《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
汪锦琛	2022010801015	A
朱若愚	2022150501027	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；

2) 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. 实现多名学生相关数据的存储，包括该学生的姓名、学号、性别以及各科成绩等。
2. 实现对单名学生数据的一系列运算，包括统计成绩总分，计算平均分等。
3. 实现对存储数据的一系列操作，包括查找、修改和删除等，并且可以以某个字段对数据进行排序操作。
4. 直观、明确的在前端网页中展示数据，并且允许通过提供的功能按钮实现包括上述在内的一系列操作。

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。

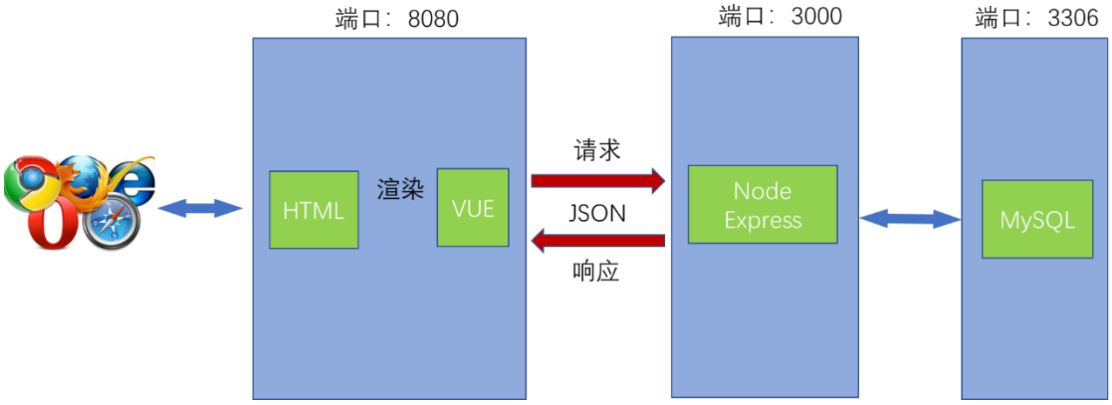


图 1 系统整体技术架构

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：

名	类型	长度	小数点	允许空值 (	
id	int	10	0	☑	🔑1
name	char	20	0	☑	
gender	char	8	0	☑	
chinese	tinyint	4	0	☑	
math	tinyint	4	0	☑	
english	tinyint	4	0	☑	

各字段的含义如下

1. id：学生的学号，以 int 类型进行存储，int 代表有符号整数，该字段长度为 10.
2. name：学生的姓名，以 char 类型进行存储，char 型用数字编码来储存字符，该字段长度为 20.
3. gender：学生的性别，以 char 类型进行存储，char 型用数字编码来储存字符，该字段长度为 8.
4. chinese、math、english：学生的语文、数学、英语成绩，以 tinyint 类型进行存储，tinyint 占用一字节储存整数，该字段长度为 4.

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*/
```

2）选择展示对应 ID 的学生成绩

方法 URL：localhost:8081/list_user

输入参数：JSON 格式如下

```
"id": 10001,  /*学号*/
```

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*/
```

3）增加一个学生的成绩

方法 URL：localhost:8081/insert_user

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/update_user

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_user

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”

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：

express scoreBack

npm install

3）配置数据库

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

```
const mysql = require('mysql')
module.exports = {
//数据库配置 config: { host: 'localhost', port: '3306', user: '*****', password: '*****',
database: '*****' },
```

```
//连接数据库
//连接池方法
sqlConnect: function (sql, sqlArr, callBack) { var pool =
mysql.createPool(this.config) pool.getConnection((err, conn) => { if (err)
{ console.log('连接失败') return }
//事件驱动回调
conn.query(sql, sqlArr, callBack) conn.release() }) },
//异步连接池
//promise 回调
SySqlConnect: function (sySql, sqlArr) { return new Promise((resolve, reject) =>
{ var pool = mysql.createPool(this.config) pool.getConnection((err, conn) => { if (err)
{ reject(err) } else { conn.query(sySql, sqlArr,(err,data)=>{ if(err){ reject(err) }
else{ resolve(data) } }) conn.release() } }) }).catch((err)=>{ console.log(err) }) } }
```

4 创建 api 方法

修改 Controller/scoreController.js 文件，创建一个后端 API。具体代码如下：

```
var dbConfig = require('../utils/dbconfig')
//获取成绩
getScore = (req, res) => {
   var sql = "select * from stuscore"
   var sqlArr = []
   var callBack = (err, data) => {
      if (err) {
         console.log('链接出错')       }
      else {
         res.send({
            'list': data          })       }    }
   dbConfig.sqlConnect(sql, sqlArr, callBack)}
//写好的方法记得暴露出去
module.exports = {
         getScore }
//在 index.js 路由中
var score = require('../controllers/ scoreController ')
/* GET home page. */
router.get('/score', score. getScore);
```

5) 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

2、修改后端 express_index.js 源代码文件

a) 在文件开始处引入相关依赖

var cors=require('cors')

b) 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 cors = require('cors');
var app = express();
app.use("/public", express.static("public"));
app.use(cors);
```

3、修改后端 express_index.js Header 设置

a) 修改 header 的"Access-Contro-Allow-Origin"
添加为将允许访问的前端网址和端口

b) 修改 header 的"Access-Control-Allow-Credentials"

c) 注意"Content-Type"的默认属性，以防跨域时出现的不统一问题，

最后的代码如图所示：

```
app.use((req, res, next) => {
    res.header('Access-Control-Allow-Origin', 'http://localhost:8082')
    res.header('Access-Control-Allow-Credentials', 'true');
    next();
})
app.options('*', (req, res) => {
    res.header('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE');
    res.header('Access-Control-Allow-Headers', 'Content-Type');
});
```

2.4 前端设计

前端采用 Vue.js 结合 axios 来实现前后端的跨域、JSON 数据交互和用户交互，包括以下步骤：

1） 安装 Vue.js 组件

使用命令如下：

npm install vue -g

npm install webpack-cli -g

npm install vue-cli -g

npm install vue-router -g

2） 创建 Vue 项目

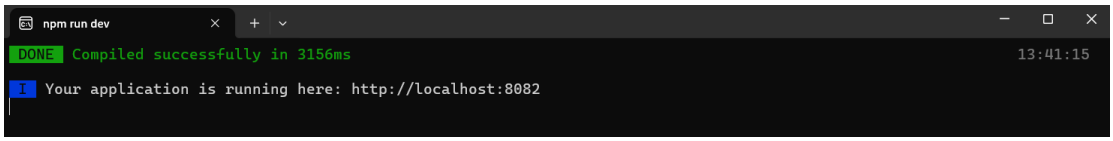
a) 创建基于 webpack 模板的 vue 应用程序项目

vue init webpack vuestu

b) 初始化测试项目

npm run dev

c) 访问前端网址



3） 利用 HBuilderX 编写前端程序

a) 引入 axios 库

i. 安装

npm install axios

ii. 在主文件中引入

```
import axios from 'axios'
import Qs from 'qs'
axios.defaults.withCredentials = true;
Vue.prototype.$axios = axios
Vue.prototype.$qs = Qs
axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded'
```

b) 确定路由

在 router 下目录中，编写 index.js，完成对三个子目录：”StudentInfo”,”StudentInsert”,”StudentEdit”的路由。

```
// 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
  }]
})
```

c) 使用 axios 组件完善对应页面功能

以 StudentInfo 为例

```
created() {
  this.$axios.get("http://127.0.0.1:8081")
  .then((response) => {
    console.log("从接口获取数据: ");
    console.log(response.data);
    let len = response.data.length;
    let stuinfo;
    for(let i = 0;i < len;i++){
      stuinfo = response.data[i];
      stuinfo.sum = Number(stuinfo.chinese)+Number(stuinfo.english)+Number(stuinfo.math);
      stuinfo.average = Math.round(stuinfo.sum/3);
      this.stu.push(stuinfo);
    }
  })
  this.stu.sort((b,a) => a.average - b.average);
  console.log("stu:",this.stu);
},
watch:{
  $route(to,from){
    this.$router.go()
  },
},
methods: {
  del_stu_info: function (stuid) {
    let params = new URLSearchParams();
    params.append("id",String(stuid));
    this.$axios.get("http://127.0.0.1:8081/del_user",{params:params})
    .then((response) => {
      console.log("数据已删除");
```

以上代码完成了在每次进入 Info 界面时，获取全体学生的成绩，并进行按照学号排序的工作后输出到 UI 中；同时还包含了进入单个学生成绩修改界面和删除单个学生成绩的按钮，过程使用 axios 组件实现。因为后端设置获取 params，使用 params 传输数据避免跨域问题。

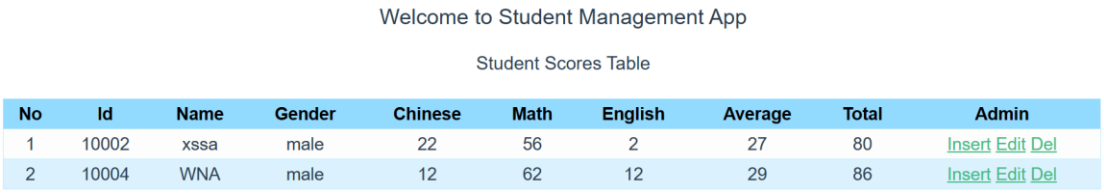
d) 跨域问题解决

在前后段跨域问题上，需要注意的一点是，只有符合同源策略的环境才能正常运行，很显然我们当前在本地开发的前后端不满足这个条件，所以我们需要修改配置文件，挂载一个虚拟的域名，在之中再申请允许跨域：

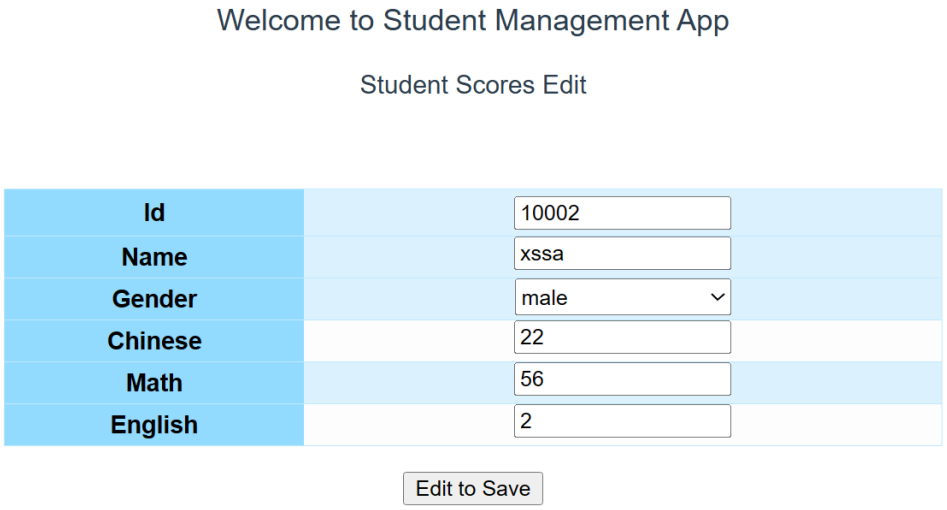
```
dev: {

  // Paths
  assetsSubDirectory: 'static',
  assetsPublicPath: '/',
  proxyTable: {
    '/api':{
      target:"http://localhost:8080",
      changeOrigin: true,
      pathRewrite:{
        '^/api':'/api'
      }
    }
```

4） 测试前后端口连接

首先，在进入页面时，前端正确获取了后端的学生成绩数据并展示：



然后，进一步测试修改学生成绩：



可以注意到前端可以直接获取后端单个学生数据并预先填入选项中，关于选项的输入限制，完成以下要求：

1. Id 必须为五位数字，否则不能提交；

2. 名字不能为空，否则不能提交；

3. 性别在选择之前为空，当性别为空时不能提交；

4. 各项成绩为 0-100 之间的整数，否则不能提交；

5. 当成绩栏什么都不填写时，默认该项成绩为 0 分；

修改英语成绩为 35 后：

No	Id	Name	Gender	Chinese	Math	English	Average	Total	Admin
1	10002	xssa	male	22	56	35	38	113	Insert Edit Del
2	10004	WNA	male	12	62	12	29	86	Insert Edit Del

主页面信息及数据库信息发生改变

进一步，测试删除和插入功能

将一号学生删除：

Welcome to Student Management App

Student Scores Table

No	Id	Name	Gender	Chinese	Math	English	Average	Total	Admin
1	10004	WNA	male	12	62	12	29	86	Insert Edit Del

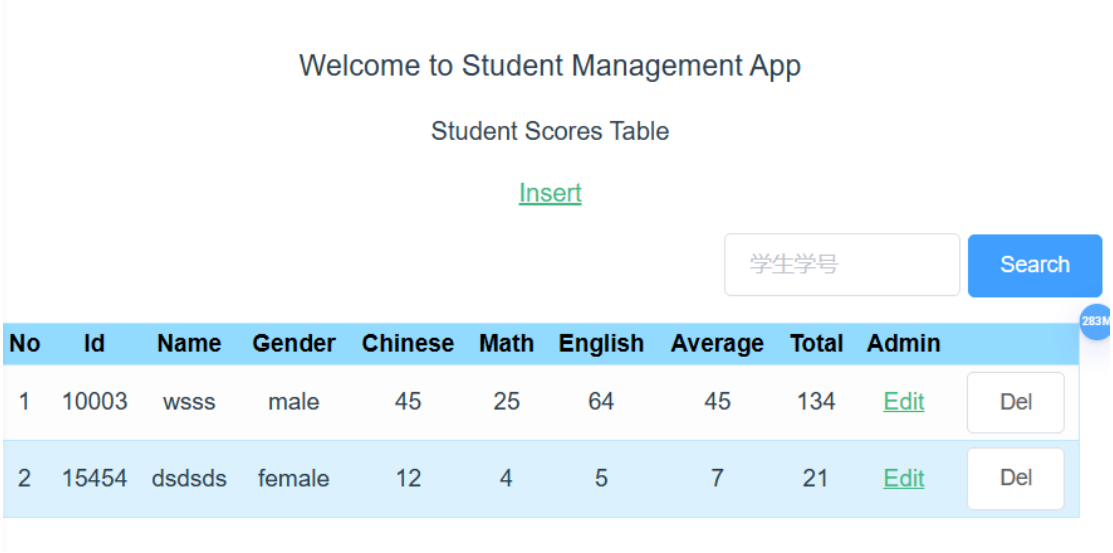
插入新的 10002 号学生：

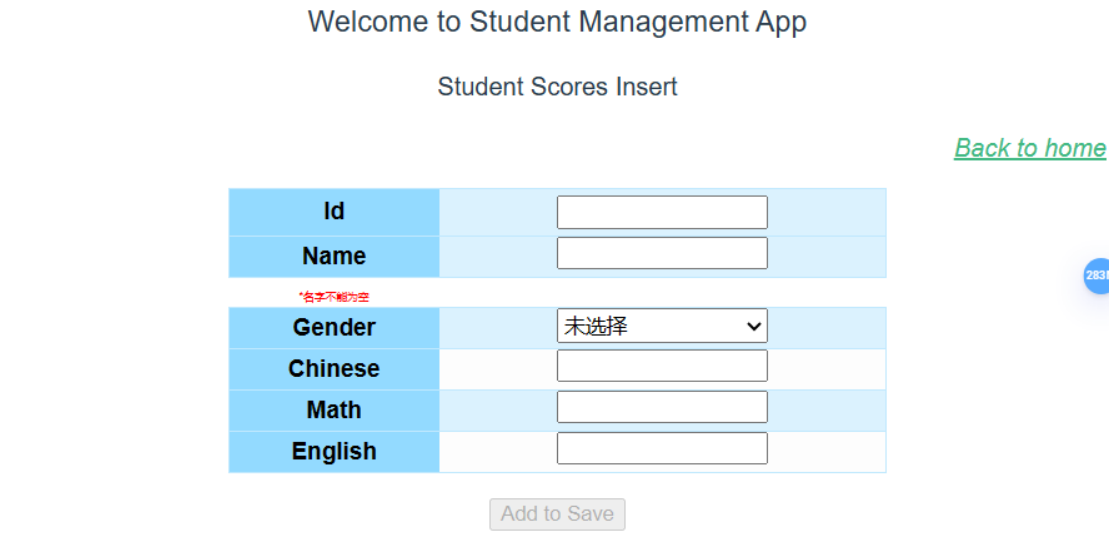
No	Id	Name	Gender	Chinese	Math	English	Average	Total	Admin
1	10004	WNA	male	12	62	12	29	86	Insert Edit Del
2	10003	wsss	male	45	25	64	45	134	Insert Edit Del

3、扩展任务实现

1. 页面整体风格以布局的修改

修改了默认的 css 样式，使得整体的页面布局更加合理，并且引入了 element-ui 组件使得局部的功能展示以及输出输入更加流畅：





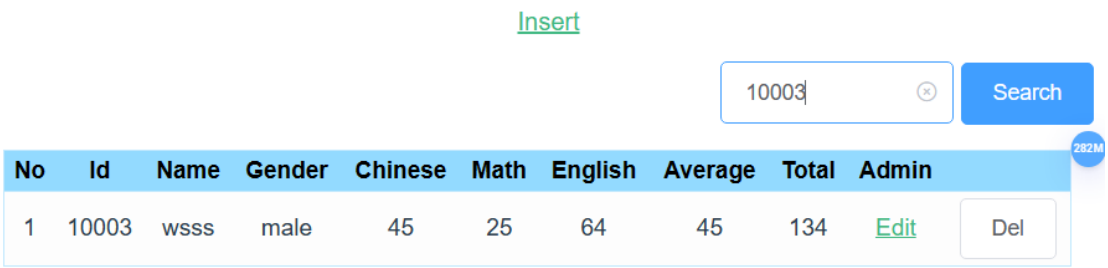
按照要求修改了 Insert 位置

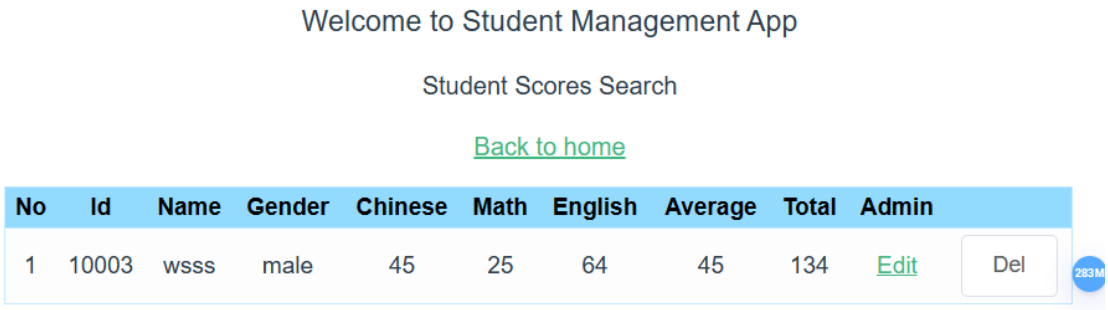
2. 添加了删除提示

在用户点击 del 按钮之后，页面会弹出提示，询问是否删除如果点击确定，则会删除该用户信息，若点击取消，则会刷新当前页面，不删除用户信息：

3. 增添搜索功能

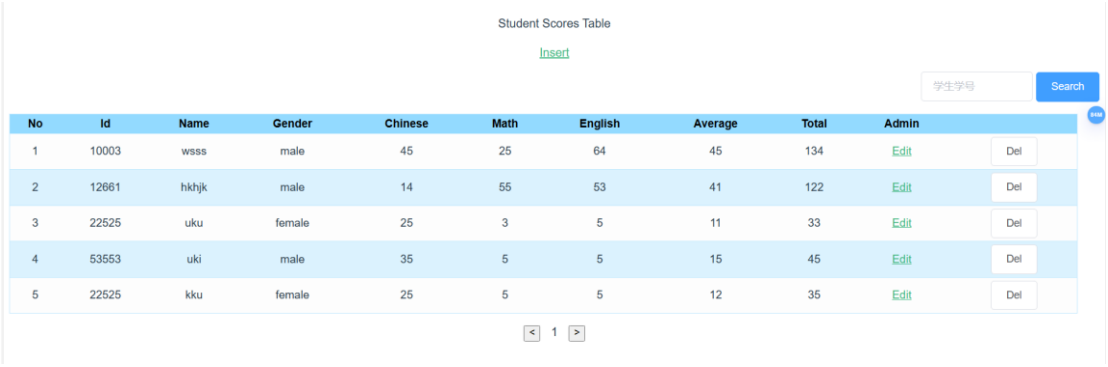
在主页面上，用户可以通过输入学生的学号来实现对单一学生成绩的搜索，当该学号下不存在学生，且该学号不违法的话，则会询问用户是否要创建该学生的的成绩信息，并跳转到对应 Insert 页面：





4. 记录翻页功能

以五页为一个界限，可以实现数据的自动翻页分页：



六、实践心得体会

1. 网页前端可以直观明了展示内容，并通过各种按钮以接口形式调用后端相关函数功能。通过各大开发平台的 Pages 服务上传源码后进行部署，可以很好的实现个人网站的部署和模拟。
2. 数据库后端能够相对容易地存储批量数据，并通过函数实现的一系列功能对其进行查找、排序、增加、修改和删除等操作。通过接口 API 与前端相连接响应相应功能。
3. 网页前端可以使用 CSS 与 html 配合，以实现对网页的排版和进一步的美化，方便使用者更加清楚直接了解网页所展示的内容和进行进一步的操作。