实现交互效果和样式。  


**所有书本展示界面/借书：**

这是一个查询所有书本信息和借书的界面，使用了LayUI框架进行页面布局和样式设计。整个界面由头部、左侧导航栏、主体内容区域和底部固定区域四个部分组成。头部包括系统logo、操作选择菜单以及用户信息展示。左侧导航栏提供了多个功能链接，包括开始窗口、查询所有书本信息/借书、输入书本信息、新增学生信息、查询所有学生信息和查询单个学生/图书信息。主体内容区域显示查询结果，展示所有书本的信息，表格呈现方式为列表形式。底部固定区域为空。在具体实现上，该页面使用了JavaServer Pages（JSP）技术和JSTL标签库来动态生成HTML内容。同时，还使用了JavaScript代码和LayUI的相关组件来实现交互效果和样式。其中，通过使用forEach标签遍历所有书本信息，并生成相应的表格行。每一行的“借书”、“删除”链接可以实现借阅、删除书本的功能。



**还书界面：**

这是一个 JSP 页面，用于展示已借书本信息的界面。它包含了一个头部区域、左侧导航区域、内容主体区域和底部固定区域。页面使用了 layui 框架提供的样式和 JS 库来实现一些页面效果。在页面上方，有一个操作选择的区域，在左侧导航区域中列举了一些菜单项，包括开始窗口、查询所有书本信息/借书、还书以及查询单个书本信息/借书。右侧则显示了当前登录用户的姓名和一个退出系统的链接。在内容主体区域中，页面展示了一个表格，其中列出了当前用户已经借阅的书本信息。表格的每行显示了借书账户、书名、书本价格、作者、借书时间以及一个“还书”操作按钮。



**新增书本界面：**

这是一个包含HTML和Java代码的web界面，用于输入书本信息。其中的HTML代码创建了一个页面布局和一个表单，而Java代码提供了一些基本的数据处理和逻辑控制功能。这个界面使用了LayUI框架，提供了一些组件和样式来美化页面。此外还包含了一些JavaScript代码，用于验证表单输入和提交表单。总体而言，这个界面功能简单，但是具有实用性。



**新增学生界面：**

这个界面是一个web页面，它由HTML，CSS和JavaScript构成。它使用了LayUI框架来美化页面，并提供了一些组件和样式。在页面的顶部有一个头部区域，包括一个logo和左侧和右侧两个导航条，在左侧的导航栏中以列表形式呈现了不同的操作选项，用户可以通过点击其中的链接来进入对应的页面。在页面的主体区域，用户可以填写学生信息，包括姓名、学号、年龄、专业、性别、账号和密码等信息，并通过提交按钮将数据传递到后端进行处理。此外，页面也包含了JavaScript代码，用于表单输入验证和处理。总体而言，这个界面具有实用性和美观性，可以满足学生信息管理的基本需求。



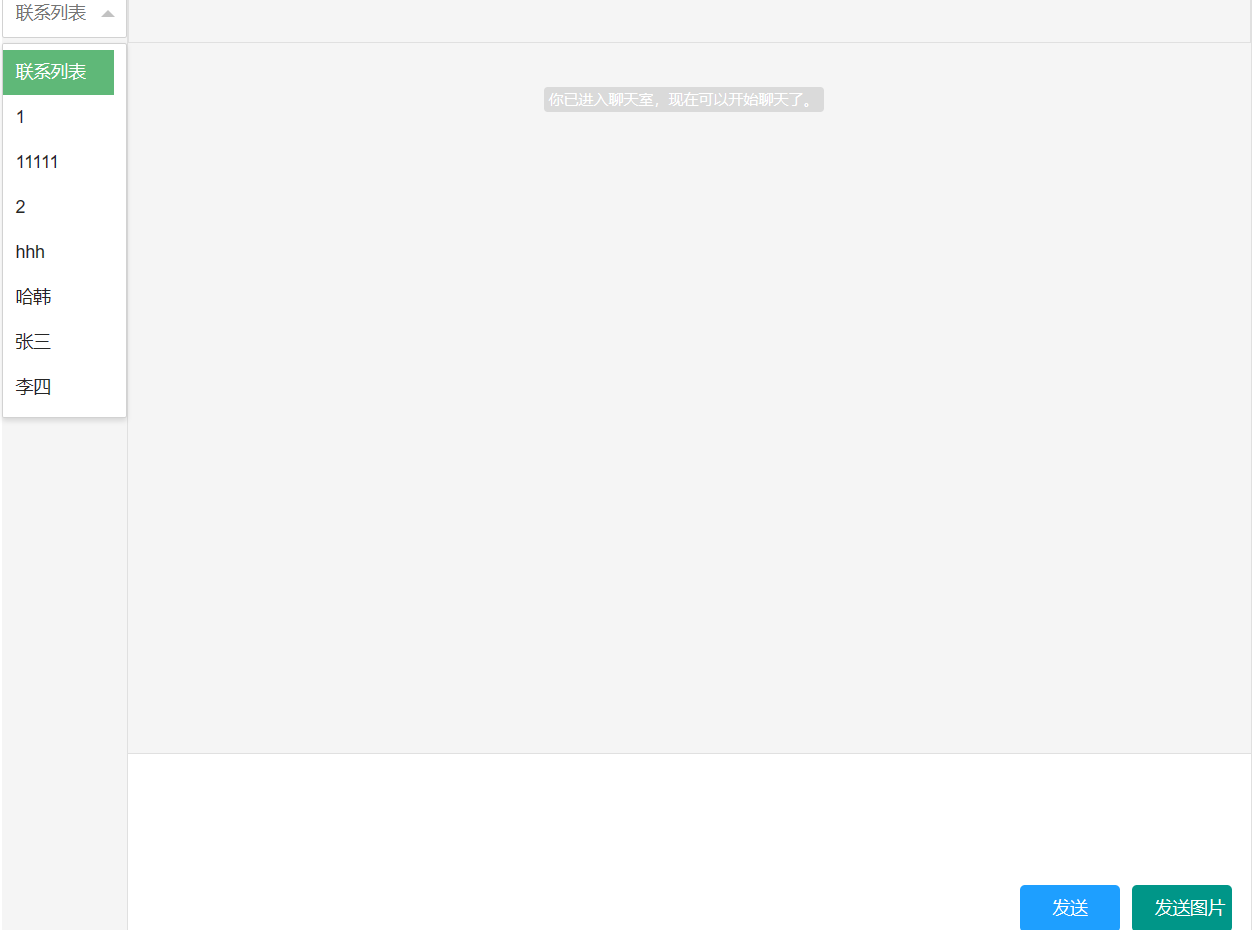
1. **仿QQWeb即时聊天系统：**

**登录注册界面：**



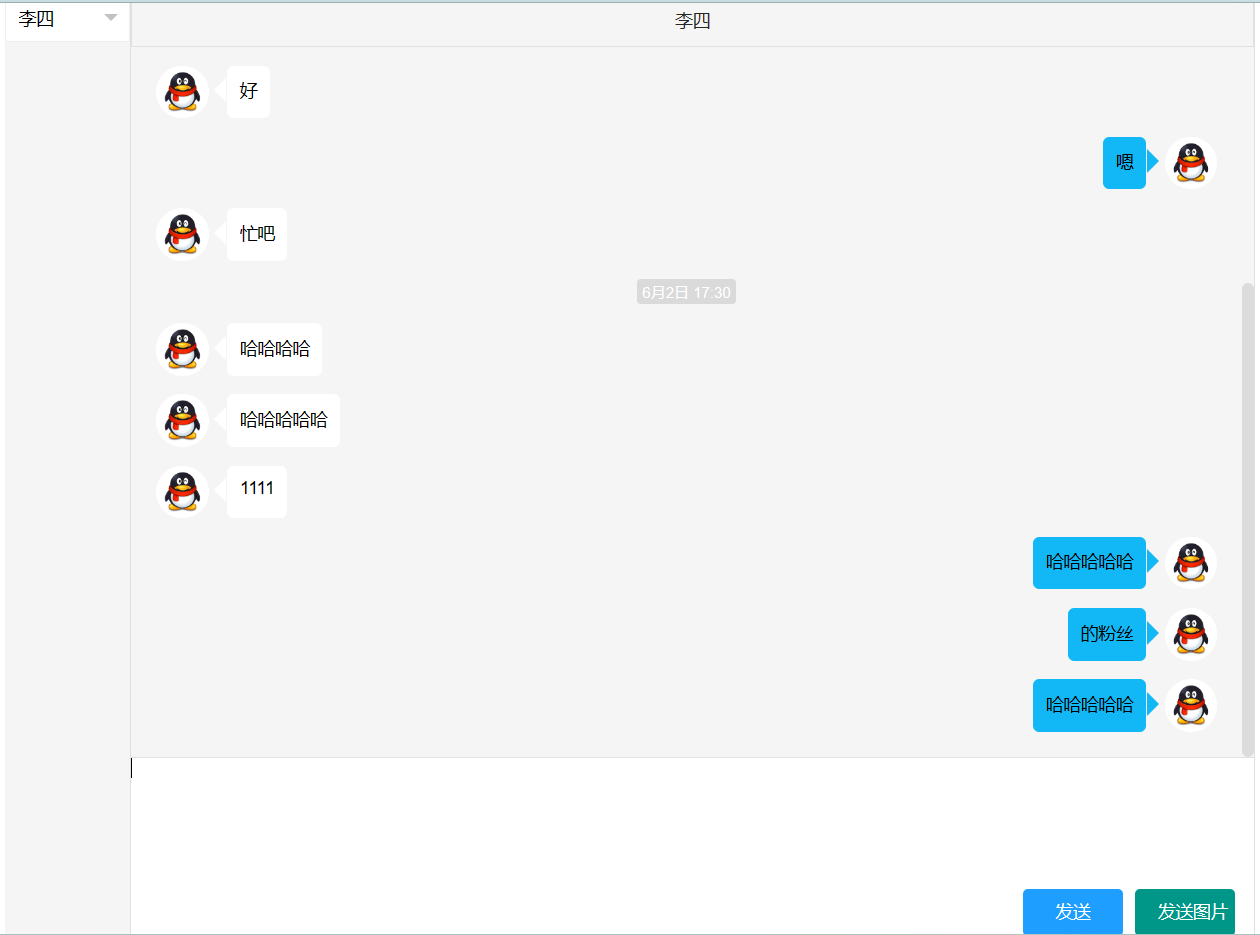
**聊天室界面：**

左上角下拉列表可以进行好友选择进行聊天，在聊天界面顶部可以显示聊天好友名称信息。在聊天框输入信息，点击发送按钮可以进行信息发送，如果聊天框为空发送，则会弹出提示要输入信息的提示。除此之外还可以发送图片。



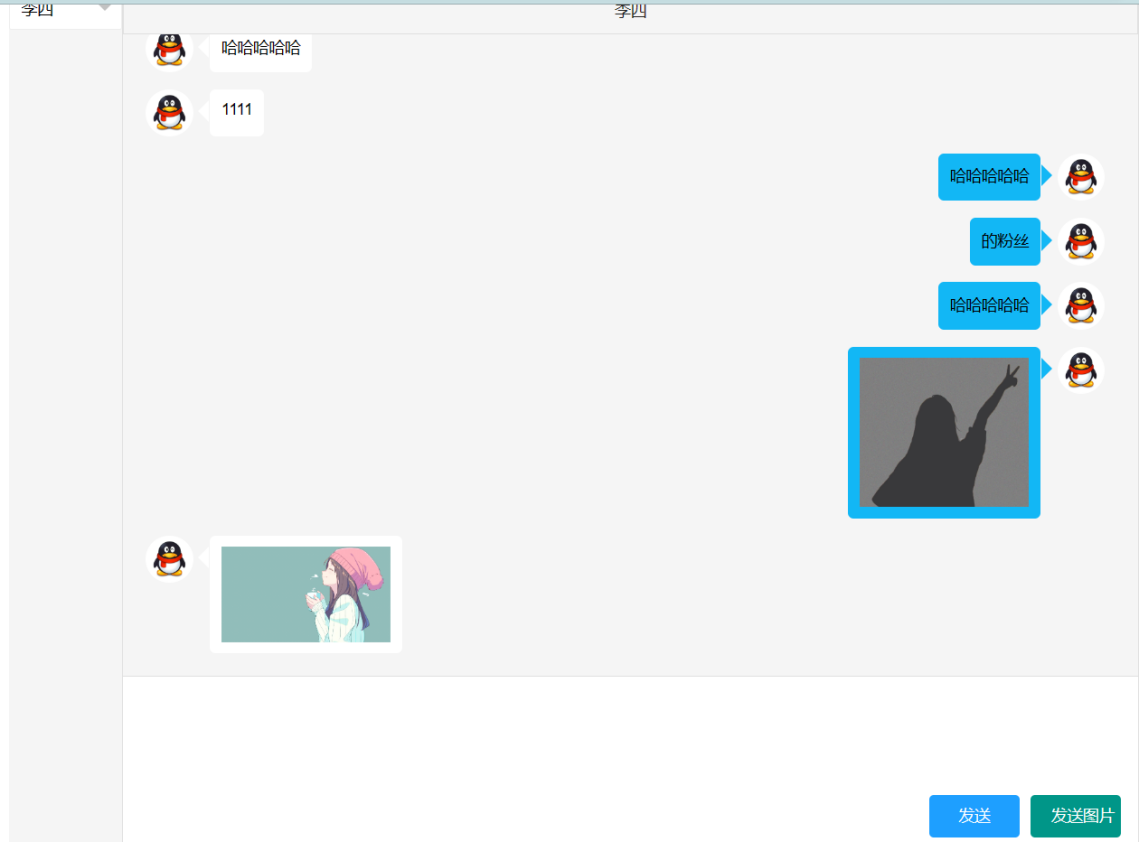
**历史消息的展示：**

在下拉框选择好友后，前端会渲染历史聊天记录，还可以每隔一天显示历史聊天时间。



**图片的发送接收：**

此系统也实现了图片的发送功能。



**六、小结与心得体会**

本次课程设计，在两周时间完成了两个系统，图书管理系统相对于聊天系统比较简单，但是工程量相对于聊天系统要大。

关于图书管理系统，首先，登录注册模块是整个系统的基础。通过该模块，学生和管理员可以分别进入系统并进行相关操作。在实际使用过程中，我发现该模块的验证机制非常严格，有效保障了账号信息的安全性。同时，注册时的账号重复提示也非常及时，避免了因账号冲突造成的不便。其次，管理员管理模块提供了丰富的功能和操作。管理员可以查询所有学生和图书信息，并可以新增学生信息、借阅和归还图书等。尤其是借阅和归还图书时，系统给出了详细的流程和提示，使得操作更加便捷和高效。最后退出系统时，系统也会给出相应的提示，提醒管理员注意信息安全。最后，学生管理模块虽然功能较为简单，但却十分实用。学生可以方便地查询图书资料，并可以借阅和退还图书，同时所借书本信息也可以清晰显示。这些功能的使用，大大提高了学生的阅读效率和体验。总的来说，该图书管理系统具有良好的用户界面和功能设计，操作简单方便，同时又保证了信息的安全性。通过这个系统的设计完成，我不仅巩固了相关知识和技能，还提高了自己的实践能力和综合素质。我相信这份经验和收获会对我的未来学习和工作都有所帮助。

关于仿QQWeb即时聊天系统，成功地实现了一个仿QQWeb即时聊天系统。通过使用WebSocket技术，实现了点对点的文本消息、表情和图片的发送、接收和显示功能。同时，我也实现了本地消息的存储，使得在离线的时候用户可以查看历史消息。在开发过程中，遇到了不少困难和挑战。其中最大的问题是如何实现图片的发送和接收功能。经过不断的尝试和调试，终于找到了一种可行的方法，即将图片转换成二进制流后再进行传输。这样不仅能够保证图片的完整性，还能避免由于跨域请求而导致的安全问题。另外，我也学习到了很多关于WebSocket技术的知识和应用。相比于传统的HTTP协议，WebSocket具有更快的响应速度和更低的带宽占用，能够更好地满足实时通信的需求。同时，WebSocket也支持服务端向客户端推送数据，从而实现了真正意义上的双向通信。通过本次实验，我深刻认识到了Web开发的复杂性和挑战性。但是，只要持之以恒，不断学习和探索，就能够不断突破自我，实现更加出色的作品。