**基于SSM开发框架的学生管理系统设计实现**

1 设计目的

（1）熟悉Spring、Springmvc、Mybatis框架的基本工作原理；

（2）熟悉SSM开发框架的搭建，并使用各个框架进行相应功能模块的开发及系统的整合的流程；

（3）掌握SSM框架进行Javaweb应用系统开发的方法。

2 设计任务及系统功能需求分析

**2.1 设计任务**

设计并实现一个学生管理系统，提供前台与后台服务功能。

前台：学生用户在成功登录后，可实现对学生信息、课程信息、班级信息和排课信息的浏览、查询和导出.xls格式文件，可修改登录密码。

后台：管理员成功登录后，可对图书学生信息、课程信息、班级学习和排课信息进行增删改查和导入导出.xls格式文件操作，可重置用户登录密码。

系统功能结构框图如图1-1所示。

|  |
| --- |
| aImg1-1 |

**2.2 系统功能需求分析**

**2.2.1 前端**

用于提供系统与用户的交互界面，包括用户登录和登录密码的修改，用户可以浏览和查询学生信息、课程信息、班级信息和排课信息，可以导出.xls格式文件。图2-1给出了用户的用例图。

1. 登录模块
2. 对各类信息进行浏览查询
3. 各类信息展示（分页插件pagehelper实现）
4. 导出.xls格式文件
5. 修改登录密码

**2.2.2 后端**

后端主要基于Spring、Springmvc、Mybatis框架，响应前端的各项操作，修改更新MySQL数据库实现各项功能，包括重置用户登录密码，对学生信息、课程信息、班级信息、排课信息和操作记录信息进行增删改查、分页展示和导入导出.xls格式文件操作。图2-2给出了管理员的用例图。

1. 登录验证
2. 对数据库中各类信息进行增删查改
3. 分页插件pagehelper实现信息分页展示功能
4. 导入导出.xls格式文件
5. 重置用户登录密码

|  |  |
| --- | --- |
| 图 2-1 | 图 2-2 |

3 系统设计

**3.1 功能模块设计**

**3.1.1 登录**

在登录模块中，后台根据登录信息判断为学生还是管理人员，并跳转到相应的页面。

**3.1.2 用户管理**

系统的用户分后台管理员用户、学生用户。

（1）后台管理员用户

后台管理员用户可以对该管理模块下的各类信息进行增删改查和导入导出.xls格式文件操作，重置用户登录密码。

1. 学生用户

学生用户可以浏览、检索和导出该模块对应的各类信息，修改登录密码。

**3.1.3 信息分类管理**

页面左侧设有不同信息页的导航栏，用户可通过导航栏跳转不同页面。

学生用户下信息页有学生信息、课程信息、班级信息、排课信息和用户中心页。

管理员用户下信息页有学生信息、课程信息、班级信息、排课信息、操作记录信息和管理员中心页（主要用于重置用户登录密码）。

**3.1.4 信息导入导出**

（1）导入

后台管理员用户可以通过导入.xls格式文件批量增加同类信息，导出相关信息。

（2）导出

学生或管理员可通过导出.xls格式文件下载相关信息。

**3.1.5 信息分页展示**

具体信息页底部有一个分页导航栏，用户可通过底部导航栏实现同类信息不用页面的跳转。

**3.2 数据库设计**

3.2.1数据库E-R图设计

从以上系统功能概述分析可得，本系统所涉及的实体包括：管理员、用户、分类、课程信息、学生信息、班级信息、排课信息、操作记录、页面导航、分页导航。

1. 管理员实体

管理员实体属性包括：管理员id、管理员用户名、密码、年龄、电话、邮箱、类别。管理员实体图如图3-1所示。

1. 用户实体

用户实体属性包括：用户id、用户名、密码、年龄、电话、邮箱、类别。用户实体图如图3-2所示。

|  |  |
| --- | --- |
| 图 3-1 | 图 3-2 |

1. 信息分类实体

信息分类实体属性包括：分类名。信息分类实体图如图3-3所示。

1. 课程信息实体

课程信息实体属性包括：课程号、课程名称、开设学院、上课周次、学分、学时。课程信息实体图如图3-4所示。

|  |  |
| --- | --- |
| 图 3-1 | 图 3-2 |

1. 学生信息实体

学生信息实体属性包括：学号、姓名、学院、专业、班级。学生信息实体图如图3-5所示。

1. 班级信息实体

班级信息实体属性包括：班级id、班级名称、班主任。班级信息实体图如图3-6所示。

|  |  |
| --- | --- |
| 图 3-1 | 图 3-2 |

1. 排课信息实体

排课信息实体属性包括：排课id、课程名称、班级名称、教师名称、上课教室。排课信息实体图如图3-7所示。

1. 页面导航栏实体

页面导航栏实体属性包括：学生信息、课程信息、班级信息、操作记录、管理员中心/用户中心。页面导航栏实体图如图3-8所示。

|  |  |
| --- | --- |
| 图 3-1 | 图 3-2 |

1. 分页导航栏实体

分页导航栏实体属性包括：首页、上一页、具体页号、下一页、末页。分页导航栏实体图如图3-9所示。

|  |  |
| --- | --- |
| 图 3-1 |  |

实体关系图如图3-10所示。

|  |
| --- |
| 图 3-10 |

3.2.2数据库表设计

本项目使用关系型数据库Mysql来完成表的设计以及数据的存储，数据库名为school，共设计了6张表。

（1）t\_user (用户表)

该表用来保存用户信息。

|  |
| --- |
| 表3-1 用户表 |

用户表主要有userid，username，password，age，email，phone，type属性，表示为用户id，用户名，密码，年龄，邮箱，电话，类型。若type值为1则为管理员，若为0则为学生用户，用户信息如表3-1所示。

（2）t\_student (学生信息表)

该表用来保存学生信息。

|  |
| --- |
| 表3-2学生信息表 |

学生信息表主要有stuid，stuname，clazzname，college，profession属性，表示为学号，学生姓名，班级名，学院，专业。学生信息如表3-2所示。

（3）t\_course (课程信息表)

该表用来保存课程信息。

|  |
| --- |
| 表3-3课程信息表 |

课程信息表主要有courseid，coursename，college，week，credit，period属性，表示为课程号，课程名，开设学院，周次，学分，学时。课程信息如表3-3所示。

（4）t\_class (班级信息表)

该表用来保存班级信息。

|  |
| --- |
| 表3-4班级信息表 |

班级信息表主要有clazzid，clazzname，clazztname属性，表示为班级id，班级名，班主任名。班级信息如表3-4所示。

（5）t\_schedule (排课信息表)

该表用来保存排课信息。

|  |
| --- |
| 表3-5排课信息表 |

排课信息表主要有scheduleid，coursename，clazzname，tname，roomname属性，表示为排课id，课程名称，班级名，教师姓名，教室名。排课信息如表3-5所示。

（6）t\_record (操作记录表)

该表用来保存操作记录信息。

|  |
| --- |
| 表3-6操作记录表 |

操作记录表主要有rid，uid，operate，oid，oname，date属性，表示为操作记录id，管理员id，操作描述，操作对象id，操作对象名，操作时间。操作记录如表3-6所示。

4 系统功能实现

4.1 登录功能实现

当用户登录时会校验用户账号和密码,校验时，后台会对用户输入的id进行是否存在的匹配，若不存在则返回3，并且后续不再进行对用户输入密码的校验。若存在，用户输入的密码会被加密并与保存在数据库的该用户的加密密码进行匹配，当账号和密码都正确时，会进一步进行用户类型的匹配，若用户类型为1则为管理员用户，返回1；若为0则为学生用户，返回2。若密码校验不通过，则返回0。由此保证受限资源在用户未登录时不允许被访问，用户信息一并响应给前端。登录的主要逻辑代码如下：

|  |
| --- |
| public int loginType(String userid, String password) {  System.out.println("Service: "+userid+" "+password);  User user =userMapper.existsByUid(userid);  if (user==null) {  System.out.println("Service:3 "+userid+" "+password);  return 3;  }  user = userMapper.findByUidAndPwd(userid, password);  if (user != null && user.getType().equals("1")) {  return 1;  }  else if (user != null && user.getType().equals("0")) {  return 2;  }  else {  System.out.println("Service:0 "+userid+" "+password);  return 0;  }  }  <!-- User findByUidAndPwd(String uid, String pwd);-->  <select id="findByUidAndPwd" resultType="com.zisu.springmvc.pojo.User">  select \* from t\_user where userid = #{uid} and password=#{pwd}  </select>  <!-- boolean existsByUid(@Param("uid") String uid);-->  <select id="existsByUid" resultType="com.zisu.springmvc.pojo.User">  select \* from t\_user where userid = #{uid}  </select> |

用户登录的总体实现流程如图4-2所示。

|  |
| --- |
| 图 4-2 用户登录总体流程图 |

4.2分类信息展示

进入首页或点击导航栏对应页时会向后端发送异步请求，分类信息展示通过请求后端接口，接口调用service层的获取全部学生信息方法，该方法通过调用mapper中获取全部学生信息方法中的sql语句获取全部学生信息，后端采用pageHelper插件将得到的学生信息列表以分页的方式传回给前端。全部学生信息展示的sql语句如下：

（1）获取所有学生信息

<select id="getAllStudent" resultType="com.zisu.springmvc.pojo.Student">

select \* from t\_student

</select>

（2）获取所有课程信息

<select id="getAllCourse" resultType="com.zisu.springmvc.pojo.Course">

select \* from t\_course

</select>

（3）获取所有班级信息

<select id="getAllClass" resultType="com.zisu.springmvc.pojo.Clazz">

select \* from t\_class

</select>

（4）获取所有排课信息

<select id="getAllSchedule" resultType="com.zisu.springmvc.pojo.Schedule">

select \* from t\_schedule

</select>

（5）获取所有操作记录信息

<select id="getAllRecord" resultType="com.zisu.springmvc.pojo.Record">

select \* from t\_record

</select>

（6）获取所有用户信息

<select id="getAllUser" resultType="com.zisu.springmvc.pojo.User">

select \* from t\_user

</select>

4.3分类信息的查询实现

分类信息的查询是指在对应分类信息页搜索框中输入对应分类关键字，点击“搜素”后，前端通过异步请求将对应分类的关键字传入到后端根据对应分类关键字查询对应分类信息接口中，接口根据对应分类关键字查询对应分类信息列表。后端将得到的对应分类信息列表传回给前端。前端会显示查到对应分类信息。按对应分类关键查询对应分类信息的sql语句如下：

（1）根据姓名查询学生信息

@Select("select \* from t\_student where stuname = #{stuname}")

List<Student> selectStudentByName(@Param("stuname") String stuname);

（2）根据学生id查询学生信息

@Select("select \* from t\_student where stuid = #{stuid}")

List<Student> selectStudentById(@Param("stuid") String stuid);

（3）根据课程名查询课程信息

@Select("select \* from t\_course where coursename = #{coursename}")

List<Course> selectCourseByName(@Param("coursename") String coursename);

（4）根据课程id查询课程信息

@Select("select \* from t\_course where courseid = #{courseid}")

List<Course> selectCourseById(@Param("courseid") String courseid);

（5）根据班级名查询班级信息

@Select("select \* from t\_class where clazzname = #{clazzname}")

List<Clazz> selectClassByName(@Param("clazzname") String clazzname);

（6）根据班级id查询班级信息

@Select("select \* from t\_class where clazzid = #{clazzid}")

List<Clazz> selectClassById(@Param("clazzid") String clazzid);

（7）根据班级名查询排课信息

@Select("select \* from t\_schedule where clazzname = #{clazzname}")

List<Schedule> selectScheduleByClazzName(String clazzname);

（8）根据排课id查询学生信息

@Select("select \* from t\_schedule where scheduleid = #{scheduleid}")

List<Schedule> selectScheduleByScheduleId(String scheduleid);

（9）根据操作者id查询操作记录信息

<!-- List<Record> selectRecordByUid(String uid);-->

<select id="selectRecordByUid" resultType="com.zisu.springmvc.pojo.Record">

select \* from t\_record where uid = #{uid}

</select>

4.4分类信息的添加实现

分类信息的增加是指在对应分类信息页点击添加对应分类信息按钮后弹出对应分类信息录入框，前端通过异步请求将在对应分类信息录入框中录入的信息，传入到后端添加对应分类信息的接口中，接口根据传入的对应分类信息添加到对应分类信息列表，并对对应分类消息列表进行更新，后端采用pageHelper插件将得到的对应分类信息列表以分页的方式更新到前端。前端会显示包含新增对应分类信息的全部对应分类列表信息。添加对应分类信息的sql语句如下：

（1）添加学生信息

@Insert("insert into t\_student values (#{stuid},#{stuname},#{clazzname},#{college},#{profession})")

int insertStudent(Student student);

（2）添加课程信息

@Insert("insert into t\_course values (#{courseid},#{coursename},#{college},#{week},#{credit},#{period})")

int insertCourse(Course course);

（3）添加班级信息

@Insert("insert into t\_class values (#{clazzid},#{clazzname},#{clazztname})")

int insertClass(Clazz clazz);

（4）添加排课信息

@Insert("insert into t\_schedule values (null,#{coursename},#{clazzname},#{tname},#{roomname})")

int insertSchedule(Schedule schedule);

4.5分类信息的删除实现

分类信息的删除是指在对应分类信息页点击具体分类信息中的删除按钮后，前端通过异步请求将点击删除按钮所对应信息的id传入到后端中删除对应分类信息中该id所对应信息的接口中，接口根据传入的id删除对应的信息，并对对应分类信息列表进行更新，后端采用pageHelper插件将删除信息后得到的对应分类信息列表以分页的方式更新到前端。前端会显示执行删除操作后的对应分类列表信息。删除对应分类信息的sql语句如下：

（1）删除学生信息

@Delete("delete from t\_student where stuid = #{stuid}")

int deleteStudentById(@Param("stuid") String stuid);

（2）删除课程信息

@Delete("delete from t\_course where courseid = #{courseid}")

int deleteCourseById(@Param("courseid") String courseid);

（3）删除班级信息

@Delete("delete from t\_class where clazzid = #{clazzid}")

int deleteClassById(@Param("clazzid") String clazzid);

（4）删除排课信息

@Delete("delete from t\_schedule where scheduleid = #{scheduleid}")

int deleteStudentById(String scheduleid);

4.6分类信息的更新实现

分类信息的更新是指在对应分类信息页点击具体分类信息中的修改按钮后，前端通过异步请求将点击修改按钮所对应信息的id传入到后端中查询对应分类信息中该id所对应信息的接口中，接口根据传入的id查询到对应信息后；将查询到的信息返回到点击修改按钮后的信息录入弹出框中，以便用户在原有信息基础上进行对应分类信息的修改。用户修改信息完毕点击确认按钮后，前端再次通过异步请求将修改信息框中录入的信息传入到后端更新对应分类信息的接口中，接口根据传入的对于分类信息数据更新到对应信息并对对应分类信息列表进行更新，后端采用pageHelper插件将更新信息后得到的对应分类信息列表以分页的方式更新到前端。前端会显示执行更新操作后的对应分类列表信息。更新对应分类信息的sql语句如下：

（1）更新学生信息

@Update("update t\_student set stuname=#{stuname},college=#{college},profession=#{profession} where stuid = #{stuid}")

int updateStudent(Student student);

（2）更新课程信息

@Update("update t\_course set college=#{college} where courseid = #{courseid}")

int updateCourseTidById(Course course);

（3）更新班级信息

@Update("update t\_class set clazzname=#{clazzname} where clazzid= #{clazzid}")

int updateClassNameById(Clazz clazz);

（4）更新排课信息

@Update("update t\_schedule set coursename=#{coursename},clazzname=#{clazzname},tname=#{tname},roomname=#{roomname} where scheduleid = #{scheduleid}")

int updateSchedule(Schedule schedule);

4.7分类信息的批量导入实现

分类信息的批量导入是指在对应分类信息页选择文件并点击导入按钮后，前端通过异步请求将点击导入按钮所对应文件信息传入到后端导入文件信息所对应的接口中，接口根据传入的文件；将文件内容批量添加到对应分类信息中，后端采用pageHelper插件将更新信息后得到的对应分类信息列表以分页的方式更新到前端。前端会显示执行导入文件操作后的对应分类列表信息。导入文件操作的主要逻辑代码如下（以导入课程信息为例，其余分类信息同理）：

|  |
| --- |
| $('#importCourse').click(function () {  let formData = new FormData();  //上传文件的名称  let name = $("#xlsfile").val();  //数据  formData.append("xlsfile", $("#xlsfile")[0].files[0]);  formData.append("name", name);  $.ajax({  type: "post",  url: "http://localhost:8080/importCourse",  data:formData,  async: false,  cache: false,  contentType: false,  processData: false,  success: function (res) {  console.log(res);  window.location.reload();  },  fail: function (e) {  console.log(e);  window.location.reload();  }  });  });  @RequestMapping(value="/importCourse")  @ResponseBody  public int handleExcelupload(MultipartFile xlsfile, HttpSession session) throws IOException {  String filename=xlsfile.getOriginalFilename();  ServletContext servletContext=session.getServletContext();  String uploadPath=servletContext.getRealPath("")+"upload";  System.out.println("uploadPath-->"+uploadPath);  File file=new File(uploadPath);  if(!file.exists())  file.mkdir();  String finalPath=uploadPath+file.separator+filename;  System.out.println("file.separator-->"+file.separator);  System.out.println("filename-->"+filename);  System.out.println("finalPath-->"+finalPath);  xlsfile.transferTo(new File(finalPath));  courseService.InsertbatchCourses(finalPath);  return 1;  }  @Override  public void InsertbatchCourses(String finalPath) {  List<Course> courseList = excel2course(new File(finalPath));  courseMapper.InsertbatchCourses(courseList);  }  <insert id="InsertbatchStudents">  insert into t\_student values  <foreach collection="students" item="student" separator=",">  (#{student.stuid},#{student.stuname},#{student.clazzname},#{student.college},#{student.profession})  </foreach>  </insert>  private List<Course> excel2course(File xlsfile) {  List<Course> courseList = new ArrayList<Course>();  Course course = null;  try {  POIFSFileSystem fs = new POIFSFileSystem(new FileInputStream(xlsfile));  HSSFWorkbook workbook = new HSSFWorkbook(fs);  //获取第一个sheet页  CellType cellType ;  HSSFSheet sheet = workbook.getSheetAt(0);  if (sheet != null) {  for (int rowNumber = 1; rowNumber < sheet.getLastRowNum(); rowNumber++) {  HSSFRow row = sheet.getRow(rowNumber);  if (row == null) {  continue;  }  course = new Course();  course.setCourseid(convertCellValueToString(row.getCell(0)));  course.setCoursename(convertCellValueToString(row.getCell(1)));  course.setCollege(convertCellValueToString(row.getCell(2)));  course.setWeek(convertCellValueToString(row.getCell(3)));  course.setCredit(Integer.parseInt(convertCellValueToString(row.getCell(4))));  course.setPeriod(Integer.parseInt(convertCellValueToString(row.getCell(5))));  courseList.add(course);  }  fs.close();  }  } catch (IOException e) {  e.printStackTrace();  }  return courseList;  }  private static String convertCellValueToString(Cell cell) {  if (null == cell) {  return null;  }  String returnValue = null;  switch (cell.getCellType()) {  case STRING: //字符串  returnValue = cell.getStringCellValue();  break;  case NUMERIC: //数字  double numericCellValue = cell.getNumericCellValue();  //清除格式，单元格不设置数字格式  //科学计数法的数字全部显示  /\*cell.setCellStyle(null);  returnValue = Double.toString(numericCellValue);\*/  boolean isInteger = isIntegerForDouble(numericCellValue);  if (isInteger) {  DecimalFormat df = new DecimalFormat("0");  returnValue = df.format(numericCellValue);  } else {  returnValue = Double.toString(numericCellValue);  }  break;  case BOOLEAN: //布尔  boolean booleanCellValue = cell.getBooleanCellValue();  returnValue = Boolean.toString(booleanCellValue);  break;  case BLANK: //空值  break;  case FORMULA: //公式  cell.getCellFormula();  break;  case ERROR: //故障  break;  default:  break;  }  return returnValue;  }  public static boolean isIntegerForDouble(Double num) {  double eqs = 1e-10; //精度范围  return num - Math.floor(num) < eqs;  } |

4.8分类信息的导出实现

分类信息的导出是指在对应分类信息页点击导出按钮后，前端通过异步请求将点击导出按钮所对应分类信息页的信息传入到后端导出文件信息所对应的接口中，接口根据点击按钮所在信息页的信息；将信息内容添加到文件中，并以.xls格式文件响应浏览器进行下载。导出文件操作的主要逻辑代码如下（以导出课程信息为例，其余分类信息同理）：

|  |
| --- |
| $('#exportCourse').click(function () {  window.location.href = 'http://localhost:8080/exportCourse';  });  @RequestMapping("/exportCourse")  public ResponseEntity<byte[]> handleExceldownload(HttpServletRequest request)throws IOException{  String filepath=request.getServletContext().getRealPath("/")+"download";  File file=new File(filepath);  if(!file.exists())  file.mkdir();  courseService.saveasexcel(filepath,"courseinfo.xls");  String finalpath=filepath+File.separator+"courseinfo.xls";  System.out.println("finalpath===="+finalpath);  InputStream is=new FileInputStream(finalpath);  byte[] bytes=new byte[is.available()];  is.read(bytes);  MultiValueMap<String,String> headers=new HttpHeaders();  headers.add("Content-Disposition","attachment;filename=courseinfo.xls");  HttpStatus statusCode=HttpStatus.OK;  ResponseEntity<byte[]> responseEntity=new ResponseEntity<byte[]>(bytes,headers,statusCode);  is.close();  return responseEntity;  }  @Override  public void saveasexcel(String finalPath,String filename) throws IOException {  List<Course> courseList = getAllCourse();  // 第一步，创建一个HSSFWorkbook，对应一个Excel文件  Workbook workbook = new HSSFWorkbook();  // 第二步，在workbook中添加一个sheet,对应Excel文件中的sheet  HSSFSheet sheet = ((HSSFWorkbook) workbook).createSheet("courseinfo");  // 第三步，在sheet中添加表头第0行,注意老版本poi对Excel的行数列数有限制  HSSFRow row = sheet.createRow(0);  // 第四步，创建单元格，并设置值表头 设置表头居中  HSSFCellStyle style = ((HSSFWorkbook) workbook).createCellStyle();  style.setAlignment(HorizontalAlignment.CENTER\_SELECTION);  //style.setAlignment("ALIGN\_CENTER"); // 创建一个居中格式  //声明列对象  HSSFCell cell11 = row.createCell(0);  cell11.setCellValue("课程id");  HSSFCell cell12 = row.createCell(1);  cell12.setCellValue("课程名称");  HSSFCell cell13 = row.createCell(2);  cell13.setCellValue("开设学院");  HSSFCell cell14 = row.createCell(3);  cell14.setCellValue("周次");  HSSFCell cell15 = row.createCell(4);  cell15.setCellValue("学分");  HSSFCell cell16 = row.createCell(5);  cell16.setCellValue("学时");  //将list记录写入excel文件  int rows = 1;  for (Course course:courseList) {  //写入每一条记录  row = sheet.createRow(rows);  row.createCell(0).setCellValue(course.getCourseid());  row.createCell(1).setCellValue(course.getCoursename());  row.createCell(2).setCellValue(course.getCollege());  row.createCell(3).setCellValue(course.getWeek());  row.createCell(4).setCellValue(course.getCredit());  row.createCell(5).setCellValue(course.getPeriod());  rows++;  }  FileOutputStream os = new FileOutputStream(finalPath+File.separator+filename);  ((HSSFWorkbook) workbook).write(os);  os.close();  } |

4.9信息的分页展示实现

分类信息的分页展示是指通过后端设置显示页数和页码在对应分类信息页底部显示对应分页导航栏，用户通过点击分页导航栏上的不同按键可跳转到相应页号对应的信息页，分页展示信息的主要逻辑代码如下（以分页展示课程信息为例，其余分类信息同理）：

|  |
| --- |
| <plugins>  <!--配置分页插件-->  <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>  </plugins>  @RequestMapping(value = "/getAllCourse/{pageNum}")  @ResponseBody  public PageInfo<Course> getCoursePage(@PathVariable("pageNum") Integer pageNum){  PageInfo<Course> page = courseService.getCoursePage(pageNum);  return page;  }  @Override  public PageInfo<Course> getCoursePage(Integer pageNum) {  //开启分页功能  PageHelper.startPage(pageNum, 6);  //查询所有的学生信息  List<Course> courseList= courseMapper.getAllCourse();  //获取分页相关数据  PageInfo<Course> page = new PageInfo<>(courseList, 3);  return page;  } |

5 系统整合及测试

本章将对系统实现的功能进行测试，检查功能是否符合预期，是否有足够的健壮性。

5.1登录功能测试

测试条件：用户在未登入状态下点击首页的登录后的页面提示。

测试功能描述：用户在登录页面输入用户名和密码，用户名不存在时，弹出账号不存在提示；密码输入错误时，弹出密码错误提示；密码正确且验证为学生用户时，弹出该账号为学生账号提示；密码正确且验证为管理员时，弹出该账号为管理员账号提示。

测试结果：当用户输入账号不存在时，页面如图5-1（a)所示；当密码输入错误时，页面如图5-1（b)所示；当用户名和密码都输入正确且验证为学生用户时，页面如图5-1（c)所示，后登录成功并自动跳转到首页；当用户名和密码都输入正确且验证为管理员用户时，页面如图5-1（d)所示，后登录成功并自动跳转到首页；

测试结果表明，该模块符合功能需求。

|  |  |
| --- | --- |
| a)账号不存在 | b)密码错误 |
| c)学生账号成功登录 | d)管理员账户成功登录 |
| 图5-1 用户登录 | |

5.2页面初次展示测试

测试条件：用户在登录成功后，页面的跳转。

测试功能描述：学生用户成功登录后页面跳转到学生用户视图下的页面，并在首页上显示用户名；管理员用户成功登录后页面跳转到管理员用户视图下的页面，并在首页上显示管理员和用户名。

测试结果：管理员用户成功登录后，页面如图5-2（a)所示，在首页上显示管理员标识和管理员用户名,页面中有增、删、改、导入等管理员特有权限的响应按钮；学生用户成功登录后，页面如图5-2（b)所示，在首页上显示用户名，页面无增、删、改、导入等管理员特有权限的响应按钮，有查询和导出响应按钮。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| a)管理员登录成功后页面初次展示 |
| b)用户登录成功后页面初次展示 |
| 图5-2 页面初次展示 |

5.3分类信息查询功能测试

测试条件：管理员和学生用户在对应分类信息页顶部的搜索栏中输入关键字进行查询。

测试功能描述：管理员和学生用户输入关键字点击查询后，下侧的信息展示部分展示相应的信息。

测试结果：管理员用户查询学生信息，页面如图5-3（a)所示，在信息展示部分展示对应学生信息；学生用户查询学生信息，页面如图5-3（b)所示，在信息展示部分展示对应学生信息；管理员用户查询课程信息，页面如图5-3（c)所示，在信息展示部分展示对应课程信息；学生用户查询课程信息，页面如图5-3（d)所示，在信息展示部分展示对应课程信息；管理员用户查询班级信息，页面如图5-3（e)所示，在信息展示部分展示对应班级信息；学生用户查询班级信息，页面如图5-3（f)所示，在信息展示部分展示对应班级信息；管理员用户查询排课信息，页面如图5-3（g)所示，在信息展示部分展示对应排课信息；学生用户查询排课信息，页面如图5-3（h)所示，在信息展示部分展示对应排课信息。管理员用户查询操作记录信息，页面如图5-3（i)所示，在信息展示部分展示对应操作记录信息。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| a)管理员-学生信息查询 |
| b)用户-学生信息查询 | |
| c)管理员-课程信息查询 | |
| d)用户-课程信息查询 | |
| e)管理员-班级信息查询 | |
| f)用户-班级信息查询 | |
| g)管理员-排课信息查询 | |
| h)用户-排课信息查询 | |
| i)管理员-操作记录查询 | |
| 图5-3 分类信息查询 | |

5.4分类信息添加功能测试

测试条件：管理员在对应分类信息页顶部点击添加对应分类信息按钮。

测试功能描述：管理员点击添加对应分类信息按钮后，弹出对应分类信息的信息录入框，管理员完成信息录入后点击确认按钮提交，即完成添加录入的信息；若录入的某项信息为空则弹出非空的信息框提示用户输入非空信息项。

测试结果：管理员添加学生信息，页面如图5-4（a)所示，若输入某信息项为空，则弹出提示，成功添加后，在信息展示末页展示对应学生信息；管理员添加课程信息，页面如图5-4（b)所示，若输入某信息项为空，则弹出提示，成功添加后，在信息展示末页展示对应课程信息；管理员添加班级信息，页面如图5-4（c)所示，若输入某信息项为空，则弹出提示，成功添加后，在信息展示末页展示对应班级信息；管理员添加排课信息，页面如图5-4（c)所示，若输入某信息项为空，则弹出提示，成功添加后，在信息展示末页展示对应排课信息。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| a)管理员-学生信息添加 |
| b)管理员-课程信息添加 |
| c)管理员-班级信息添加 |
| d)管理员-排课信息添加 |
| 图5-4 分类信息添加 |

5.5分类信息删除功能测试

测试条件：管理员在对应分类信息页的具体信息项点击删除按钮。

测试功能描述：管理员点击删除按钮后，该项信息删除，对应分类信息页刷新，不再有该项信息。

测试结果：管理员删除各类具体信息后，在操作记录页，页面如图5-5所示。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| 图5-5 分类信息删除 |

5.6分类信息修改功能测试

测试条件：管理员在对应分类信息页的具体信息项点击修改按钮。

测试功能描述：管理员点击具体信息项修改按钮后，弹出对应信息项的信息修改框，管理员完成信息修改后点击确认按钮提交，即完成修改该项信息；若修改的某项信息为空则弹出非空的信息框提示用户输入非空信息。

测试结果：管理员修改学生信息，页面如图5-6（a)所示，若输入某信息项为空，则弹出提示，成功修改后，对应信息项更新显示修改后的信息；管理员修改课程信息，页面如图5-6（b)所示，若输入某信息项为空，则弹出提示，成功修改后，对应信息项更新显示修改后的信息；管理员修改班级信息，页面如图5-6（c)所示，若输入某信息项为空，则弹出提示，成功修改后，对应信息项更新显示修改后的信息；管理员修改排课信息，页面如图5-6（d)所示，若输入某信息项为空，则弹出提示，成功修改后，对应信息项更新显示修改后的信息；

测试结果表明，该模块符合功能需求。

|  |
| --- |
| a)管理员-学生信息修改 |
| b)管理员-课程信息修改 |
| c)管理员-班级信息修改 |
| d)管理员-排课信息修改 |
| e)管理员-操作记录页记录 |
| 图5-6 分类信息修改 |

5.7分类信息导入功能测试

测试条件：管理员在对应分类信息页顶部选择要导入的.xls文件后点击导入按钮。

测试功能描述：管理员点击导入按钮后，按对应分类信息中service定义的导入规则，将.xls格式文件中的内容追加到该类信息列表中。

测试结果：清空对应分类信息项后，页面如图5-7（a)所示，选择文件，执行导入操作后，页面如图5-7（b)所示，文件中的信息以追加到对应分类信息列表(以学生信息页为例，其余类别信息经测试均可实现文件导入)。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| 1. 管理员-导入文件前信息页   b) 管理员-导入文件后信息页 |
| 图5-7 分类信息导入 |

5.8分类信息导出功能测试

测试条件：管理员在对应分类信息页顶部点击导出按钮。

测试功能描述：管理员点击导出按钮后，将对应分类信息页中的信息以.xls格式文件导出下载。

测试结果：点击导出按钮后，页面如图5-8（a)所示，文件下载成功后，打开导出的文件，页面如图5-8（b)所示(以学生信息页为例，其余类别信息经测试均可实现信息导出)。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| 1. 管理员-导出显示   b) 管理员-导出文件内容 |
| 图5-8 分类信息导出 |

5.9信息分页展示功能测试

测试条件：管理员在对应分类信息页底部点击分页导航栏。

测试结果：管理员点击分页导航栏后，页面对应分类信息跳转至对应的页号，页面如图5-9所示。

测试结果表明，该模块符合功能需求。

|  |
| --- |
| 图5-9 信息分页展示 |

6 总结与展望

6.1总结

本文大致可分为6个模块：分别为设计目的、设计任务及系统功能需求分析、系统设计、系统功能实现、系统整合及测试以及总结。其中设计目的部分主要是对该课程设计所期望达到的目标阐述，设计任务部分给出了系统功能结构图，更为直观的展示了项目最终实现的效果，用到的一些前后端框架技术。系统功能需求分析前端和后端两个方面对项目进行了整体分析，并给出管理员和学生用户的用例图。系统设计部分通过分析项目中涉及的实体对象给出了实体图及数据库表中字段对应的实体类。系统功能实现部分通过流程图更加直观的对不同类型用户进行用户分析，按模块进行功能需求分析并详细介绍了系统的功能具体实现，其中包括大致的实现的思路以及代码等。系统整合及测试部分对项目的一个整体结构进行阐述，以及对各功能的实现效果进行测试。本项目还有些许不足之处，例如系统功能还不够完善，数据校验部分还有较大的提升空间。

6.2感悟及展望

在完成本项目期间，系统回顾了本学期老师课上讲解的内容并应用到项目中去，如文件的上传与下载、信息的分页显示等，加深了对本学期所学知识的理解；此外，由于有限的前端知识，拉长了本项目设计实现的时间线，不过依靠CSDN等网站弥补了这个缺点，也解决了许多问题；最后，针对本项目的不足，后续还可以进一步完善，如学生选课、成绩录入以及一些更为复杂的功能实现。