# 

**《复杂软件开发的实践方法》**

**课程实践报告**

项目名称：学生成绩管理系统

项目组成员：

|  |  |  |
| --- | --- | --- |
| 姓名 | 学号 | 评分 |
| 谢卿云 | 2022010910017 | A |
| 张魏晋 | 2022080301012 | C |
| 秦宇韬 | 2022080301017 | B |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

注：组内评分采用分档制，由高到低依次为：A、B、C、D、E等，要求不能并列，每个档次都要有。

## 一、实践目标

本课程以开发学生成绩管理系统为目标，在教师的指引下，完成对应的需求分析、系统架构设计、系统详细设计、编码、测试和部署等工作。

**二、实践原理**

本实践课程主要以数据库、Web前后端开发为主，主要需要掌握的技术包括MySQL、Node Express框架和Vue框架等方面内容。

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB公司开发，属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。 本项目采用MySQL来存储学生成绩数据。

Node Express 是一个简洁而灵活的 Node.js Web应用框架, 提供了一系列强大特性和丰富的 HTTP 工具来帮助创建各种 Web 应用，使用 Express 可以快速地搭建一个完整功能的网站。Express 框架核心特性包括：可以设置中间件来响应 HTTP 请求；定义了路由表用于执行不同的 HTTP 请求动作；可以通过向模板传递参数来动态渲染 HTML 页面。本项目采用Node Express作为学生成绩管理系统的后端开发框架。

Vue是一套用于构建用户界面的渐进式框架。Vue 的核心库只关注视图层，采用自底向上增量开发的设计，通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。其特点包括易上手，易与第三方库或既有项目整合，能够为复杂的单页应用提供驱动。本项目采用Vue作为前端框架。

**三、实践内容与要求**

1、安装并搭建开发环境；

a)安装MySQL和数据库管理和设计工具Navicat；

b)安装Visual Studio Code, node.js和npm；

c)安装Vue和Node Express。

2、基于Vue.js完成学生成绩管理系统前端设计；

3、基于Node Express和MySQL完成学生成绩管理系统后端设计；

4、完成学生成绩管理系统的前后端集成、功能完善和扩展。

具体的实践要求如下：分组协作完成一个前后端分离的学生成绩管理Web系统，最后提交内容：学生成绩管理系统成品、设计报告。

**四、实践器材（设备、元器件）：**

1. 电脑硬件要求：

酷睿i5以上CPU、1G内存、1T 硬盘；

1. 系统平台要求：

Windows 7以上；

3) 软件及相应开发工具要求：

Visual Studio Code、Node.js、Vue.js、Node Express、MySQL 5.0以上。

**五、实践过程与结果**

本实践内容包含四个阶段：需求分析阶段、系统设计阶段、编码实现阶段、测试部署阶段。

**1、需求分析**

本课程实践的目标是开发学生成绩管理系统网络应用。教师在该系统中可以输入各科成绩，如语文、数学、英语、物理、化学等科目，同时可查询、修改、删除学生成绩。该系统可统计学生各科成绩总分、平均分并可对学生成绩进行排序等功能。

作为一个软件开发项目，我们首先需要明确系统的功能需求。具体的功能需求如下：

1. 展示数据库中已有的学生信息及成绩，包括学生id、学生的姓名、语数英三科的成绩、三科总分及均分；
2. 对学生信息按照学生三科总分高低排序，采取降序形式；
3. 增加学生信息并成功显示在展示页面；
4. 修改已有学生成绩记录；
5. 删除已有学生成绩记录；
6. 查询某个特定id的学生的成绩信息；

本系统的非功能性需求如下：开发环境后端为MySQL + Node Express，前端为Vue。

**2、系统设计**

本项目采用前后端分离设计，前端应用专门负责数据展示和用户交互，后端应用专门负责提供数据处理接口，前端 HTML 页面通过 Axios调用后端 RESTful API 接口进行数据交互。前后端程序员只需要提前约定好接口文档（参数、数据类型），然后并行开发即可，最后完成前后端集成，遇到问题同步修改即可，真正实现了前后端应用的解耦合，可以极大地提升开发效率。

本项目的整体架构如图1。

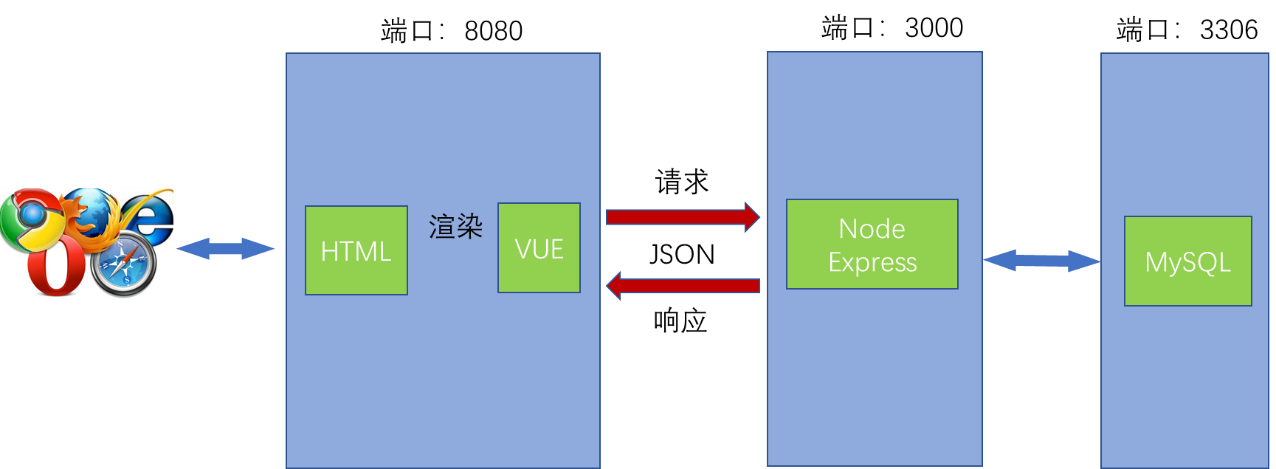
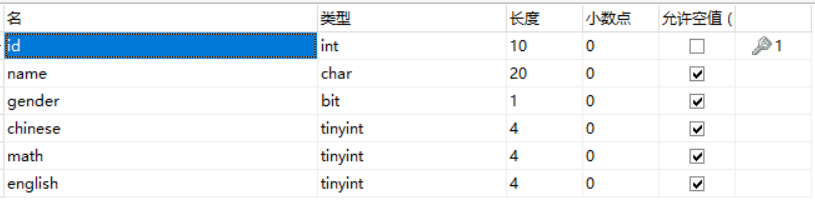


图1 系统整体技术架构

本系统的设计包括四个部分：数据库设计、前后端接口设计、前端设计、后端设计，下面分别进行介绍。

**2.1 数据库设计**

本系统需新建一个数据库vuestu，新建一张数据表stuScore，该表字段信息如下：



各字段的含义如下

id： 录入的学生的ID（学号）；

name： 录入的学生的姓名；

gender： 录入的学生的性别；

chinese： 录入的学生的语文成绩（0~100）；

math： 录入的学生的数学成绩（0~100）；

english： 录入的学生的英语成绩（0~100）；

**2.2 前后端接口设计**

本系统前后端API交互采用HTTP POST请求，输入和输出参数都采用JSON格式。相应的接口定义如下。

1. 按学号顺序显示学生成绩

方法URL：localhost:8081/

输入参数：无

输出参数：学生成绩列表，每个学生信息的JSON格式如下

"id": 10001, /\*学号\*/

"name": "test", /\*学生姓名\*/

"gender": "male", /\*学生性别\*/

"chinese": 100, /\*语文成绩\*/

"math": 100, /\*数学成绩\*/

"english": 100 /\*英语成绩\*/

"aggregate": 300 /\*三科总成绩\*/

"average": 100 /\*三科均分\*/

2）按平均成绩由高到低显示学生成绩

方法URL：localhost:8081/api

输出参数：按平均成绩由高到低排序的学生成绩列表，每个学生信息的JSON格式如下

"id": 10001, /\*学号\*/

"name": "test", /\*学生姓名\*/

"gender": "male", /\*学生性别\*/

"chinese": 100, /\*语文成绩\*/

"math": 100, /\*数学成绩\*/

"english": 100 /\*英语成绩\*/

"aggregate": 300 /\*三科总成绩\*/

"average": 100 /\*三科均分\*/

3）增加一个学生的成绩

方法URL：localhost:8081/insert

输入参数：学生信息的JSON格式如下

"id": 10001, /\*学号\*/

"name": "test", /\*学生姓名\*/

"gender": "male", /\*学生性别\*/

"chinese": 100, /\*语文成绩\*/

"math": 100, /\*数学成绩\*/

"english": 100 /\*英语成绩\*/

"aggregate": 300 /\*三科总成绩\*/

"average": 100 /\*三科均分\*/

输出参数：正确增加时，返回“成功添加该学生成绩”

未能正确增加时，返回“添加该学生成绩失败”

4）修改一个学生的成绩

方法URL：localhost:8081/edit

输入参数：学生信息的JSON格式如下

"id": 10001, /\*学号\*/

"name": "test", /\*学生姓名\*/

"gender": "male", /\*学生性别\*/

"chinese": 100, /\*语文成绩\*/

"math": 100, /\*数学成绩\*/

"english": 100 /\*英语成绩\*/

输出参数：正确修改时，返回“成功修改该学生成绩”

未能正确修改时，返回“修改该学生成绩失败”

5）删除一个学生的成绩

方法URL：localhost:8081/del

输入参数：学生信息只包含学号，其对应的JSON格式如下

"id": 10001, /\*学号\*/

输出参数：正确删除时，返回“成功删除该学生成绩”

未能正确删除时，返回“删除该学生成绩失败”

6）获取指定id的学生的信息

方法URL：localhost:8081/api/:id

输入参数：学生信息只包含学号，其对应的JSON格式如下

"id": 10001, /\*学号\*/

输出参数：正确获取时，返回“成功获取该学生成绩”

未能正确获取时，返回“获取该学生成绩失败”

**2.3 后端设计**

后端采用express来实现JSON数据格式交互，包括以下步骤：

1. 安装express

使用的命令如下：

|  |
| --- |
| npm install express -g  npm install express-generator |

2） 创建目录结构和整体框架

使用的命令如下：

|  |
| --- |
| npm i back |

3）配置数据库

修改back/express\_index.js文件，并实现异步连接池数据库连接。相关代码如下：

|  |
| --- |
| var mysql = require('mysql')  //创建数据库连接  var connection = mysql.createConnection({   host: "localhost",   user: "root",   password: "123456",   database: "vuestu",  });  connection.connect(); |

4）配置body-parser中间件

修改back/express\_index.js文件，以实现后端对传入的请求体进行解析。具体代码如下：

|  |
| --- |
| //配置body-parser中间件  const BodyParser = require('body-parser')  app.use(BodyParser.urlencoded({ extended: false }))  app.use(BodyParser.json()) |

5）创建api方法

修改back/express\_index.js文件，创建多个后端API。具体代码如下：

|  |
| --- |
| // 获取stuScore表中的所有数据  app.get("/api", function(req, res) {   console.log("主页 GET 请求");   connection.query("SELECT \* FROM stuScore1", function(error, results, fields) {   if (error) throw error;   console.log("The result is: ", results);   res.writeHead(200, { 'Content-Type': 'application/json' });   res.end(JSON.stringify(results));   });  });  // /del接收前端要删除的请求体，获取id，在数据库中进行删除操作  app.post("/del", function(req, res) {   console.log("/del 响应 DELETE 请求");   var { id, } = req.body;   console.log('Id Retrieved: ' + id)   var sql = "DELETE FROM stuScore1 WHERE id = ?";   connection.query(sql, id, function(error, results) {   if (error) {   console.error("Delete Failed: " + error.message);   return;   }   console.log("The Result is : ", results);   });  });  // 配置了一个返回指定id的学生的信息的服务器  app.get('/api/:id', (req, res) => {   const studentId = parseInt(req.params.id); // 从URL参数中获取ID并转换为整数   const query = 'SELECT \* FROM stuScore1 WHERE id = ?';   connection.query(query, [studentId], (error, results) => {   if (error) {   res.status(500).send('Server error: ' + error.message);   } else if (results.length > 0) {   console.log(results);   res.json(results[0]);   } else {   res.status(404).send('Student not found');   }   });  });  // /edit 页面 POST 请求,获取要修改学生的表单  // 在数据库用id查到后，用表单更新数据库原来的数据  app.post("/edit", function(req, res) {   console.log("/edit Post Request");   // 从请求体中获取学生ID和成绩   console.log(req.body)   var { id, name, gender, chinese, math, english } = req.body;   var dt = converter(id, name, gender, chinese, math, english)   console.log(dt)   var sql = "UPDATE stuScore1 SET name = ?, gender = ?, chinese = ?, math = ?, english = ? WHERE id = ?";   connection.query(sql, [dt.name, dt.gender, dt.chinese, dt.math, dt.english, dt.id], function(error, results) {   if (error) {   console.error("Updata Failed: " + error.message);   return;   }   console.log("Update Results: ", results);   });  });  // /insert 页面post请求，插入一个新学生的信息  // 直接传表单，简单处理后，插入进数据库  app.post("/insert", function(req, res) {   console.log("/insert Post 请求");   var { id, name, gender, chinese, math, english } = req.body;   console.log(id, name, gender, chinese, math, english)   var dt = converter(id, name, gender, chinese, math, english)   console.log(dt)   var sql = "INSERT INTO stuScore1 (id, name, gender, chinese, math, english) VALUES (?, ?, ?, ?, ?, ?)";   connection.query(sql, [dt.id, dt.name, dt.gender, dt.chinese, dt.math, dt.english], function(error, results) {   if (error) {   console.error("Insert Failed: " + error.message);   return;   }   console.log("Insert Results: ", results);   });  }); |

6）解决后端跨域问题

由于本系统采用的前后端分离的方式，前后端不在同一个域中，需要在后端解决跨域问题。这里通过修改后端代码来实现，具体方法是采用cors第三方模块实现。

具体步骤包括：

1. 安装第三方cors模块

npm i -g cors

1. 修改后端express\_index.js源代码文件
   1. 在文件开始处引入相关依赖

var cors=require(‘cors’)

* 1. 在app变量创建之后加入如下语句

app.use(cors())

最后的代码如下所示：

|  |
| --- |
| var express = require("express") var mysql = require("mysql") var app = express();  var cors = require('cors')  app.use(cors())  app.use("/public", express.static("public")) |

7）解决调整数据类型

由于前端传回的gender属性的类型为string，而数据库中类型为bit，需要将前端打包传回的数据在后端进行处理后再存入数据库。具体代码如下：

8）启动服务器

|  |
| --- |
| function converter(id, name, gender, chinese, math, english) {   var g   if (gender === 'male') {   g = 1;   } else if (gender === 'female') {   g = 0;   } else {   g = null;   }   return {   id: Number(id),   name: String(name),   gender: g,   chinese: Number(chinese),   math: Number(math),   english: Number(english)   };  } |

当访问网站根目录时，服务器执行查询操作同时对已有数据进行排序展示。当根目录接收到POST请求时，服务器响应。具体代码如下：

|  |
| --- |
| var server = app.listen(8081, '127.0.0.1', function() {   var host = server.address().address;   var port = server.address().port;   console.log("应用实例，访问地址为 http://%s:%s/api", host, port);     // 指定id学生的信息   console.log("应用实例，访问地址为 http://%s:%s/api/1", host, port);     //当 /add\_user接收到POST请求时， 服务器响应“ 增加成绩”。   console.log("插入成绩页面，访问地址为 http://%s:%s/insert", host, port);     //当 /update\_user接收到POST请求时， 服务器响应“ 修改成绩”。   console.log("修改成绩页面，访问地址为 http://%s:%s/edit", host, port);     //访问 /del\_user时，服务器响应“删除成绩”。   console.log("删除学生成绩页面，访问地址为 http://%s:%s/del", host, port);     //访问/list\_user时，服务器响应“成绩列表”。   console.log("成绩列表页面，访问地址为 http://%s:%s/list\_user", host, port);  }); |

**2.4 前端设计**

前端使用Vue.js构建前端页面并实现与后端的交互。

1. 安装Vue.js

使用的命令如下：

|  |
| --- |
| npm install vue -g  npm install webpack -g  npm install vue-cli -g  npm install vue-router -g |

1. 初始化前端项目和整体框架

使用的命令如下：

|  |
| --- |
| vue init webpack front |

1. 配置路由router

前端的多种功能在多个不同的页面进行实现，需要对根目录进行路由配置，以实现各个页面之间的变化。具体代码如下：

|  |
| --- |
| import Vue from 'vue'  import Router from 'vue-router'  import StudentInfo from '@/components/StudentInfo'  import StudentInsert from '@/components/StudentInsert'  import StudentEdit from '@/components/StudentEdit'  // Vue use Router  Vue.use(Router)  // export Router  export default new Router({   routes: [{   path: '/',   name: 'StudentInfo',   component: StudentInfo   }, {   path: '/info',   name: 'StudentInfo',   component: StudentInfo   }, {   path: '/insert',   name: 'StudentInsert',   component: StudentInsert   }, {   path: '/edit/:id',   name: 'StudentEdit',   component: StudentEdit   }]  }) |

1. 功能实现

在该项目中，前端需要实现学生成绩管理系统功能中的展示已录入的学生的成绩信息，新增、修改或产出学生成绩记录，同时也需要不同的页面来实现这些功能。在具体操作中通过修改Vue.js生成的框架中的布局组件来实现之一设计需要。

1. 界面美化

在实际中，页面除功能能够正常运行外还需要进行美化。在该项目中，通过前端引入css文件来对界面进行美化。具体代码如下：

|  |
| --- |
| .page-background {   position: relative;   min-height: 100vh;   width: 100%;   z-index: 1;   /\* 设置背景图片，固定及覆盖 \*/   background-image: url('../img/back.png');   background-size: cover;   background-attachment: fixed;   background-position: center;  }    .page-background::before {   content: "";   position: absolute;   top: 0;   left: 0;   right: 0;   bottom: 0;   background: rgba(243, 242, 242, 0.5);   z-index: -1;  }  .highlight {   background-color: #97eda7;   font-weight: bold;  } |

**3、编码实现**

VuePress项目在创建的时候，会有默认主题（Default）。主题的概念可以理解为一个成型的项目，有现成的UI界面设计和完整的功能。我们要做的就是选择一个主题然后二次开发，这个主题可以选择默认的或者npm下载使用别人配置好的。 在此基础上，加入自己的博客文章，界面风格，功能点的增删等等。

具体而言，我们需要基于默认主题主题进行二次开发。打开src /components文件夹，可以看到很多的Vue布局组件。这里根据具体需求选择性修改即可。详细的修改过程如下：

1. StudentInfo.vue

该vue文件中的页面为根页面，是已录入的学生成绩按总分降序排序的展示页面，并且还需要实现各操作的跳转。具体代码如下：

|  |
| --- |
| <template>   <div class="page-background">   <h1>{{ msg }}</h1>   <!-- <h3>Student Scores Table</h3> -->   <div>   <button>   <router-link to="/insert" active-class="active">   <img src="../img/添加.svg" height = '12' weight = '12' />   add   </router-link>   </button>   <button @click="searchStudent">Serach</button>   <input type="text" v-model="searchQuery" placeholder="输入学生姓名或学号">   </div>   <table>   <tr>   <th>No</th>   <th>Id</th>   <th>Name</th>   <th>Gender</th>   <th>Chinese</th>   <th>Math</th>   <th>English</th>   <th>Aggregate</th>   <th>Average</th>   <th>Admin</th>   </tr>   <tr v-for="(s, index) in currentPageData" :key="index">   <td :class="{ 'highlight': s.id === highlightedId }">{{ index+1 }}</td>   <td :class="{ 'highlight': s.id === highlightedId }">{{ s.id }}</td>   <td :class="{ 'highlight': s.id === highlightedId }">{{ s.name }}</td>   <td>{{ s.gender }}</td>   <td>{{ s.chinese }}</td>   <td>{{ s.math }}</td>   <td>{{ s.english }}</td>   <td>{{ s.aggregate }}</td>   <td>{{ s.average }}</td>   <td>   <button>   <router-link :to="'/edit/' + s.id" active-class="active">   <img src="../img/编辑.svg" height='12' weight='12'/>   edit   </router-link>   </button>   <button>   <a href @click="del\_stu\_info(s.id,s.name)" active-class="active">   <img src="../img/删除.svg" height='12' weight='12'/>   del   </a>   </button>   </td>   </tr>   </table>   <!-- 前一页和后一页按钮 -->   <button @click="changePage(-1)" :disabled="currentPage === 1">Previous</button>   <!-- 页码按钮 -->   <span v-for="page in pages" :key="page">   <button @click="goToPage(page)" :disabled="currentPage === page">{{ page }}</button>   </span>   <button @click="changePage(1)" :disabled="currentPage === totalPages">Next</button>   </div>  </template>  <script>  export default {   name: "StudentInfo",   data() {   return {   msg: "Student Management App",   stu: Array(),   currentPage: 1,   pageSize: 10,   searchQuery: '',   highlightedId: null,   };   },   computed: {   totalPages() {   return Math.ceil(this.stu.length / this.pageSize);   },   pages() {   let pages = [];   for (let i = 1; i <= this.totalPages; i++) {   pages.push(i);   }   return pages;   },   currentPageData() {   const start = (this.currentPage - 1) \* this.pageSize;   const end = this.currentPage \* this.pageSize;   return this.stu.slice(start, end);   }   },   methods: {   searchStudent() {   console.log(this.searchQuery)   console.log(this.stu)   const matchedStudent = this.stu.find((student) =>   String(student.name) === String(this.searchQuery)   || String(student.id) === String(this.searchQuery)   );   if (matchedStudent) {   // 计算应跳转的页面   const pageIndex = Math.ceil((this.stu.indexOf(matchedStudent) + 1) / this.pageSize);   this.currentPage = pageIndex; // 设置当前页码   this.highlightedId = matchedStudent.id; // 设置高亮显示的学生ID   console.log(this.highlightedId)   } else {   alert("未找到对应的学生，请检查输入！");   this.highlightedId = null;   }   },   changePage(step) {   const requestedPage = this.currentPage + step;   if(requestedPage >= 1 && requestedPage <= this.totalPages) {   this.currentPage = requestedPage;   }   },   goToPage(pageNumber) {   this.currentPage = pageNumber;   },   // info从后端数据库获得   infoFetchData: async function(){   console.log('Data fethed from back')   try {   console.log('Data retrieved:')   const response = await this.$axios.get('http://127.0.0.1:8081/api')   console.log(response.data)   return response.data   } catch(error) {   console.error('Error fetching data:', error);   return []; // 发生错误时返回空数组   }   },   // 前端展示从后端取得的数据，排序处理   showData: function (data) {   let len = data.length;   let stuinfo;   for (let i = 0; i < len; i++) {   stuinfo = data[i]   stuinfo.gender = (data[i].gender.data[0]===1)?"男":"女"   stuinfo.aggregate = stuinfo.chinese + stuinfo.math + stuinfo.english   stuinfo.average = stuinfo.aggregate / 3   this.stu.push(stuinfo)   }   this.stu.sort(function(a,b){   return b.aggregate - a.aggregate //分数从大到小，最上面的是king   //return a.aggregate-b.aggregate;//分数从小到大   })   return this.stu;   },   // 删除的操作要返回到数据库中   // 后端，让后端执行删除的操作   del\_stu\_info: function (stuid,stuname) {   window.confirm(`你正在删除${stuname}(id:${stuid})的信息，你个大坏蛋！`)   this.$axios({   method:"post",   url:"http://127.0.0.1:8081/del",   data: {"id":stuid,},   }).then((res)=>{   console.log("Success Post");   console.log(res.data);   })   .catch(function (error) {   console.log(error);   })   },   },   created: async function(){   const data = await this.infoFetchData(); // 等待数据被获取,注意异步函数的调整   console.log(data);   this.showData(data); // 使用获取到的数据调用showData方法   },  };  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>  h3,  h4 {   font-weight: normal;  }  ul {   list-style-type: none;   padding: 0;  }  li {   display: inline-block;   margin: 0 10px;  }  a {   color: #42b983;  }  table {   width: 98%;   margin: 16px 0;   border-collapse: collapse;   border: 0;  }  th {   background-color: #93daff;   color: #000000;  }  th,  td {   font-size: 0.95em;   text-align: center;   padding: 4px;   border-collapse: collapse;  }  th,  td {   border: 1px solid #c1e9fe;   border-width: 1px 0 1px 0;  }  tr {   border: 1px solid #c1e9fe;  }  tr:nth-child(odd) {   background-color: #dbf2fe;  }  tr:nth-child(even) {   background-color: #fdfdfd;  }  </style> |

1. StudentEdit.vue

该页面是对已录入的学生的成绩等信息进行修改或更新。在修改过程中，除学生id不可以更改外，其余各属性值均可以更改。具体代码如下：

|  |
| --- |
| <template>   <div class="divCen">   <h3>{{ msg }}</h3>   <h4>Student Scores Edit</h4>   <p class="p-right">   <router-link to="/" active-class="active">Back to home</router-link>   </p>   <table>   <tr>   <th>Id</th>   <td>   <input type="text" :value="id" readonly />   </td>   </tr>   <tr>   <th>Name</th>   <input type="text" v-model="name" />   </tr>   <tr>   <th>Gender</th>   <input type="radio" v-model="gender" value = 'male'/>男   <input type="radio" v-model="gender" value = 'female'/>女   </tr>   <tr>   <th>Chinese</th>   <input type="text" v-model="chinese" />   </tr>   <tr>   <th>Math</th>   <input type="text" v-model="math" />   </tr>   <tr>   <th>English</th>   <input type="text" v-model="english" />   </tr>   </table>   <p>   <button @click="btn\_edit\_stuinfo">Edit to Save</button>   </p>   </div>  </template>  <script>  export default {   name: "StudentEdit",   data() {   return {   msg: "Welcome to Student Management App",   id: "",   name: "",   gender: "",   chinese: "",   math: "",   english: "",   };   },   // 从后端获取指定id的数据   created: function () {   this.id = this.$route.params.id;   try {   console.log('Data retrieved:')   response = this.$axios.get('http://127.0.0.1:8081/edit',{   params: {   id:this.id   }   })   console.log(response.data)   } catch(error) {   console.error('Error fetching data:', error);   return []; // 发生错误时返回空数组   }   // let stuinfo = JSON.parse(localStorage.getItem("stu" + this.id));   // let stuinfo = JSON.parse(localStorage.getItem("stu" + this.id));   // this.name = stuinfo.name;   // this.gender = stuinfo.gender;   // this.chinese = stuinfo.chinese;   // this.math = stuinfo.math;   // this.english = stuinfo.english;   // this.aggregate = stuinfo.aggregate;   // this.average = stuinfo.average;   },   computed:{   verify: function(){   if(!/^\d{5}$/.test(this.id)){//5位数字的正则表达式   console.log(this.id)   return {"flag":false,"msg":'请填写有效的学号(5位数字),例如:10000'}   }   if(!this.gender){   console.log(this.gender)   return {"flag":false,"msg":'性别必须被选择!'}   }   if(this.chinese<0||this.chinese>100){//检查语文成绩是否在0-100   console.log(this.chinese)   return {"flag":false,"msg":'请填写有效的语文成绩(范围是0~100),例如:82'}   }   if(this.math<0||this.math>100){//检查数学成绩是否在0-100   console.log(this.math)   return {"flag":false,"msg":'请填写有效的数学成绩(范围是0~100),例如:82'}   }   if(this.english<0||this.english>100){//检查英语成绩是否在0-100   console.log(this.english)   return {"flag":false,"msg":'请填写有效的英语成绩(范围是0~100),例如:82'}   }   return {"flag":true,"msg":"成功生成新的学生信息"}   },   },   methods: {   editFetchData: async function(ID\_edit) {   console.log('Data fetched from back');   let url = 'http://127.0.0.1:8081/api/'+ ID\_edit.toString()   try {   let response = await this.$axios.get(url);   console.log('Data retrieved:', response.data);   return response.data;   } catch(error) {   console.error('Error fetching data:', error);   }   },   showOriginInfo: function(data){   this.id = data.id;   this.name = data.name;   this.gender = data.gender;   this.chinese = data.chinese;   this.math = data.math;   this.english = data.english;   },   // 把修改后的数据发送给后端   sendDataToBack: function(oStu){   this.$axios({   method:"post",   url:"http://127.0.0.1:8081/edit",   data:oStu,   }).then((res)=>{   console.log("Succsess Post");   console.log(res.data);   })   .catch(function (error) {   console.log(error);   })   this.$router.push({ path: "/info" });   },   btn\_edit\_stuinfo: function () {   // 确保分数为数字类型   let chineseScore = parseFloat(this.chinese);   let mathScore = parseFloat(this.math);   let englishScore = parseFloat(this.english);   let aggregate = chineseScore + mathScore + englishScore;   let average = aggregate / 3.0;   let oStu = {   id: this.id,   name: this.name,   gender: this.gender,   chinese: this.chinese,   math: this.math,   english: this.english,   };   this.sendDataToBack(oStu)   this.$router.push({ path: "/info" });   },   },   created: async function(){   this.id = this.$route.params.id   console.log(this.id)   let data = await this.editFetchData(this.id)   this.showOriginInfo(data)   }  };  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>  h3,  h4 {   font-weight: normal;  }  ul {   list-style-type: none;   padding: 0;  }  li {   display: inline-block;   margin: 0 10px;  }  a {   color: #42b983;  }  input {   width: 128px;  }  p {   font-style: normal;  }  p.p-right {   font-style: italic;   text-align: right;  }  div.divCen {   margin: 2px auto;   width: auto;  }  table {   width: 60%;   margin: auto;   border-collapse: collapse;   border: 0;   text-align: center;  }  th {   background-color: #93daff;   color: #000000;  }  th,  td {   font-size: 0.95em;   text-align: center;   padding: 4px;   border-collapse: collapse;  }  th,  td {   border: 1px solid #c1e9fe;   border-width: 1px 0 1px 0;  }  tr {   border: 1px solid #c1e9fe;  }  tr:nth-child(odd) {   background-color: #dbf2fe;  }  tr:nth-child(even) {   background-color: #fdfdfd;  }  </style> |

1. StudentInsert.vue

该页面实现的是对于新增学生成绩信息记录的功能。该页面需要用户输入学生的id、姓名、性别和各科成绩，同时输入的值要符合设定的要求。具体代码如下：

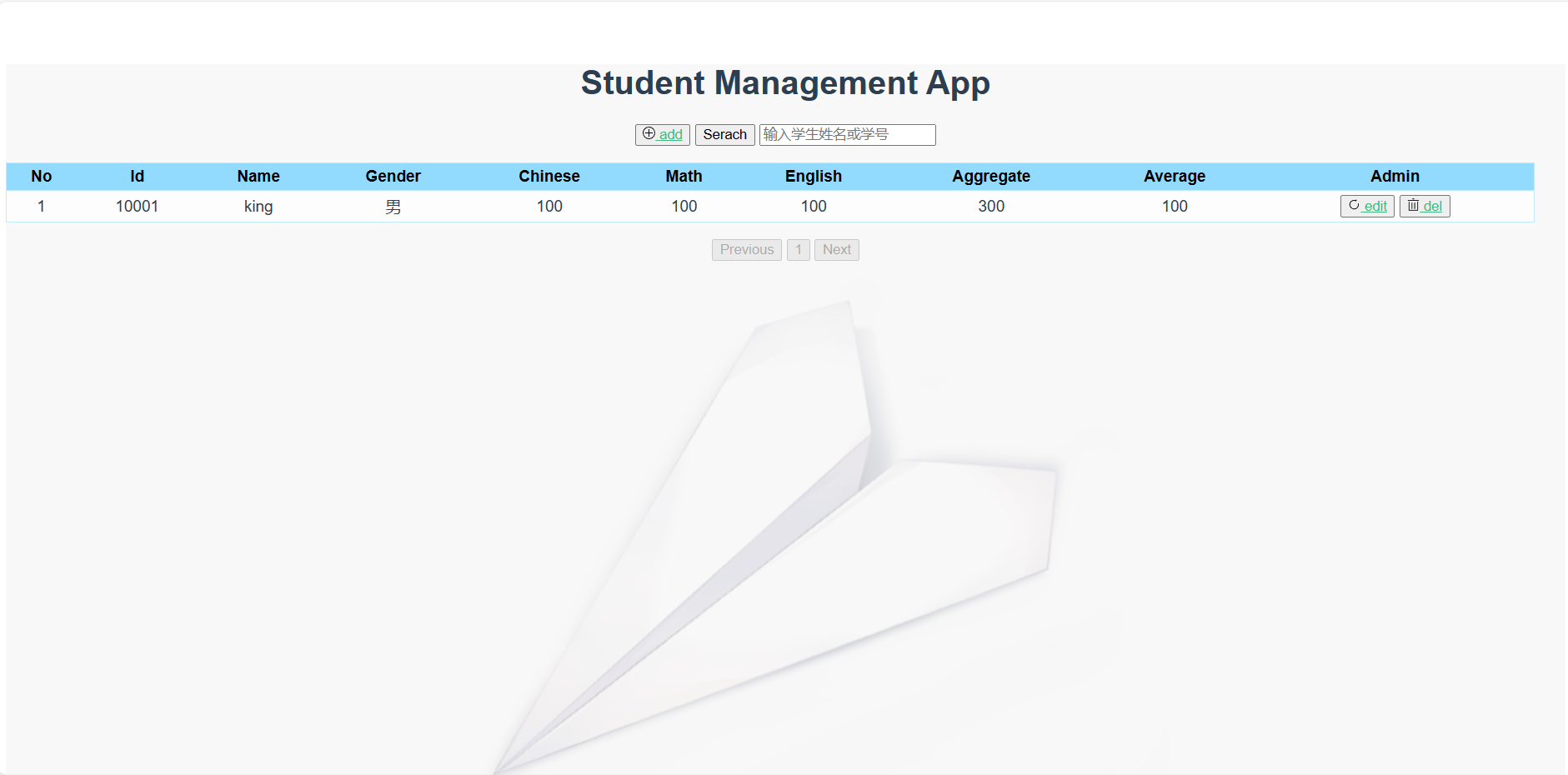
|  |
| --- |
| <template>   <div class="divCen">   <h3>{{ msg }}</h3>   <h4>Student Scores Insert</h4>   <p class="p-right">   <router-link to="/" active-class="active">Back to home</router-link>   </p>   <table>   <tr>   <th>Id</th>   <td>   <input type="text" v-model="id" />   </td>   </tr>   <tr>   <th>Name</th>   <input type="text" v-model="name" />   </tr>   <tr>   <th>Gender</th>   <td><input type="radio" v-model="gender" value = 'male'/><label>男</label>   <input type="radio" v-model="gender" value = 'female'/><label>女</label></td>   </tr>   <tr>   <th>Chinese</th>   <input type="text" v-model="chinese" />   </tr>   <tr>   <th>Math</th>   <input type="text" v-model="math" />   </tr>   <tr>   <th>English</th>   <input type="text" v-model="english" />   </tr>   </table>   <p>   <button @click="btn\_add\_stuinfo">Add to Save</button>   </p>   </div>  </template>  <script>  export default {   name: "StudentInsert",   data() {   return {   msg: "Welcome to Student Management App",   id: "",   name: "",   gender: "",   chinese: "",   math: "",   english: "",   };   },   computed:{   verify: function(){   if(!/^\d{5}$/.test(this.id)){//5位数字的正则表达式   console.log(this.id)   return {"flag":false,"msg":'请填写有效的学号(5位数字),例如:10000'}   }   if(!this.gender){   console.log(this.gender)   return {"falg":false,"msg":'性别必须被选择!除非真的是武装直升机~'}   }   if(this.chinese<0||this.chinese>100||isNaN(this.chinese)){  //检查语文成绩是否在0-100   console.log(this.chinese)   return {"flag":false,"msg":'请填写有效的语文成绩(范围是0~100),例如:82'}   }   if(this.math<0||this.math>100||isNaN(this.math)){  //检查数学成绩是否在0-100   console.log(this.math)   return {"flag":false,"msg":'请填写有效的数学成绩(范围是0~100),例如:82'}   }   if(this.english<0||this.english>100||isNaN(this.english)){  //检查英语成绩是否在0-100   console.log(this.english)   return {"flag":false,"msg":'请填写有效的英语成绩(范围是0~100),例如:82'}   }   return {"flag":true,"msg":"成功生成新的学生信息"}   },   },   methods: {   sendDataToBack: function(oStu){   this.$axios({   method:"post",   url:"http://127.0.0.1:8081/insert",   data:oStu,   }).then((res)=>{   console.log("Success Post:");   console.log(res.data);   this.$router.push({ path: "/info" }); // 转到主页   this.$nextTick(() => { // 确保在DOM更新完成后执行   window.location.reload();   });   })   .catch(function (error) {   console.log(error);   })   },   btn\_add\_stuinfo: function () {   // 确保分数为数字类型   let chineseScore = Number(this.chinese);   let mathScore = Number(this.math);   let englishScore = Number(this.english);   let aggregate = chineseScore + mathScore + englishScore;   let average = aggregate / 3.0;   let oStu = {   id: this.id,   name: this.name,   gender: this.gender,   chinese: this.chinese,   math: this.math,   english: this.english,   aggregate: aggregate,   average: average   };   if(!this.verify.flag){   alert(this.verify.msg)   return ;   }   console.log(oStu)   this.sendDataToBack(oStu)   this.$router.push({ path: "/info" });   },   },  };  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>  h3,  h4 {   font-weight: normal;  }  ul {   list-style-type: none;   padding: 0;  }  li {   display: inline-block;   margin: 0 10px;  }  a {   color: #42b983;  }  input {   width: 128px;  }  p {   font-style: normal;  }  p.p-right {   font-style: italic;   text-align: right;  }  div.divCen {   margin: 2px auto;   width: auto;  }  table {   width: 60%;   margin: auto;   border-collapse: collapse;   border: 0;   text-align: center;  }  th {   background-color: #93daff;   color: #000000;  }  th,  td {   font-size: 0.95em;   text-align: center;   padding: 4px;   border-collapse: collapse;  }  th,  td {   border: 1px solid #c1e9fe;   border-width: 1px 0 1px 0;  }  tr {   border: 1px solid #c1e9fe;  }  tr:nth-child(odd) {   background-color: #dbf2fe;  }  tr:nth-child(even) {   background-color: #fdfdfd;  }  </style> |

4、测试部署

经过前期的开发，我们已经完成了一个可以在本地访问的网站。这里我们将网站部署到各大网站的pages服务中。Pages服务是指用于搭建个人网站的静态站点托管服务。他最大的优点就是免费！使用Github Pages 的好处是平台更大，方便，坏处是速度可能会有些慢、不能被国内的搜索引擎收录；Coding平台的Pages服务配置相较麻烦一点；所以权衡之下，就选择了Gitee平台。

本网站上线后，相关的结果如下所示。

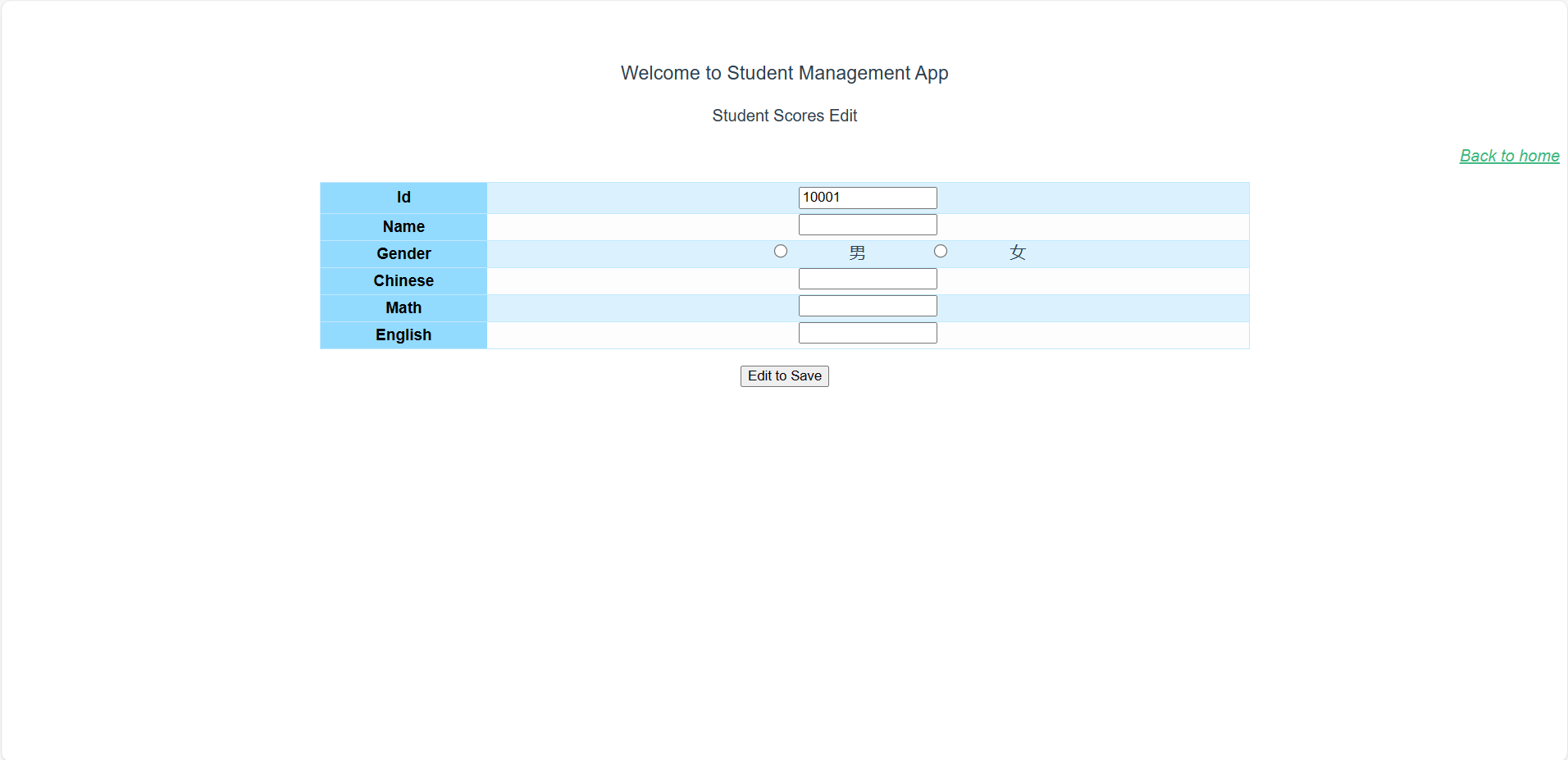
1）网页根目录初始页面

****

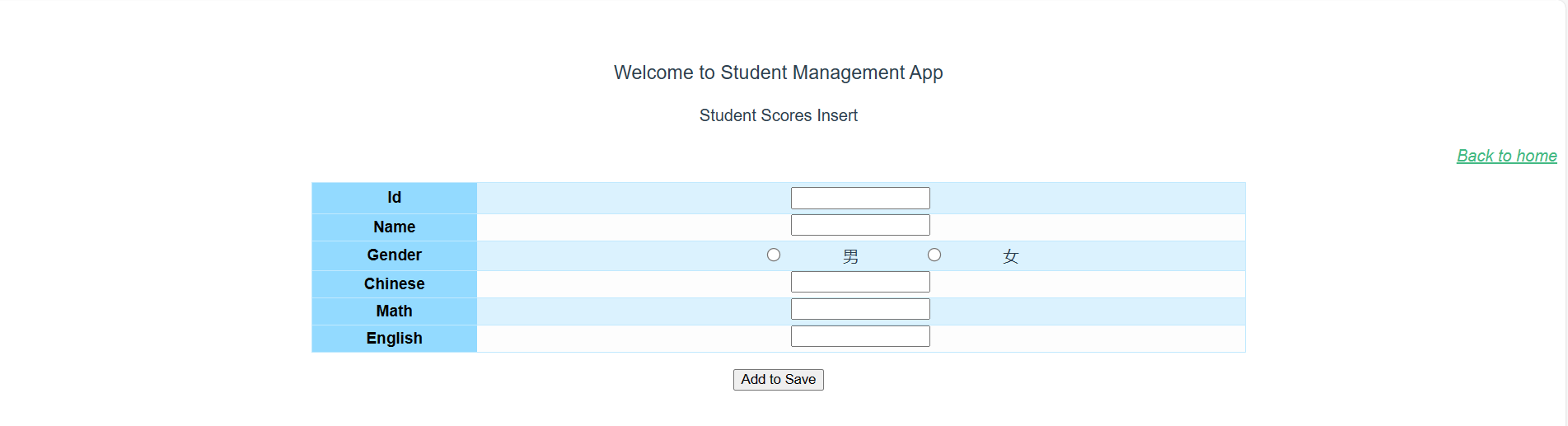
2）学生成绩修改页面

****

页面的URL为localhost：8080/#/edit/+‘选择修改信息的学生的id’。



3）学生成绩录入页面



录入页面异常检查与处理：

1. 检查Id是否为5位数字：

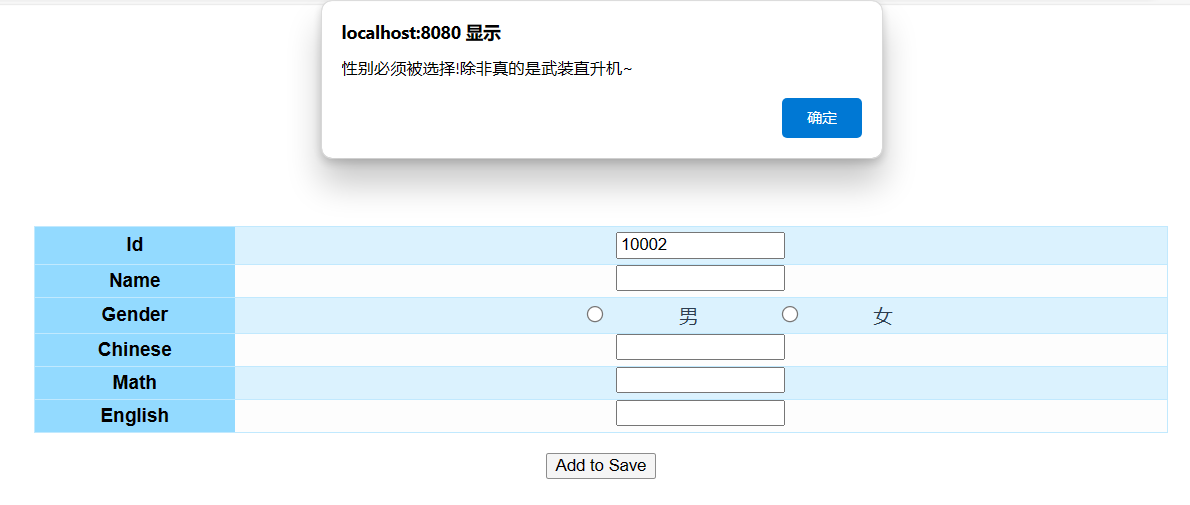


1. 检查各科成绩是否在规定范围内（0~100）：

以语文成绩（chinese）为例：

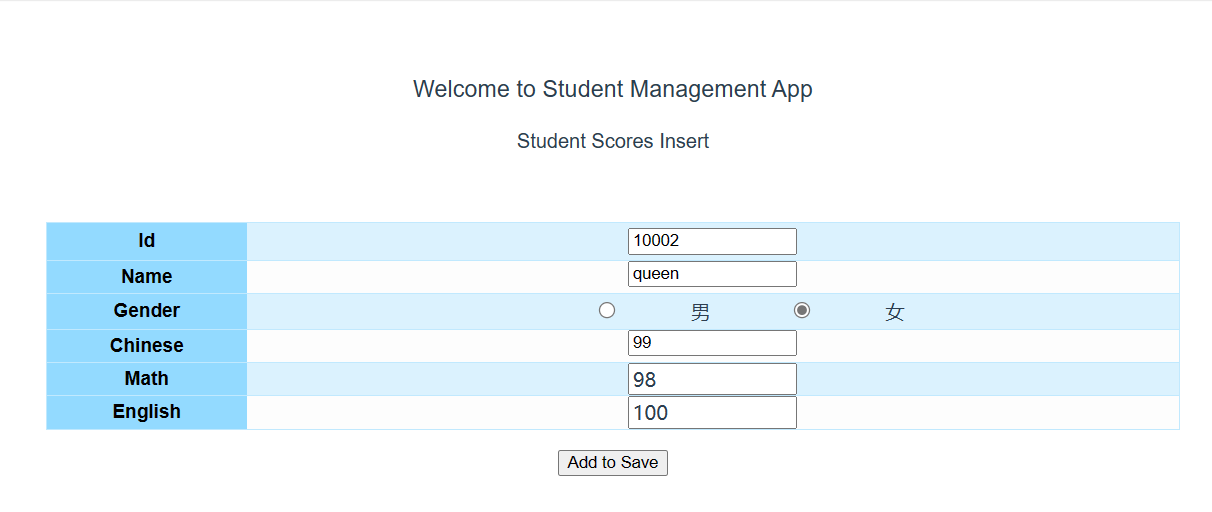


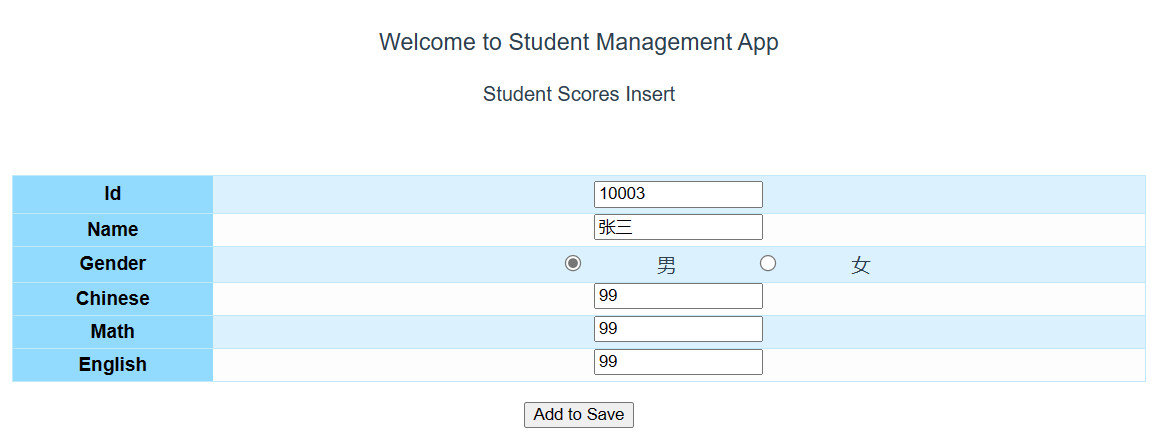
1. 检查性别是否被选择：

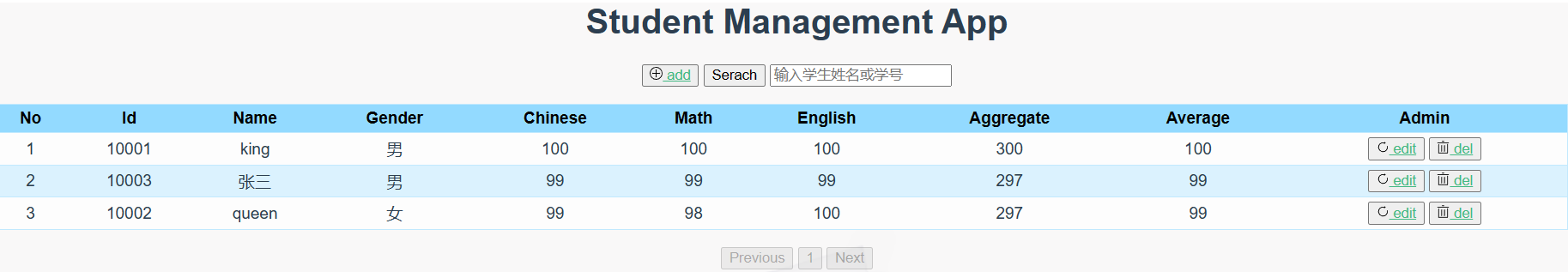


4）录入新记录后的网页根页面

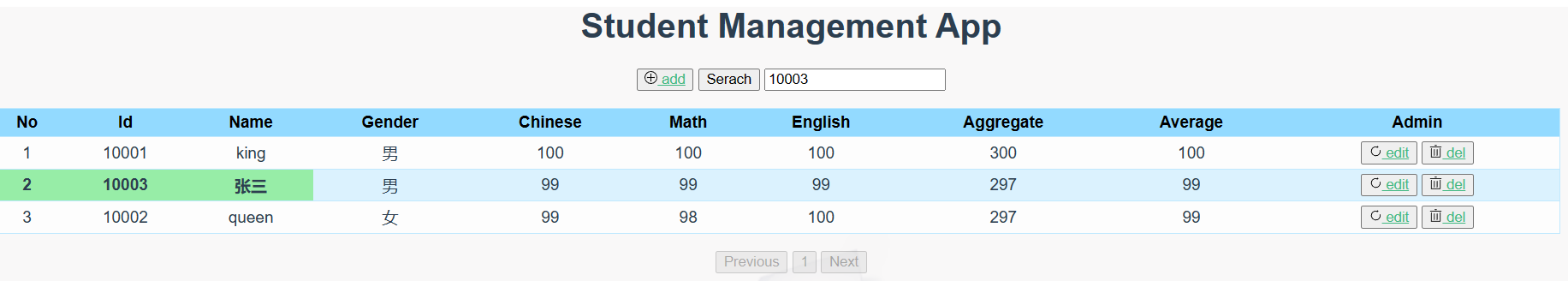
新纪录分别为





插入后成绩按总分降序排序：

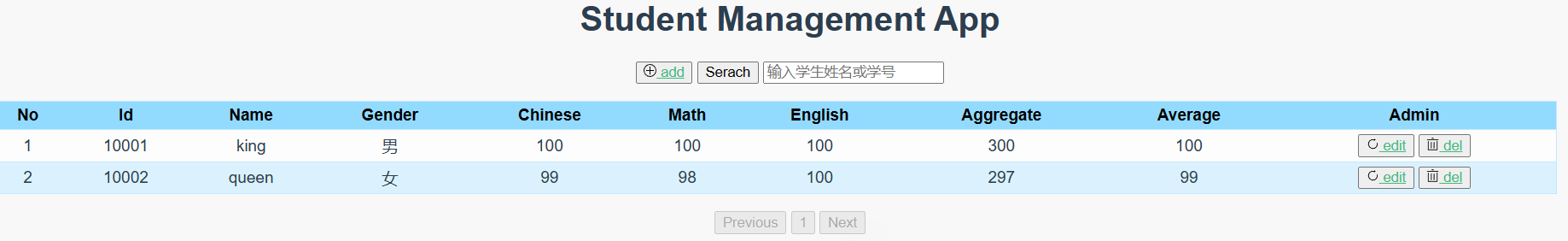
5）搜索指定id的学生的信息



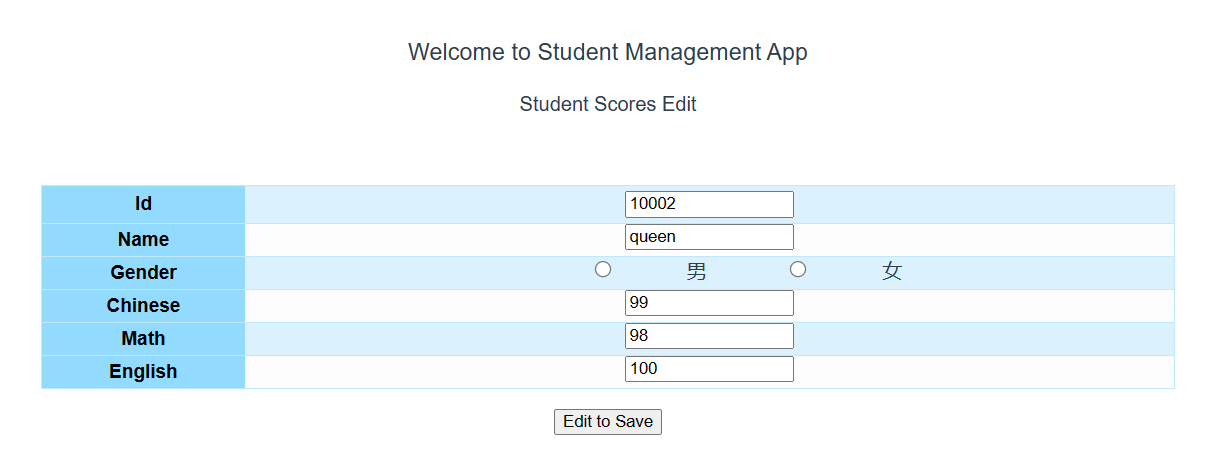
6）删除一条学生信息记录



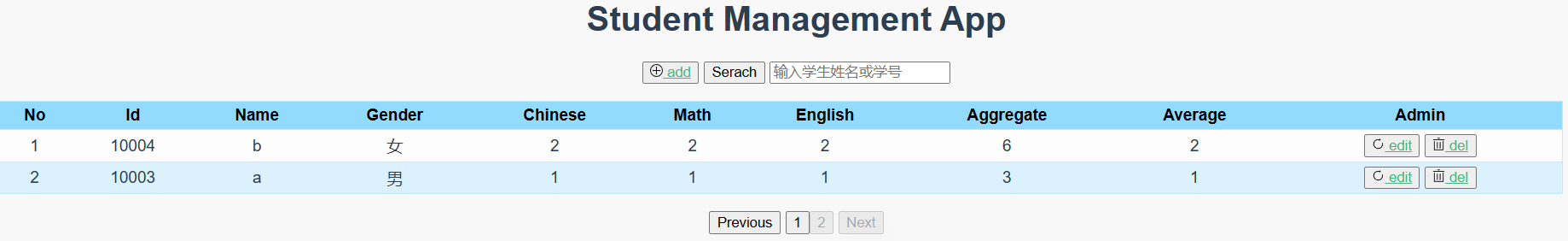
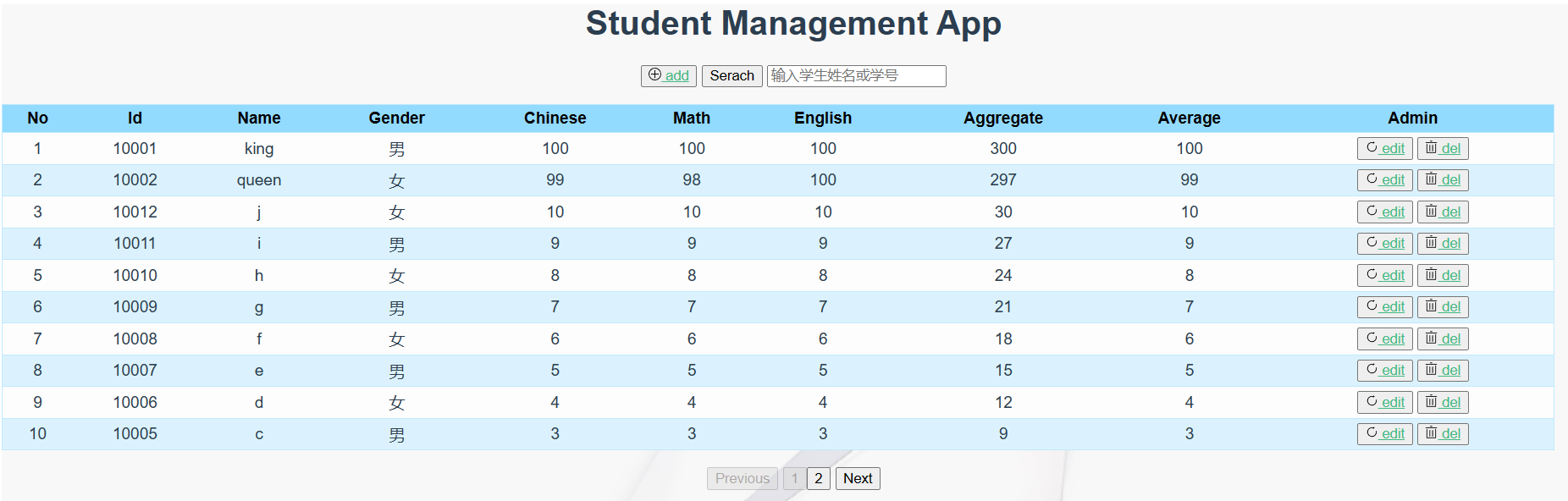
删除操作结果：



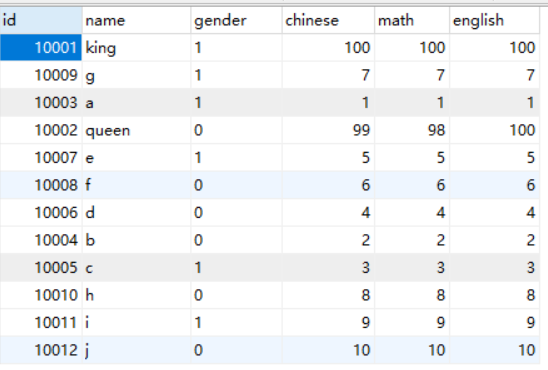
7）修改一个学生的成绩信息



8）成绩展示分页



9）运行后数据库结果



**六、实践心得体会**

1. 在项目的设计和实施过程中，我学到了很多新的技术和知识。从数据库设计到前端页面布局，再到后端逻辑处理，涉及HTML5+CSS+JavaScript+Node+Vue+Express+MySQL+Axios+Git,拓展了我们团队的技术栈。我们不得不学会如何高效地查找资料、学习新工具和技术，借鉴开源项目的经验，并将它们应用到项目中，也成功将我们的项目开源到Github上。这个过程极大地促进了我们的学习成长和编程能力的提高。
2. 本次实验也有诸多不足，比如在开发过程中，并没有及时更新我们的开发文档，没有做到敏捷开发，前端页面交互设计不足，后端数据库不够丰富等等，这些都需要在不断地确认需求与开发中不断改进，迫于时间有限，没有进行进一步开发了。