湖南科技大学计算机科学与工程学院

综合实践能力创新实训（1） 课程设计报告

**专业班级：** 23物联网工程一班

**姓 名：** 曹烨

**学 号：** 2305040107

**指导教师：** 刘毅志

**时 间**： 2025.05.26-2025.06.06

**地 点**： 逸夫楼426

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

一、实验题目  
**学生成绩管理系统**

二、实验目的  
WEB应用技术课程设计是计算机软件工程专业的一个综合性实践教学环节，是学习完《高级WEB技术》课程后进行的一次全面的综合练习。其目的在于促进学生复习和巩固计算机软件设计知识，加深对软件设计方法、软件设计技术和设计思想的理解，并能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，掌握软件设计的方法和面向对象程序设计的基本技术，提高学生在软件开发方面的综合应用能力。

三、总体设计（含背景知识或基本原理与算法、或模块介绍、设  
计步骤等）

## 1. 背景知识与基本原理

前后端分离架构：

前后端分离是现代Web开发的一种常见架构，将用户界面与业务逻辑、数据处理分离开来，提高代码的可维护性和开发效率。前端通常使用HTML、CSS和JavaScript构建，后端使用各种编程语言和框架（如Java、Node.js、Go等）处理业务逻辑和数据库交互。

MVC设计模式：

MVC设计模式（模型-视图-控制器）是一种软件架构范式，旨在通过职责分离提升代码可维护性。该模式将应用程序划分为三个核心组件：模型（Model）负责封装业务逻辑和数据操作，例如学生成绩管理系统中计算平均分的算法和数据库交互；视图（View）专注于用户界面呈现，如HTML页面或Vue.js组件展示成绩表格；控制器（Controller）则充当协调者，接收用户从浏览器提交的请求（如成绩查询指令），调用对应模型处理业务逻辑，再选择合适的视图渲染结果返回给用户。

通义千问AI大模型：

通义千问（Qwen）是阿里云推出的通用语言大模型系列，基于Transformer架构构建，采用大规模预训练+指令微调的技术路线，具备强大的语言理解与生成能力。其训练语料涵盖中英文网页、书籍、代码等多领域数据，模型通过自回归方式学习语言分布，并具备上下文语义建模、多轮对话、文本生成、逻辑推理等能力。通义千问支持以“系统指令 + 用户输入”的消息格式进行交互，结合API调用接口，可广泛应用于问答、写作辅助、信息抽取、代码生成等自然语言处理场景，是当前人工智能自然语言生成技术的重要代表之一。

## 2. 模块介绍

整个学生成绩管理系统可以划分为以下几个主要模块：

* 学生管理模块
  + 负责管理学生的相关信息，如学号、姓名、性别、出生年月日等
  + 实现学生表的增删改查等功能，不能修改学号
  + 通过学院号、专业号、班级号等生成对应的学号，使学号唯一
* 成绩管理模块
  + 提供新增学生成绩的功能，且可完成增删改查的功能
  + 可通过学生成绩表，生成报表展示，包含平均成绩、该科目班级平均值，总成绩以及班级总成绩平均值
  + 根据单个学生生成成绩分布柱状图
  + 通过筛选条件调用AI大模型进行数据分析
* 用户登录模块
  + 管理员可进行用户管理，给其他老师分配账号
* 前端界面模块
  + 提供友好的管理系统页面，包含菜单、表单、图表等
  + 实现信息的实时更新和显示

## 3. 设计步骤：

### 3.1 系统功能需求

* 添加学生信息（自动生成唯一学号）
* 添加学生成绩（数学、Java、英语、体育）
* 学号精确查询成绩
* 姓名模糊查询成绩（支持多条结果）
* 绘制单个学生各科成绩柱状图
* 学生信息修改与删除（学号不可更改）
* 生成包含平均成绩的学习情况报表，并导出为Excel
* 调用AI大模型，实现成绩数据的分析与总结

### 3.2 技术选型

* 前端框架：Vue 3 + Element Plus + Axios
* 后端框架：Spring Boot + MyBatis
* 数据库：MySQL
* 工具与依赖：IDEA、DataGrip、Apifox、VSCode、ECharts、EasyExcel
* 部署服务器：Tomcat、Nginx
* AI大模型：通义千问-Turbo

### 3.3 数据库设计

* 设计用户表、学生表、课程表、成绩表、学院表、专业表等
* 学生表包含学号、入学年份、学院代码、专业代码、班级编号、姓名等
* 成绩表包含编号、学号、课程号、成绩
* 学院表包含学院代码、学院名称

### 前端开发

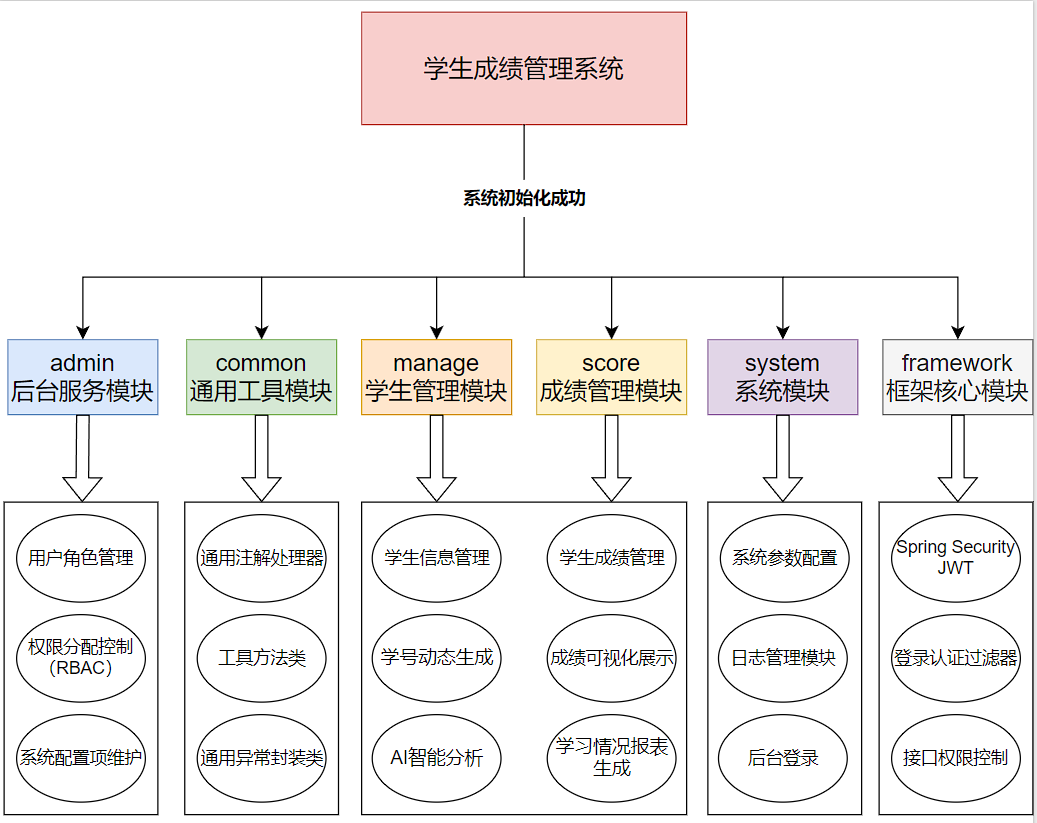
* 前端采用基本vue框架，设计管理系统界面
* 实现与前后端的对接
* 采用Echart组件生成柱状图图表

### 3.5 后端开发

* 实现用户管理功能
* 实现学生数据和成绩数据的管理，并可通过大模型对成绩进行分析总结
* 采用了EasyExcel生成Excel成绩报表

四、详细设计（含主要的数据结构、程序流程图、关键代码等）

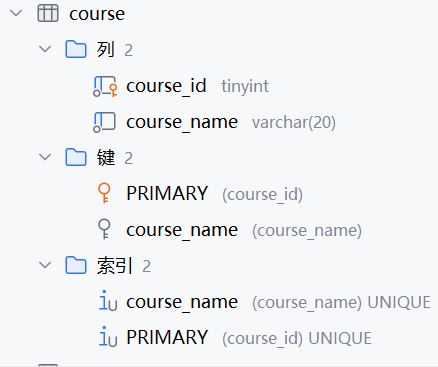
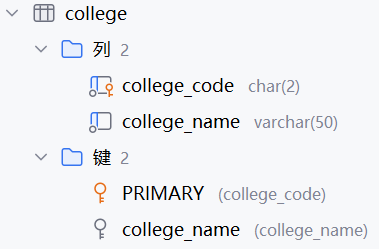
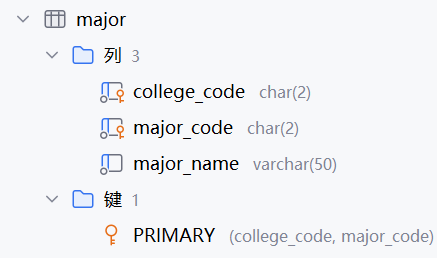
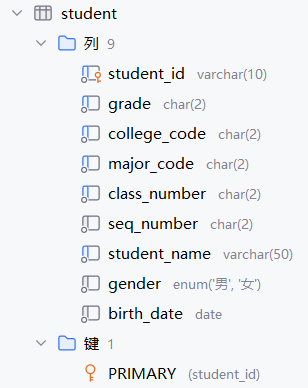
系统流程图：



主要部分为学生管理模块、成绩管理模块两部分

## 1. 数据库核心表结构说明

表数据结构如下：



学号根据“年级 + 学院代码 + 专业代码 + 班级编号 + 班内序号”生成，其中学号不可改变，更改对应的属性可模拟转专业，学号作为主键保持不变

学生表包含学号、年级、学院代码、专业代码、班级编号、班内序号、姓名、性别、出生日期

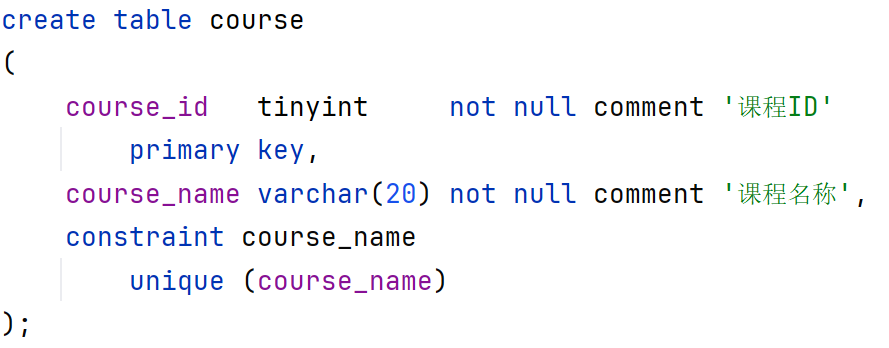
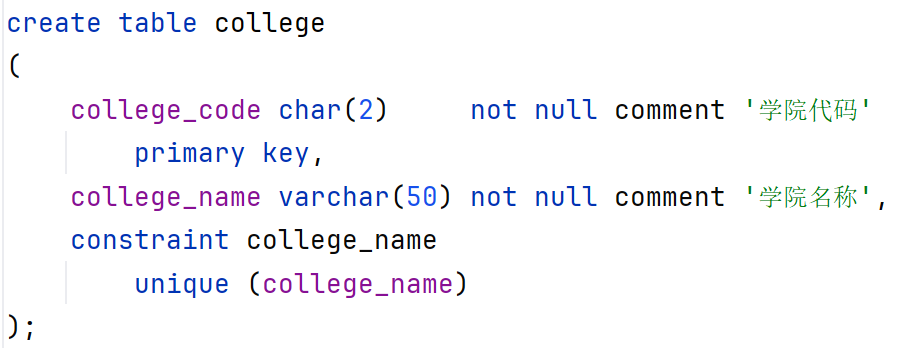
成绩表包含编号、学号、课程号、分数

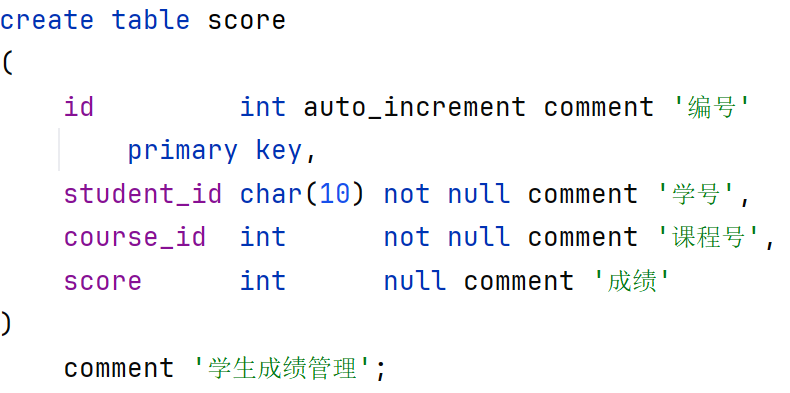
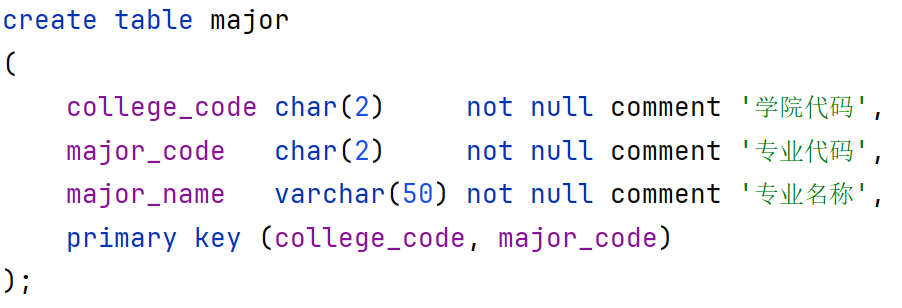
专业表包含学院代码、专业代码、专业名称

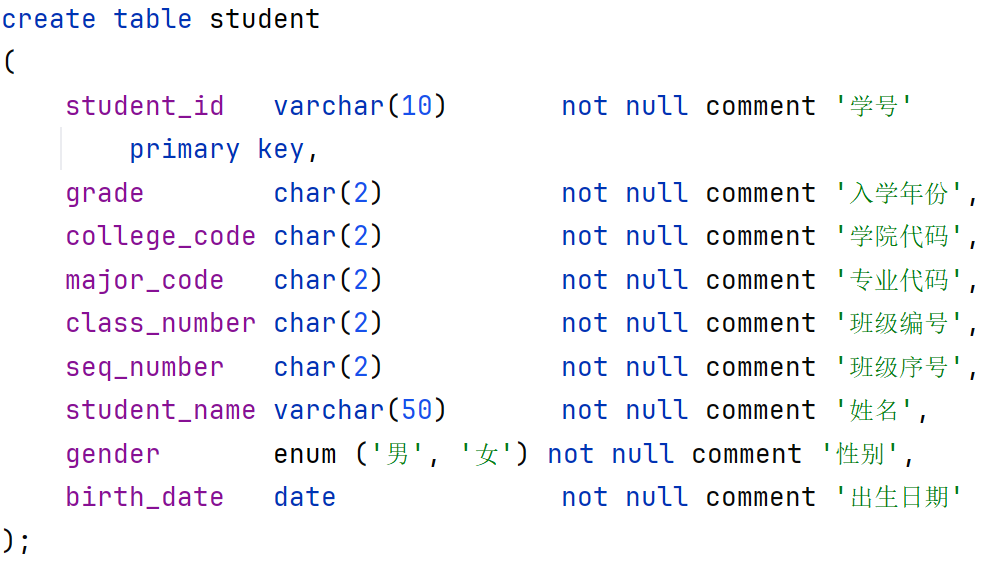
学院表包含学院代码、学院名称

课程表包含课程号和课程名称

对应SQL语句如下：







## 2. 后端核心设计

实体类Entity设计：

学生实体类：

public class Student extends BaseEntity  
{  
 private static final long *serialVersionUID* = 1L;  
 */\*\* 学号 \*/* private String studentId;@Excel(name = "入学年份")  
 private String grade;@Excel(name = "学院代码")  
 private String collegeCode;@Excel(name = "专业代码")  
 private String majorCode;@Excel(name = "班级编号")  
 private String classNumber;private String seqNumber;@Excel(name = "姓名")  
 private String studentName;@Excel(name = "性别")  
 private String gender;@JsonFormat(pattern = "yyyy-MM-dd")  
 @Excel(name = "出生日期", width = 30, dateFormat = "yyyy-MM-dd")  
 private Date birthDate;

}

成绩实体类：

public class Score extends BaseEntity  
{  
 private static final long *serialVersionUID* = 1L;private Long id;  
 private String studentId;  
 private String studentName;private Long courseId;  
 private String courseName;  
 private Long score;  
 private String classNumber;  
 private Double classAvgScore;  
 private Double courseAvgScore;  
 private Double totalScore;  
 private Double classAvgTotal;

}

控制器Controller设计：

学生信息控制器：

public class StudentController extends BaseController  
{  
 @Autowired  
 private IStudentService studentService;  
 @GetMapping("/list")  
 public TableDataInfo list(Student student)  
 {  
 startPage();  
 List<Student> list=studentService.selectStudentList(student);  
 return getDataTable(list);  
 }  
 @GetMapping(value = "/{studentId}")  
 public AjaxResult getInfo(@PathVariable("studentId") String studentId)  
 {  
 return success(studentService.selectStudentByStudentId(studentId));  
 }  
 @PostMapping  
 public AjaxResult add(@RequestBody Student student)  
 {  
 return toAjax(studentService.insertStudent(student));  
 }  
 @PutMapping  
 public AjaxResult edit(@RequestBody Student student)  
 {  
 return toAjax(studentService.updateStudent(student));  
 }  
 @DeleteMapping("/{studentIds}")  
 public AjaxResult remove(@PathVariable String[] studentIds)  
 {  
 return toAjax(studentService.deleteStudentByStudentIds(studentIds));  
 }  
}

其中TableDataInfo利用了分页插件进行分页，返回特定的类型给前端，使前端能方便交互并展示表格内容  
*/\*\* 总记录数 \*/*private long total;  
*/\*\* 列表数据 \*/*private List<?> rows;  
*/\*\* 消息状态码 \*/*private int code;  
*/\*\* 消息内容 \*/*private String msg;

AjaxResult是统一结果返回类,其中包含：

*/\*\* 状态码 \*/*public static final String *CODE\_TAG* = "code";  
*/\*\* 返回内容 \*/*public static final String *MSG\_TAG* = "msg";  
*/\*\* 数据对象 \*/*public static final String *DATA\_TAG* = "data";

学生成绩控制器：

public class ScoreController extends BaseController  
{  
 @Autowired  
 private IScoreService scoreService;  
 @Autowired  
 private AIService aiService;  
 @GetMapping("/list")  
 public TableDataInfo list(Score score)  
 {  
 startPage();  
 List<Score> list = scoreService.selectScoreList(score);  
 return getDataTable(list);  
 }  
 @GetMapping(value = "/{id}")  
 public AjaxResult getInfo(@PathVariable("id") Long id)  
 {  
 return success(scoreService.selectScoreById(id));  
 }  
 @PostMapping  
 public AjaxResult add(@RequestBody Score score)  
 {  
 return toAjax(scoreService.insertScore(score));  
 }  
 @PutMapping  
 public AjaxResult edit(@RequestBody Score score)  
 {  
 return toAjax(scoreService.updateScore(score));  
 }  
 @DeleteMapping("/{ids}")  
 public AjaxResult remove(@PathVariable Long[] ids)  
 {  
 return toAjax(scoreService.deleteScoreByIds(ids));  
 }  
 @PostMapping("/export")  
 public void exportStatistics(HttpServletResponse response)  
 {  
 List<Score> list = scoreService.selectScoreStatistics();  
 ExcelUtil<Score> util = new ExcelUtil<Score>(Score.class);  
 util.exportExcel(response, list, "学生成绩统计报表");  
 }  
 @GetMapping("/analyze")  
 public AjaxResult analyzeScores(Score score) {  
 List<Score> scores = scoreService.selectScoreList(score); *// 获取所有成绩数据* DashScopeResponse analysisResult = aiService.analyzeScoresWithAI(scores); *// 调用 AI 服务* return success(analysisResult); *// 返回完整响应对象* }  
}

其中exportStatistics方法调用了封装好的ExcelUtil工具类，可以直接生成对应的Excel报表数据，实现方便快捷。

<select id="selectScoreStatistics" resultMap="ScoreResult">  
 WITH StudentTotalScore AS (  
 SELECT  
 s.student\_id,  
 st.student\_name,  
 st.class\_number,  
 SUM(s.score) as total\_score,  
 COUNT(\*) as course\_count  
 FROM score s  
 LEFT JOIN student st ON st.student\_id = s.student\_id  
 GROUP BY s.student\_id, st.student\_name, st.class\_number  
 ),  
 ClassAvgTotal AS (  
 SELECT  
 class\_number,  
 AVG(total\_score) as class\_avg\_total  
 FROM StudentTotalScore  
 GROUP BY class\_number  
 )  
 SELECT  
 s.student\_id,  
 st.student\_name,  
 c.course\_name,  
 s.score,  
 st.class\_number,  
 sts.total\_score,  
 cat.class\_avg\_total,  
 (SELECT AVG(s2.score)  
 FROM score s2  
 LEFT JOIN student st2 ON st2.student\_id = s2.student\_id  
 WHERE s2.course\_id = s.course\_id AND st2.class\_number = st.class\_number) as class\_avg\_score,  
 (SELECT AVG(s3.score) FROM score s3 WHERE s3.course\_id = s.course\_id) as course\_avg\_score  
 FROM score s  
 LEFT JOIN student st ON st.student\_id = s.student\_id  
 LEFT JOIN course c ON c.course\_id = s.course\_id  
 LEFT JOIN StudentTotalScore sts ON sts.student\_id = s.student\_id  
 LEFT JOIN ClassAvgTotal cat ON cat.class\_number = st.class\_number  
 ORDER BY st.class\_number, sts.total\_score DESC, s.student\_id, s.course\_id  
</select>

上述代码通过SQL语句，将两个子查询结果与score表连接起来，获取完整的成绩数据，并计算出学生总分、班级总分平均值、课程平均分和班级课程平均分，展示于Excel表上

此外我们采用了通义千问Qwen的大模型“通义千问-Turbo”进行API调用，具体代码如下：

public DashScopeResponse analyzeScoresWithAI(List<Score> scores) {  
 StringBuilder dataBuilder = new StringBuilder();  
 dataBuilder.append("以下是学生成绩数据：其中学号前两位表示入学年份，之后每两位分别表示：学院，专业，班级，序号\n");  
 for (Score score : scores) {  
 dataBuilder.append(String.*format*(  
 "学生姓名：%s, 学号：%s, 课程名称：%s, 班级：%s, 成绩：%d\n",  
 score.getStudentName(), score.getStudentId(), score.getCourseName(),score.getClassNumber(),score.getScore()  
 ));  
 }  
 return callTongyiQianwen(dataBuilder.toString());  
}  
private DashScopeResponse callTongyiQianwen(String inputData) {  
 try {  
 Generation generation = new Generation();  
 Message systemMsg = Message.*builder*()  
 .role(Role.*SYSTEM*.getValue())  
 .content("你是教育数据分析专家，请根据以下成绩数据进行分析，并用一段话总结整体情况。可以分为年级、班级等进行分析")  
 .build();  
 Message userMsg = Message.*builder*()  
 .role(Role.*USER*.getValue())  
 .content(inputData)  
 .build();  
 GenerationParam param = GenerationParam.*builder*()  
 .apiKey(apiKey)  
 .model("qwen-turbo")  
 .messages(Arrays.*asList*(systemMsg, userMsg))  
 .resultFormat(GenerationParam.ResultFormat.*MESSAGE*)  
 .build();  
 GenerationResult result = generation.call(param);  
 *// 手动转换为 DashScopeResponse* String jsonResponse = JsonUtils.*toJson*(result);  
 return JsonUtils.*fromJson*(jsonResponse, DashScopeResponse.class);  
 } catch (Exception e) {  
 e.printStackTrace();  
 DashScopeResponse response = new DashScopeResponse();  
 response.setStatus\_code(500);  
 response.setMessage("AI 分析失败：" + e.getMessage());  
 return response;  
 }  
}

我们根据官网的官方API文档进行操作，导入相关依赖安装好SDK，再将对应的API-key和Base-url进行加载。

先采用analyzeScoresWithAI(List<Score> scores) 方法接收一个学生成绩列表，将每条成绩信息格式化为“姓名、学号、课程、班级、成绩”的结构化文本，并说明学号结构含义，统一拼接成一段待分析的数据文本。

再使用callTongyiQianwen(String inputData) 方法将生成的成绩文本作为“用户消息”，再配合一段“系统提示词”（提示大模型你是教育分析专家，请进行分析），一并封装为消息列表，发送给AI大模型。

接着使用 GenerationParam 构造模型调用请求，指定使用模型为 "qwen-turbo"，传入 API 密钥、消息列表、输出格式等参数，并通过 generation.call(param) 发起分析请求。

最后将通义千问的返回结果通过 JSON 工具解析为 DashScopeResponse 对象。如果调用过程中出现异常，则返回一个包含错误信息的失败响应。

DashScope类代码如下：

public class DashScopeResponse {  
 private int status\_code;  
 private String request\_id;  
 private String code;  
 private String message;  
 private Output output;  
 private Usage usage;

*// 内部类 Output*public static class Output {  
 private String text;  
 private String finish\_reason;  
 private List<Choice> choices;

}

*// 内部类 Choice*public static class Choice {  
 private String finish\_reason;  
 private Message message;

}

*// 内部类 Message*public static class Message {  
 private String role;  
 private String content;

}

*// 使用情况统计*public static class Usage {  
 private int input\_tokens;  
 private int output\_tokens;  
 private int total\_tokens;

}

}

采用了DashScope 的方式调用通义千问 API，其中在pom.xml导入相关SDK

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>dashscope-sdk-java</artifactId>  
 <version>2.12.0</version>

</dependency>

## 3. 前端核心设计

调用了echarts组件进行柱状图的动态显示

这里重点展示柱状图的实现逻辑：

import \* as echarts from 'echarts';

const initChart = async () => {

  if (!chartRef.value) return;

  const chart = echarts.init(chartRef.value);

  try {

    // 获取指定学生的成绩数据

    const res = await listScore({

      studentId: selectedStudent.value,

      pageSize: 100 // 获取所有数据

    });

    // 处理课程成绩数据

    const courseData = res.rows.reduce((acc, cur) => {

      if (cur.courseId && cur.score !== null) {

        const courseName = course.value.find(c => c.value == cur.courseId)?.label || '未知课程';

        acc.push({ name: courseName, value: cur.score });

      }

      return acc;

    }, []);

    // 配置图表选项

    const option = {

      title: {

        text: '各科成绩分布',

        left: 'center'

      },

      tooltip: {},

      xAxis: {

        type: 'category',

        data: courseData.map(item => item.name)

      },

      yAxis: {

        type: 'value',

        max: 100,

        min: 0

      },

      series: [{

        type: 'bar',

        data: courseData.map(item => item.value),

        itemStyle: {

          color: '#409EFF'

        }

      }]

    };

    chart.setOption(option);

    // 窗口变化时自适应

    window.addEventListener('resize', () => chart.resize());

  } catch (error) {

    console.error('图表数据加载失败:', error);

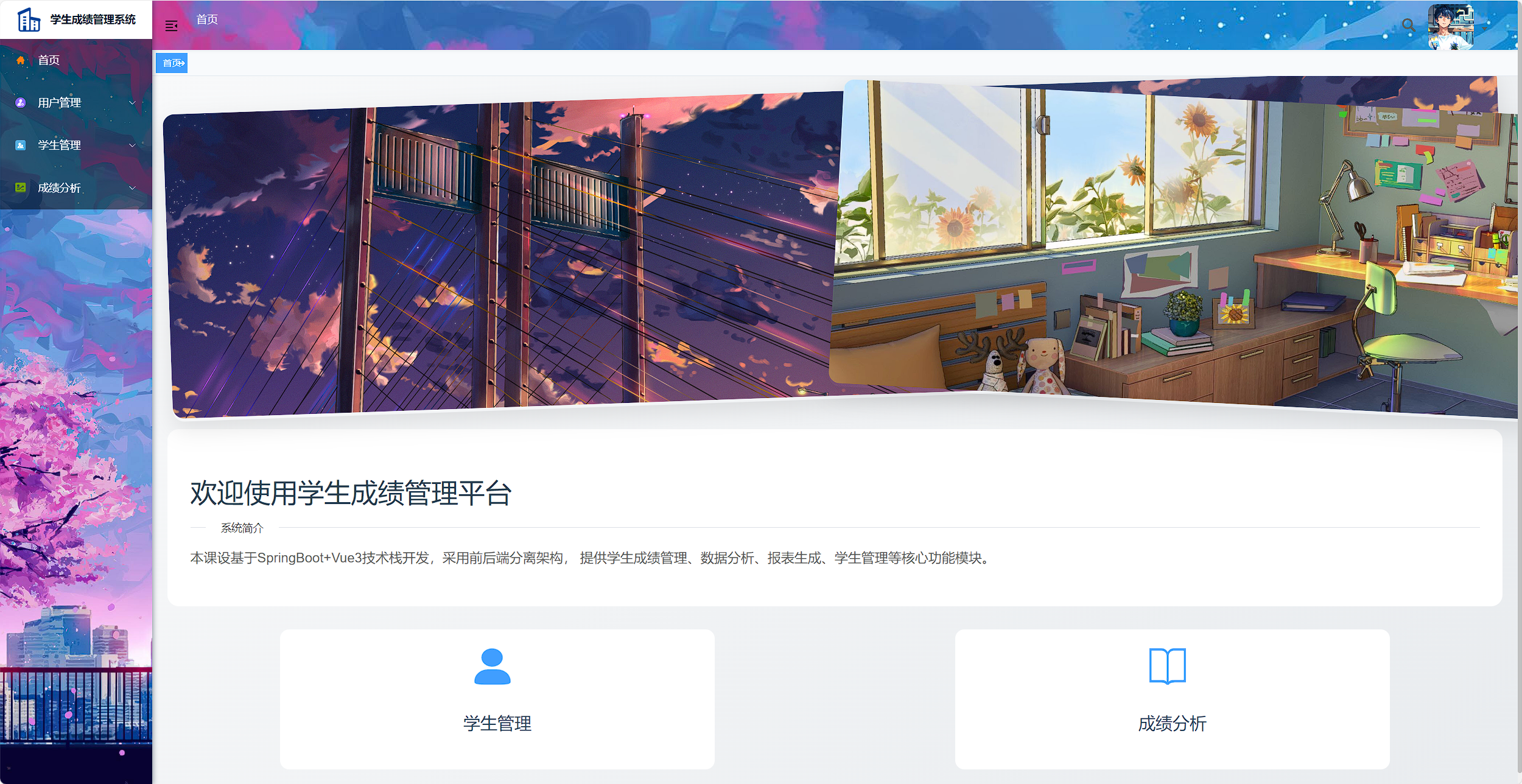
  }

};

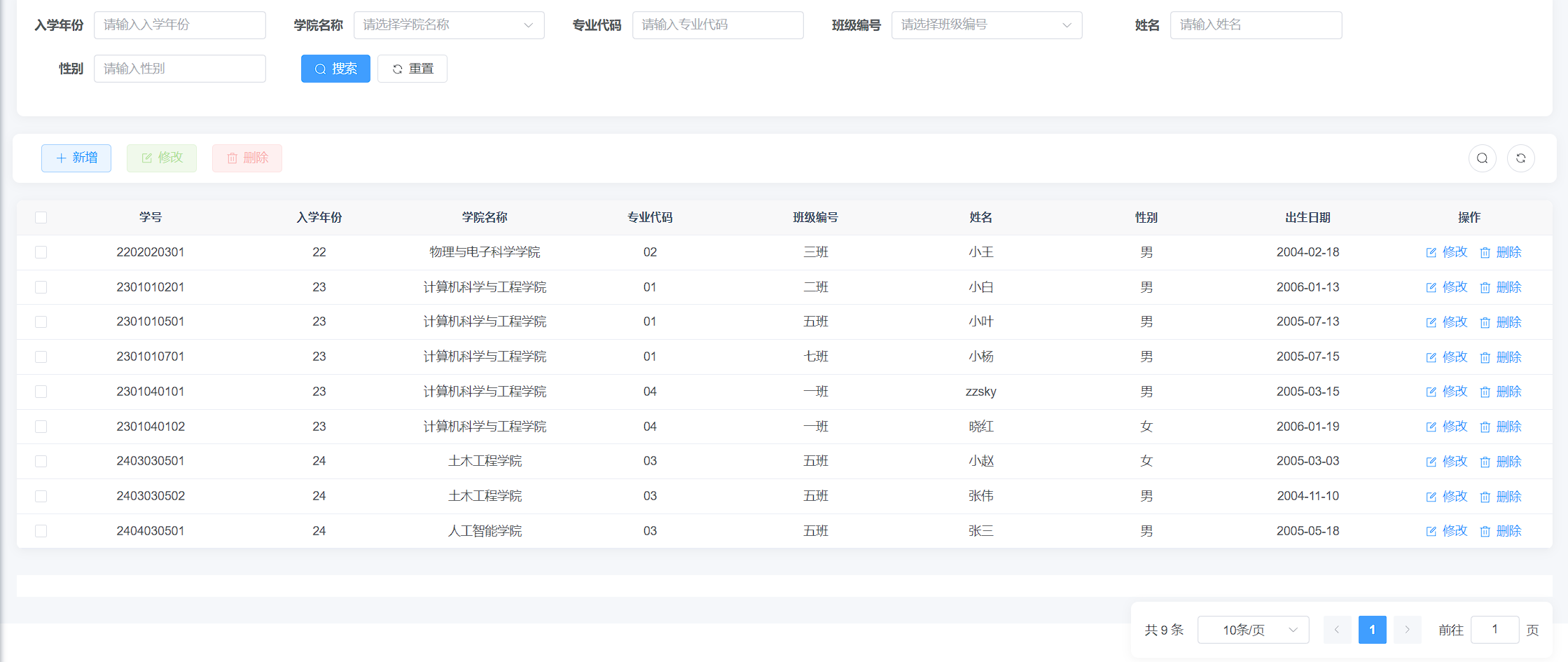
登录页面展示：



主页展示：



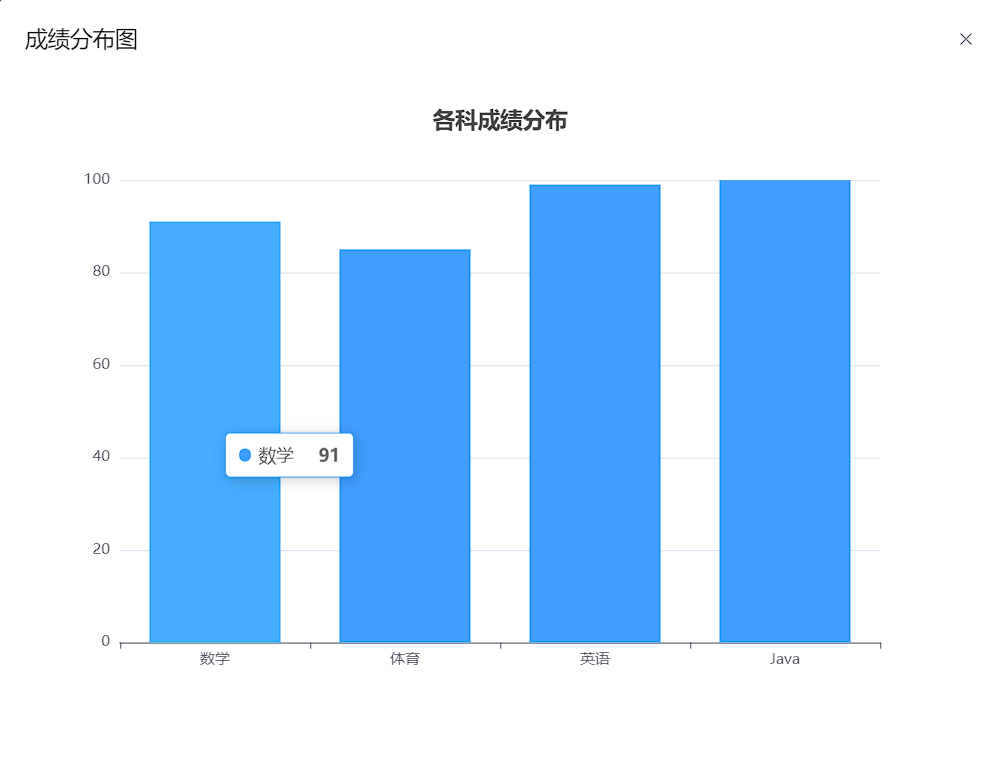
学生信息管理页面展示：



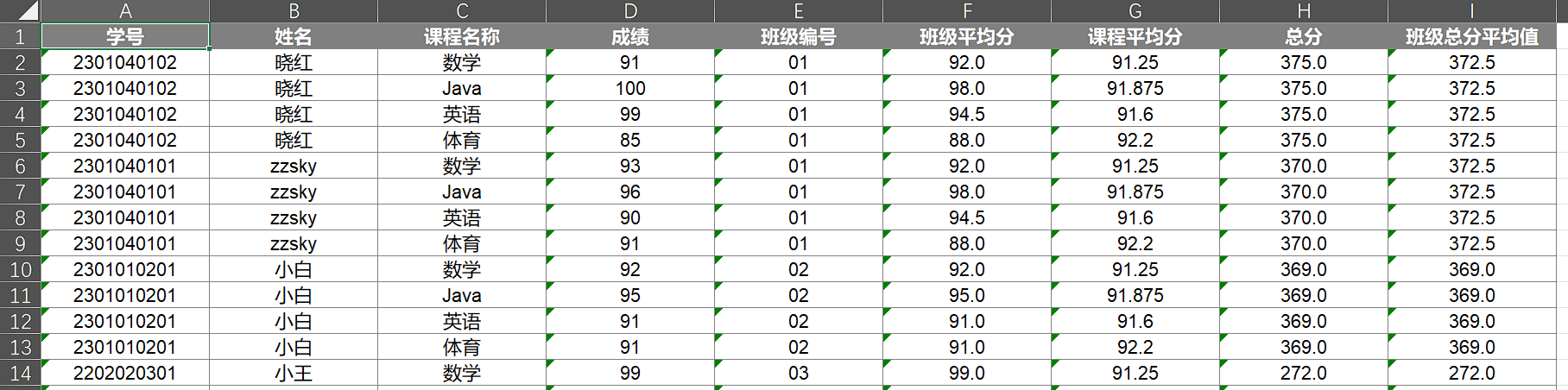
学生成绩管理页面展示：



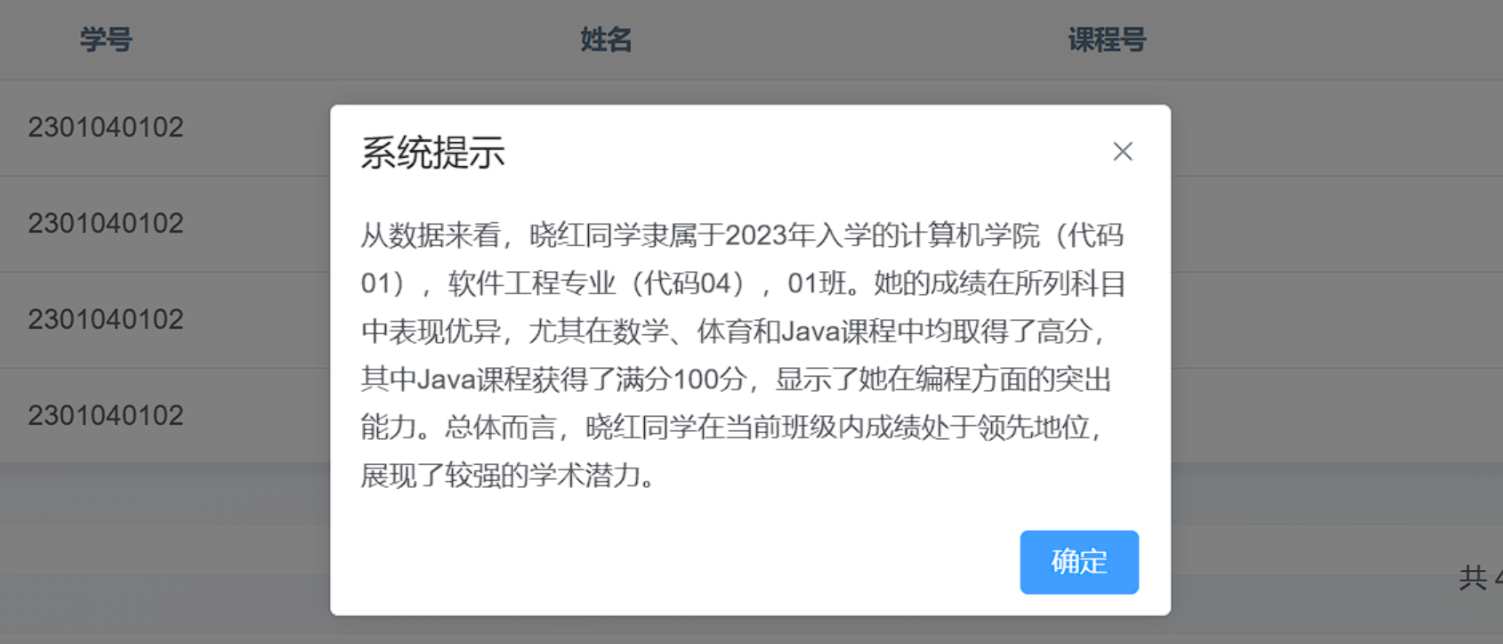
成绩数据可视化展示：

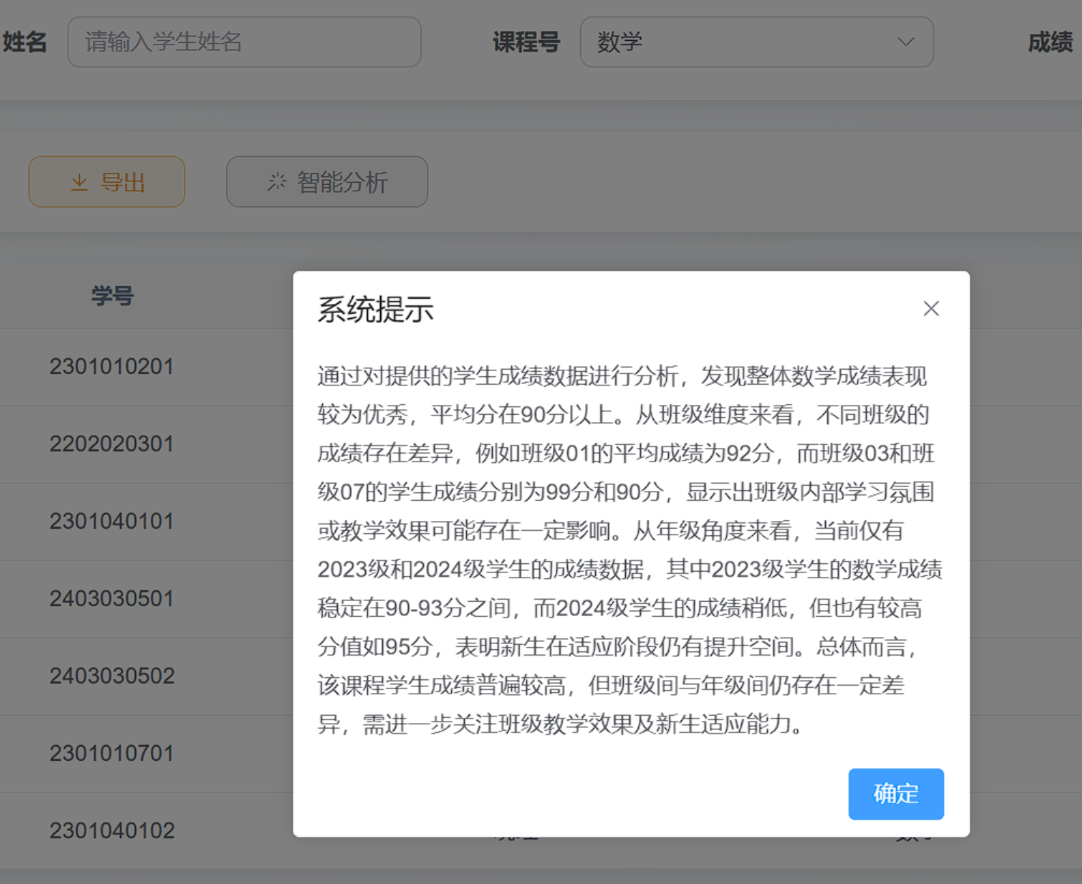


学习情况报表展示：



AI大模型分析展示：



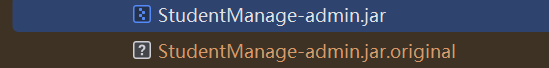


## 4. 本地部署实现

本实验实现部分基本部署功能，以更加深入的了解web项目的上线运行流程

采用npm run build进行前端打包，maven的package完成后端的jar包打包





前端通过配置nginx.conf配置文件，启动Nginx运行项目

server {

        listen       70;//端口

        server\_name  /\*your\_ip\*/;

        location / {

            root   html;

            index  index.html index.htm;

            try\_files $uri $uri/ /index.html;

        }

            location /prod-api/ {

                proxy\_pass <http://localhost:8080/>;

            }

后端通过命令java -jar “jar包名”运行项目

五、实验结果与分析

本系统已实现题目要求的各项核心功能，实验效果如下：

1. 功能完整性

* 学生信息添加功能运行正常，学号通过系统自动拼接生成，结构合理，确保唯一性与可追溯性；
* 查询功能支持按学号（精确匹配）和姓名（模糊匹配）两种方式，支持多名重名学生同时显示，查询结果准确，提示机制完善；
* 成绩可视化功能使用 ECharts 实现柱状图展示，图表美观、数值标注清晰，支持响应式缩放；
* 支持对学生信息的修改与删除，修改时学号字段被锁定，避免主键错误更新；
* 报表导出功能集成 EasyExcel 库，能够生成包含每位学生单科成绩、总成绩、班级均值等数据的综合报表，数据结构完整，导出格式标准；
* 系统集成调用了 通义千问-Turbo 大模型，将选中学生或报表数据作为输入参数，生成智能化的成绩分析与学习建议文本，提升系统的智能交互能力与实际参考价值。

1. 系统稳定性与用户体验

* 前端采用 Vue 3 + Element Plus 实现响应式页面，表单录入、分页展示、数据搜索等交互操作流畅，用户体验良好；
* 后端接口基于 Spring Boot + MyBatis 构建，结构清晰，接口响应速度快，前后端数据交互准确；
* 报表导出、图表绘制和数据分析等功能在不同数据量下运行稳定，未出现卡顿或错误。

1. 部署与运行情况

* 系统采用前后端分离架构，前端项目通过 npm build 打包部署至 Nginx，后端使用 IDEA 自带 Tomcat 成功运行；
* 本地部署顺利，本地运行环境下，系统在浏览器中可正常访问使用，同时通过局域网 IP 地址和对应端口，也可在其他设备访问，实现多终端访问与测试。

综上，学生成绩管理系统功能齐全、逻辑严谨，界面美观、操作简洁，具备良好的用户体验和实际教学管理价值。系统在数据处理能力、稳定性、可扩展性等方面表现优异，同时结合 AI 大模型进行成绩分析，体现了智能化和创新性，完全满足课程设计的需求目标，具有良好的用户体验和实用价值。

六、小结与心得体会

通过本次学生成绩管理系统的开发，我熟悉了前后端分离项目的整体开发流程，系统采用 Vue + Spring Boot 技术栈实现，具备良好的功能结构与可维护性。系统实现了学生信息管理、成绩录入与查询、成绩可视化（柱状图）、学习报表导出、成绩智能分析等关键功能，功能完整、界面清晰、交互友好。

我们完成了前后端打包部署，前端通过 Nginx 进行静态资源托管，后端运行于 IDEA 内置的 Tomcat 容器，整个系统在本地环境中可独立运行，部署过程顺利，运行稳定。

项目开发过程中也遇到了不少挑战，例如前后端分离带来的跨域访问问题、数据库设计初期未规范导致的数据冗余、多表关联 SQL 编写错误，以及对 ECharts 图表配置不熟悉等问题。通过不断查阅资料、调试日志、分析问题并动手解决，我逐步提升了数据库建模能力、接口联调能力、前端组件开发能力以及图表配置能力，增强了综合实践水平。

值得一提的是，在后期的成绩分析模块中，我们尝试集成调用了通义千问大模型，将部分学生成绩的数据发送至大模型接口，由其生成智能分析结论和个性化学习建议。这一创新增强了系统的智能化水平，也让我掌握了调用大模型 API 进行数据智能分析的实际流程，包括数据封装、API对接、结果解析与展示等关键操作。

本次课程设计让我对 Web 项目的需求分析、模块设计、接口开发、数据处理和智能化扩展有了系统性的理解，为我未来参与更复杂、更智能的系统开发项目打下了坚实基础。