普通高等学校信息查询平台

数据库原理课程设计报告

学　　号 _____　　____

姓　　名 _____　　____

专　　业 _____　　_____

授课老师 _____　　_____

目录

摘 要 目前国内对教育的重视程度不断加深，尤其是高等教育。然而，网络上的信息纷繁复杂，学生寻找信息会遇到很多困难，甚至查找到错误的信息。并且目前市面上缺少对学校的系统的评分与评价，主要是以口口相传为主，这样就会存在信息闭塞的问题。本次课程设计即为开发一个普通高等学校信息查询平台，可以查询学校信息,检索学校,以及进行评价等等。本次实验报告为最终报告设计，包含需求分析以及可行性分析、概念分析、逻辑设计、具体实现、成果展示、项目总结。

关键词：数据库；高等教育；信息查询

一.概述

1.1 课题背景

本次数据库应用设计是设计了一个普通高等学校信息查询平台，为用户提供了一个查询信息的平台，并可对学校进行评价。学校也可以通过此平台发布通知。

1.2 编写目的

目前国内对教育的重视程度不断加深，尤其是高等教育。然而，网络上的信息纷繁复杂，学生寻找信息会遇到很多困难，甚至查找到错误的信息。并且目前市面上缺少对学校的系统的评分与评价，主要是以口口相传为主，这样就会存在信息闭塞的问题。本设计就是为了解决此问题，使得信息公开、透明。

二.需求分析

2.1 功能需求

平台的核心功能主要是用户进行信息检索、发表评论与学校发布通知。用户能够在平台上检索学校，查看学校信息，并且进行评论。此外，平台还需要良好的可扩展性，能够根据实际需求完善设计，进一步增加功能。学校可以在平台上发布通知，使得用户可以及时了解最新信息。

2.2 数据字典

用户

属性	类型	属性描述
ID	数值	用户的 ID 号，作为平台账号的标识
姓名	字符串	用户的姓名信息
密码	字符串	用户的平台密码

学校

属性	类型	属性描述
SID	int	学校的 ID 号
学校名称	text	学校的名称
学校标识码	Bigiint	学校的标识码
主管部门	Text	学校的主管部门
所在地	Text	学校的所在地
办学层次	Text	学校是本科还是专科

备注	Text	是民办则写在备注里
注册码	Varchar	单独告知学校的注册码，可用于注册管理员

评论

属性	类型	属性描述
ID	数值	评论的 ID 号
发布者 ID	数值	发布评论的用户 ID
评论学校 ID	数值	被评论的学校 ID
评论内容	字符串	评论的内容
发布时间	时间	评论的发布时间

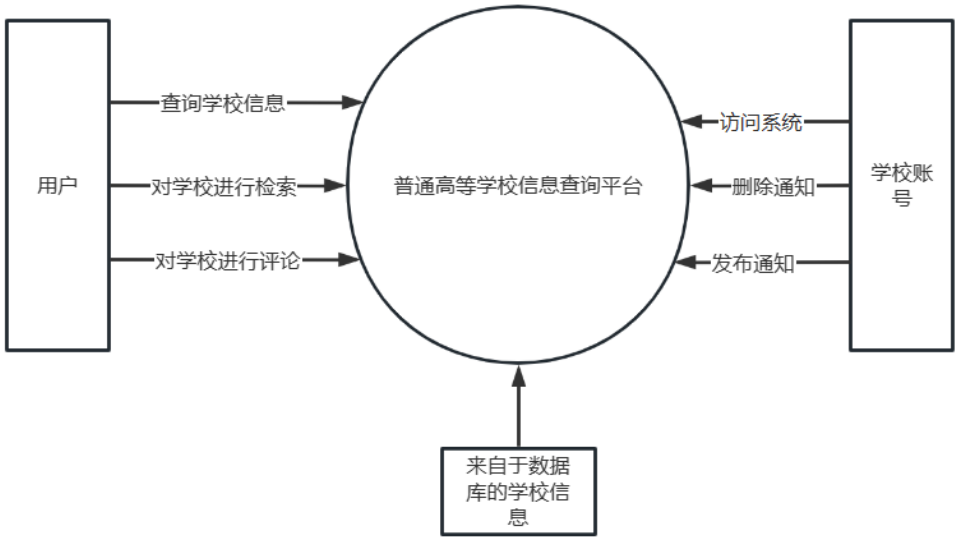
通知

属性	类型	属性描述
ID	数值	通知的 ID 号
学校 ID	数值	发布通知的学校 ID
通知大标题	字符串	通知的大标题
通知内容	字符串	通知的内容
通知时间	时间	通知的发布时间

学校管理员

属性	类型	属性描述
ID	数值	管理员的 ID 号，作为平台账号的标识
姓名	字符串	管理员的姓名信息
密码	字符串	管理员的平台密码
学校 ID	数值	管理员所在学校的 id

2.3 数据流图



三.可行性分析

3.1 技术可行性

项目在技术上较为成熟，信息查询平台有着许多完整的开发与解决方案，同时 在数据存储上以高数据量为主，不涉及复杂的数据库模型设计。在对各种模型经过充分比较后，选定了 Web 应用作为方案，可以提升平台访问的便捷性，使其支持多种设备。除此之外，本课设中可将前端与后端分开建构，使得其更具有可靠性与可读性。

3.2 应用可行性

本项目的背景是目前国内对教育的重视程度不断加深，尤其是高等教育。然而，网络上的信息纷繁复杂，学生寻找信息会遇到很多困难，甚至查找到错误的信息。并且目前市面上缺少对学校的系统的评分与评价，主要是以口口相传为主，这样就会存在信息闭塞的问题。本次数据库应用设计是设计了一个普通高等学校信息查询平台，为用户提供了一个查询信息的平台，并可对学校进行评价。学校也可以通过此平台发布通知。本设计可以解决信息闭塞的问题，使得信息公开、透明，在应用上具有可行性。

四.概念设计

4.1 实体

1.用户实体

实体描述：用户是本数据库的基本实体之一，表示信息查询平台的使用者。主码为用户 ID 号，可以对用户进行唯一区分。

属性说明：

属性	字段名	类型	属性描述
ID	Uid	数值	用户的 ID 号，作为平台账号的标识
姓名	Uname	字符串	用户的姓名信息
密码	Psw	字符串	用户的平台密码

2.学校实体

实体描述：学校信息是本数据库的基本实体之一，表示学校的信息以及也是另一种层面上的用户。主码为学校 ID 号，可以对学校进行唯一区分。

属性说明：

属性	字段名	类型	属性描述
SID	Sid	int	学校的 ID 号
学校名称	Xxmc	text	学校的名称
学校标识码	Xxbsm	Bigint	学校的标识码
主管部门	Zgbm	Text	学校的主管部门
所在地	Szd	Text	学校的所在地
办学层次	Bxcc	Text	学校是本科还是专科
备注	Bz	Text	是民办则写在备注里
注册码	Zcm	Varchar	单独告知学校的注册码，可用于注册管理员

3. 评论实体

实体描述：用户可以对学校进行评论，每个评论都会自动生成编号作为主码。

属性说明：

属性	字段名	类型	属性描述
ID	Cid	数值	评论的 ID 号
发布者 ID	Uid	数值	发布评论的用户 ID，外键
评论学校 ID	Sid	数值	被评论的学校 ID,外键
评论内容	Ccontent	字符串	评论的内容
发布时间	Ctime	时间	评论的发布时间
发布者姓名	Uname	字符串	发布评论的用户名

4. 通知

实体描述：学校可以发布通知，每个通知都会自动生成编号作为主码。

属性说明：

属性	字段名	类型	属性描述
ID	Nid	数值	通知的 ID 号
学校 ID	Sid	数值	发布通知的学校 ID，外键
通知大标题	Nheader	字符串	通知的大标题
通知内容	Ncontent	字符串	通知的内容
通知时间	Ntime	时间	通知的发布时间

5. 学校管理员

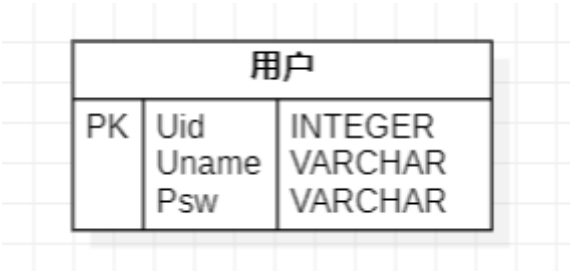
实体描述：学校的管理员，按照注册码进行注册。

属性说明：

属性	字段名	类型	属性描述
ID	Mid	数值	管理员的 ID 号，作为平台账号的标识
姓名	Mname	字符串	管理员的姓名信息
密码	Mpassword	字符串	管理员的平台密码
学校 ID	Sid	数值	管理员所在学校的 id

4.2 实体属性局部 E-R 图

1.用户



2.学校

3.评论

评论		
PK	Cid	INTEGER
FK	Uid	INTEGER
FK	Sid	INTEGER
	Ccontent	VARCHAR
	Ctime	TIME
	Uname	VARCHAR

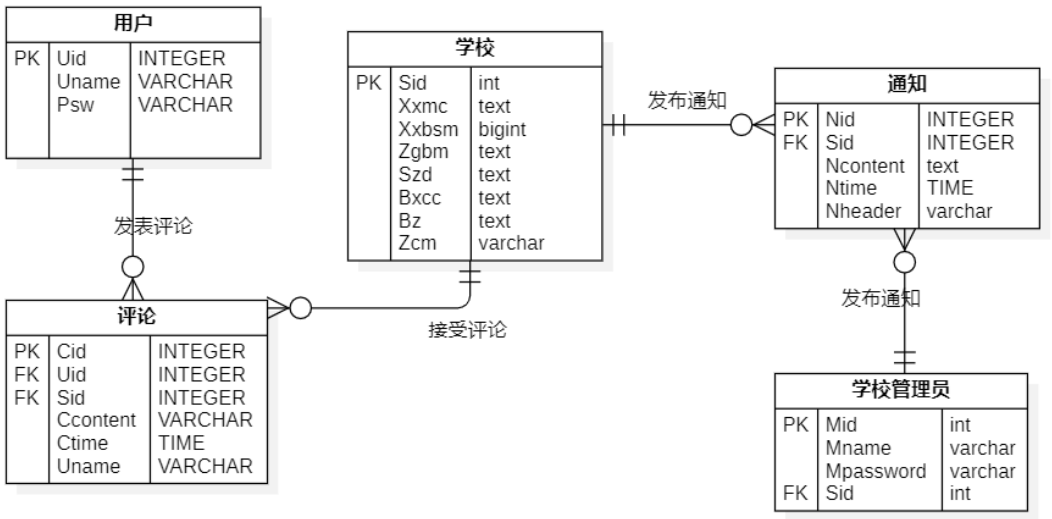
4.通知

通知		
PK	Nid	INTEGER
FK	Sid	INTEGER
	Ncontent	text
	Ntime	TIME
	Nheader	varchar

5.学校管理员

学校管理员		
PK	Mid	int
	Mname	varchar
	Mpassword	varchar
FK	Sid	int

4.3 全局 E-R 图



对概念设计中的关系进行说明：
一个用户可以发表多条评论；
一个学校可以对应多条通知；
一个学校可以收到多条评论；

一个管理员可以发布多条通知。

五.逻辑设计

5.1 E-R 图向关系模型的转变

在完成数据库的 E-R 模型设计后，需要将 E-R 图按照一定规则转化为相应的关系模式并合并冗余，得到对应表单。在将 E-R 模型转化为关系模式时，主要遵循如下规则：

① 检查所有强实体集，为其建立对应的关系模式，对应关系都包含实体集的所有属性，且主码与实体集相同。

② 检查所有弱实体集，使用弱实体集及其依赖强实体集的主码属性建立对应的关系模式。

③ 检查所有联系集，依据映射基数决定是否建立单独关系。若映射为一对一或一对多，则在其中一项实体集的关系中添加另一项实体集（应当为“一”）的主码属性；若为多对多，则需要为该联系建立一个新关系，关系的两个属性分别为两个实体集的主码属性。

在设计的 E-R 模型中，所有实体集均为强实体集，因此可以构造如下关系模式：

用户：User (Uid , Uname , Psw)

学校：School (Sid , Lx , Xxmc , Jbzqk , Bs , Spsw)

评论：Comment (Cid , Uid , Sid , Ccontent , Ctime)

通知：Notice (Nid , Sid , Ncontent , Ntime)

管理员：SchoolManager(Mid，Sid，Mname，Mpassword)

联系集：发表评论（Post）,发表通知（Inform）均为一对多，所添加的属性均已在逻辑设计中添加到实体集中，不需进行额外操作。

5.2 数据模型的优化及规范化设计

将 E-R 模型转化为关系模式后，还需对得到的关系模式进行规范化，使其满足 3NF 范式。3NF 范式消除了关系中的传递依赖，可以减少数据库中的信息冗余。规范化时依据 3NF 定义对所有关系模式进行检查，若不满足 3NF 范式，则按照分解规则对其进行分解。可以证明，3NF 分解是无损分解，而且能够保持依赖。

为了保证安全性，从 School 关系中，额外分解出 School_password(Sid , Spsw)，实现对密码的单独存放。

User 表：

函数依赖集为 Uid→ Uname , Psw 。符合 3NF 范式。

School 表：

函数依赖集为 Sid→ Lx , Xxmc , Jbzqk , Bs。符合 3NF 范式。

Comment 表：

函数依赖集为 Cid→Uid , Sid , Ccontent , Ctime。符合 3NF 范式。

Notice 表：

函数依赖集为 Nid→Sid , Ncontent , Ntime。符合 3NF 范式。

SchoolManager 表：

函数依赖集为 Mid→Mname,Mpassword,Sid。符合 3NF 范式。

六.项目管理

6.1 总体架构

本次课程设计的开发形式是 Web 应用形式，这是一种便于用户访问与跨平台移植的开发形式，用户可以通过访问网站同系统进行交互。

本项目中应用系统的实现方式还进行了分层的设计，使用了前端-后端-数据库的三级架构。其中前端以网页形式加载后，运行在用户的主机上，负责实现同用户的具体交互，并且处理一些简单的运算逻辑；后端运行在远程服务器上，负责处理前端发出的请求，控制数据库进行查询，并且管理系统的运行逻辑；数据库运行在专用的数据库服务器上，负责存储数据并对数据进行直接操作，是本次课设的重点。

当前端需要获取数据时，向后端发送 HTTP 请求，后端处理后将结果回传，即完成了交互操作。后端与数据库通过 SQL 请求进行交互，后端对前端发来的请求进行分析，从中提取出需要对数据进行的操作，以 SQL 语句形式发往数据库执行。

在本次的课设中，前端是使用 Vue 完成的，后端使用了 Nodejs 中的 express，数据库部分使用了 MySQL。

前端-后端-数据库的分级架构是 Web 开发过程中的常用模式，因此有着较多成熟的技术范例，同时每一部分的任务明确，开发过程中遇到的障碍较低。这种方式还能够降低模块间的耦合度。提升应用系统的可维护性。

6.2 前端部分

前端部分是基于 VUE 完成的，主要是注重与用户之间的交互，设计一个美观的页面，以增加用户的使用体验感。除了美观之外，还考虑到了人们的习惯，添加了很多人性化的设计，方便人们使用。Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。选择 VUE 作为开发工具的原因是 vue 可以根据数据的变化进行实时渲染，并将视图各部分切分为功能组件分层实现。对于这样一个全栈开发而言，也会更方便快捷。

6.3 后端部分

后端部分基于 express 完成，实现了前后端的分离，主要是负责处理前端发出的请求，控制数据库进行查询，并且管理系统的运行逻辑。Express 是最流行的 Node 框架，是许多其他流行 Node 框架 的底层库。它提供了以下机制：1.为不同 URL 路径中使用不同 HTTP 动词的请求（路由）编写处理程序。2.集成了“视图”渲染引擎，以便通过将数据插入模板来生成响应。3.设置常见 web 应用设置，比如用于连接的端口，以及渲染响应模板的位置。4.在请求处理管道的任何位置添加额外的请求处理“中间件”。

6.4 数据库部分

数据库部分采用的是 MySQL，并且用到了它的 workbench。这一部分主要就是负责根据数据库的逻辑设计建立表单，存储数据，并且根据后端请求对数据进行操作。

数据库按照数据结构来组织、存储和管理数据的仓库。每个数据库都有一个或多个不同的 API 用于创建，访问，管理，搜索和复制所保存的数据。MySQL 是最流行的关系型数据库管理系统。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

6.5 开发环境

运行环境：Windows10 x64
运用软件：VSCode、MySQL8.0、 Vue2、 Nodejs
浏览器：Microsoft Edge

七.系统实现

7.1 系统架构搭建

7.1.1 前端搭建

前端是基于 Vue2 完成的，具体操作为，下载安装 Node. js 之后，打开 cmd 输入：

Npm install vue-cli -g ： 下载 vue-cli 脚手架

Vue init webpack xxx ： 创建项目

7.1.2 后端搭建

首先在前端搭建好框架之后，要再安装后端用到的模块和依赖：

Npm install vue-resource：安装 vue-resource 依赖

Npm install express mysql body-parser：安装模块和依赖

打开创建好的项目后，在项目根目录文件下新建 server 文件夹，在文件夹中新建配置后端服务器的文件和数据库连接配置的文件，并新建 api 文件夹，存储接口文件。

创建好文件夹后，还要在前端的 config 文件中加入后端接口的地址，这样才能连好。

```
proxyTable: {
    '/api': {
      target: 'http://localhost:3000/api/',
      changeOrigin: true,
      pathRewrite: {
        '^/api': ''
```

```
        }
      }
    }
```

7.1.3 数据库搭建

后端连数据库的方式很简单，在安装好 mysql 依赖之后，要配置好数据库用户名，密码及数据库库名，再使用具体语句进行连接就可以。

这是我的配置文件：

```
// 数据库连接配置

module.exports = {
    mysql: {
      host: 'localhost',
      user: 'root',//数据库用户名
      password: '123456',//数据库密码
      database: 'sspdb',//所用数据库 school search platform database
      port: '3306'
    }
  };
```

7.2 系统逻辑设计

7.2.1 软件系统功能

在此软件系统中，主要的功能需求为：

1.针对普通用户：

（1）用户注册：用户输入用户名和密码完成注册

（2）用户登录：用户输入用户名和密码完成登录

（3）根据学校名查询学校：用户在主页输入学校名称查询学校，可以输入部分名称来查询。

（4）根据筛选条件查询学校：用户在筛选学校页面输入学校的任意信息筛选学校。

（5）查看学校基本信息：用户在学校主页查看学校的基本信息。

（6）查看学校通知：用户在学校主页查看学校的通知，可点击某通知进入查看详情。

（7）查看学校留言板：用户在学校主页查看学校留言板。

（8）在学校留言板上进行留言：用户在学校主页对学校留言板进行实名留言。

2.针对管理员用户：

（1）管理员注册：管理员输入注册码、学校编号、用户名、密码完成注册。

（2）管理员登录：管理员输入用户名、密码、学校编号完成登录。

（3）查看学校基本信息：管理员可在学校主页查看学校基本信息。

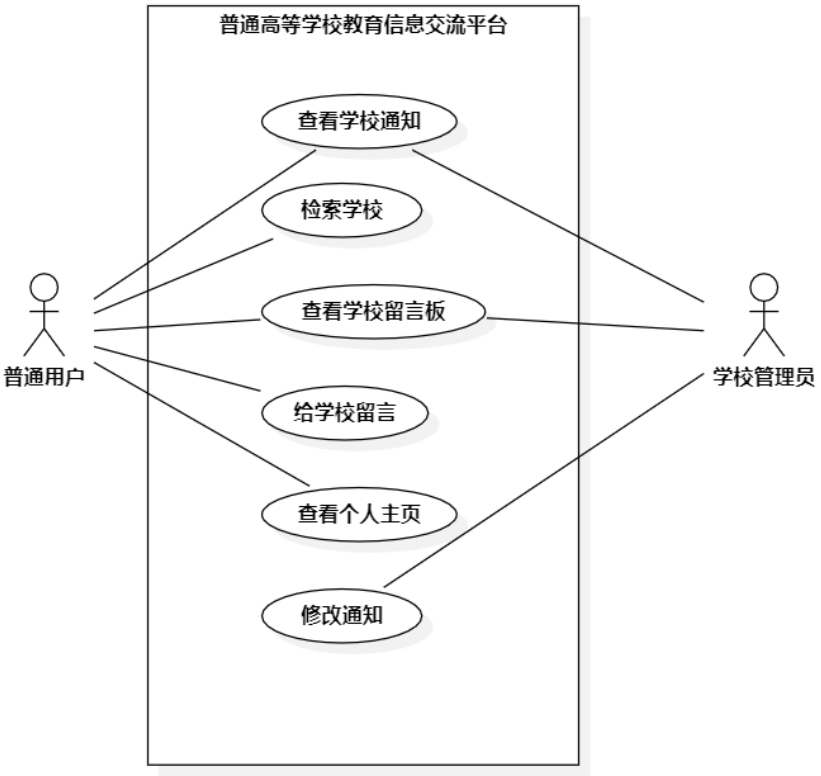
（4）查看学校通知：管理员可在学校主页查看学校通知。

（5）删除学校通知：管理员可在学校主页删除学校通知。

（6）添加学校通知：管理员可在学校主页点击添加学校通知按钮，然后跳转到另一个页面添加学校通知。

（7）查看学校留言板：管理员可查看学校留言板，但不能评论。

功能需求的用例图如下:



7.2.2 系统页面功能

本系统中含有多个页面，在这里，将对本系统几个比较重要的页面的功能进行详细的描述：

1. 登录注册页面(包含管理员和普通用户的登陆注册)：

在此页面中，可以对管理员和普通用户进行登录或者注册。然后在完成注册或登录后，自动跳转到下一个页面。

2. 主页页面

此页面是仅用户可见，普通用户登录或者注册成功后自动跳转到此页面。此页面的功能主要为：查看筛选学校页面，查看个人主页，搜索学校、进入学校主页。

点击前两项会跳转到其他页面，点击搜索学校会在主页显示搜索的结果，搜索到学校后，可以在相应学校下面点击进入学校主页从而跳转到学校主页页面。

3. 个人主页

通过主页面跳转而来，可以查看用户的个人信息。

4. 筛选学校

通过主页跳转到筛选学校界面，可以输入筛选条件进行筛选，并可通过筛选到的学校点击该学校进入学校主页页面。

5. 学校主页

学校主页页面分为两种模式：

第一种模式是普通用户模式，在此模式下，用户可以查看学校信息，查看学校通知栏，并可点击对应的通知进入相应的通知详情页面。在此模式下，用户还可以查看留言板，并在留言板下留言。

第二种模式是管理员模式，在此模式下，管理员可以发布通知，点击发布通知按钮进入发布通知页面。还可以删除通知，只需在对应通知旁边点击删除按钮即可。但是管理员仅可以查看留言板，而没有留言的权限。

6. 通知详情页

这个页面为通知详情页，可以显示通知的具体内容。

7.3 具体功能编写

7.3.1 数据库编写

1.建表语句：

（1）user 表

属性	字段名	类型	属性描述
ID	Uid	数值	用户的 ID 号，作为平台账号的标识
姓名	Uname	字符串	用户的姓名信息
密码	Psw	字符串	用户的平台密码

```
CREATE TABLE `user` (
  `Uid` int NOT NULL AUTO_INCREMENT,
  `Uname` varchar(20) DEFAULT NULL,
  `Psw` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`Uid`)
)
```

（2）school 表

属性	字段名	类型	属性描述
SID	Sid	int	学校的 ID 号
学校名称	Xxmc	text	学校的名称
学校标识码	Xxbsm	Bigint	学校的标识码
主管部门	Zgbm	Text	学校的主管部门
所在地	Szd	Text	学校的所在地
办学层次	Bxcc	Text	学校是本科还是专科
备注	Bz	Text	是民办则写在备注里
注册码	Zcm	Varchar	单独告知学校的注册码，可用于注册管理员

```
CREATE TABLE `school` (
  `Sid` int NOT NULL,
  `Xxmc` text,
  `Xxbsm` bigint DEFAULT NULL,
  `Zgbm` text,
  `Szd` text,
```

```
  `Bxcc` text,
  `Bz` text,
  `Zcm` varchar(10) DEFAULT 'zhucema123',
  PRIMARY KEY (`Sid`)
)
```

（3）noti 表

属性	字段名	类型	属性描述
ID	Nid	数值	通知的 ID 号
学校 ID	Sid	数值	发布通知的学校 ID，外键
通知大标题	Nheader	字符串	通知的大标题
通知内容	Ncontent	字符串	通知的内容
通知时间	Ntime	时间	通知的发布时间

```
CREATE TABLE `noti` (
  `Nid` int NOT NULL AUTO_INCREMENT,
  `Nheader` varchar(50) DEFAULT NULL,
  `Ncontent` text,
  `Ntime` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `Sid` int NOT NULL,
  PRIMARY KEY (`Nid`),
  KEY `Sid` (`Sid`),
  CONSTRAINT `noti_ibfk_1` FOREIGN KEY (`Sid`) REFERENCES `school` (`Sid`)
)
```

（4）comment 表

属性	字段名	类型	属性描述
ID	Cid	数值	评论的 ID 号
发布者 ID	Uid	数值	发布评论的用户 ID，外键
评论学校 ID	Sid	数值	被评论的学校 ID,外键
评论内容	Ccontent	字符串	评论的内容
发布时间	Ctime	时间	评论的发布时间
发布者姓名	Uname	字符串	发布评论的用户名

```
CREATE TABLE `comment` (
  `Cid` int NOT NULL AUTO_INCREMENT,
  `Ccontent` varchar(1000) DEFAULT NULL,
  `Ctime` timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  `Uid` int NOT NULL,
  `Uname` varchar(20) DEFAULT NULL,
  `Sid` int DEFAULT NULL,
  `LikeCount` int DEFAULT NULL,
  PRIMARY KEY (`Cid`),
  KEY `Sid` (`Sid`),
  KEY `Uid` (`Uid`),
  CONSTRAINT `comment_ibfk_1` FOREIGN KEY (`Sid`) REFERENCES `school` (`Sid`),
```

CONSTRAINT `comment_ibfk_2` FOREIGN KEY (`Uid`) REFERENCES `user` (`Uid`)
)

（5）schoolmanager 表

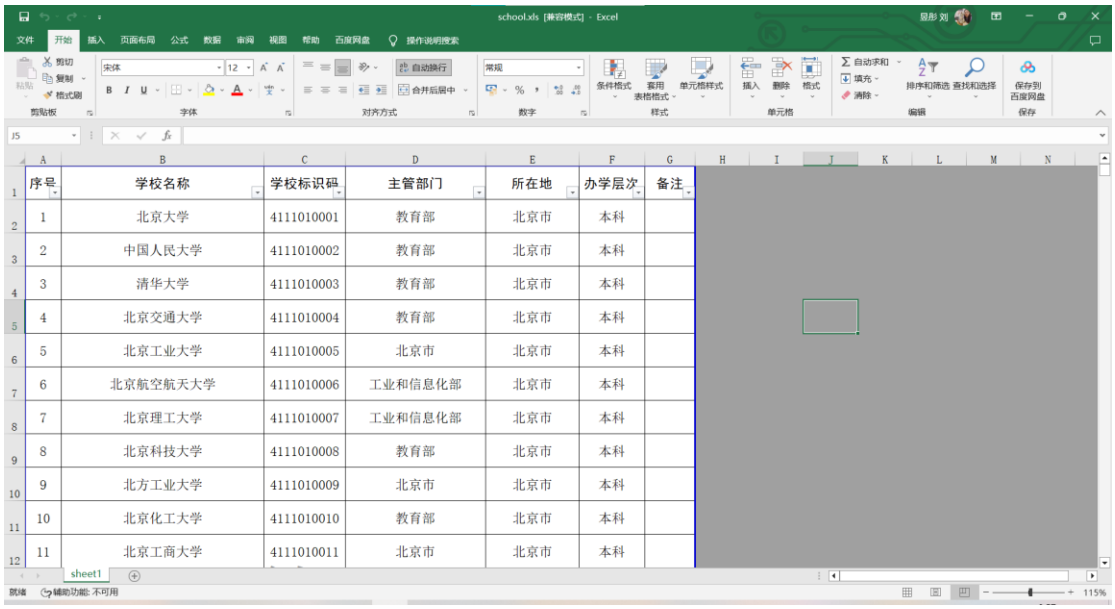
属性	字段名	类型	属性描述
ID	Mid	数值	管理员的 ID 号，作为平台账号的标识
姓名	Mname	字符串	管理员的姓名信息
密码	Mpassword	字符串	管理员的平台密码
学校 ID	Sid	数值	管理员所在学校的 id

```
CREATE TABLE `schoolmanager` (
  `Mid` int NOT NULL AUTO_INCREMENT,
  `Sid` int NOT NULL,
  `Mname` varchar(30) DEFAULT NULL,
  `Mpassword` varchar(30) DEFAULT NULL,
  PRIMARY KEY (`Mid`),
  KEY `Sid` (`Sid`),
  CONSTRAINT `schoolmanager_ibfk_1` FOREIGN KEY (`Sid`) REFERENCES `school` (`Sid`)
)
```

2.数据库信息导入

本数据库课设的内容为普通高等学校查询网站，因此数据的真实性非常重要。我选用的数据为中国人民共和国教育部政府门户网站中给出的全国高等学校名单中的附件一：全国普通高等学校名单。下载网址为：[全国高等学校名单 - 中华人民共和国教育部政府门户网站(moe.gov.cn)]

表格内容分为序号、学校名称、学校标识码、主管部门、所在地、办学层次、备注几栏。在对数据进行了一些处理后的 excel 表格如下：



序号	学校名称	学校标识码	主管部门	所在地	办学层次	备注
1	北京大学	4111010001	教育部	北京市	本科	
2	中国人民大学	4111010002	教育部	北京市	本科	
3	清华大学	4111010003	教育部	北京市	本科	
4	北京交通大学	4111010004	教育部	北京市	本科	
5	北京工业大学	4111010005	北京市	北京市	本科	
6	北京航空航天大学	4111010006	工业和信息化部	北京市	本科	
7	北京理工大学	4111010007	工业和信息化部	北京市	本科	
8	北京科技大学	4111010008	教育部	北京市	本科	
9	北方工业大学	4111010009	北京市	北京市	本科	
10	北京化工大学	4111010010	教育部	北京市	本科	
11	北京工商大学	4111010011	北京市	北京市	本科	

下一步需要做的就是把这个 excel 表导入到数据库中，具体的操作非常简单，只需要先将 excel 表格转为 csv 格式。我采用的是 MySQL 的 Workbench，图形化界面操作起来非常简单，按要求输入想导入的格式就可以，下图就是导入的过程以及导入成功的截图。

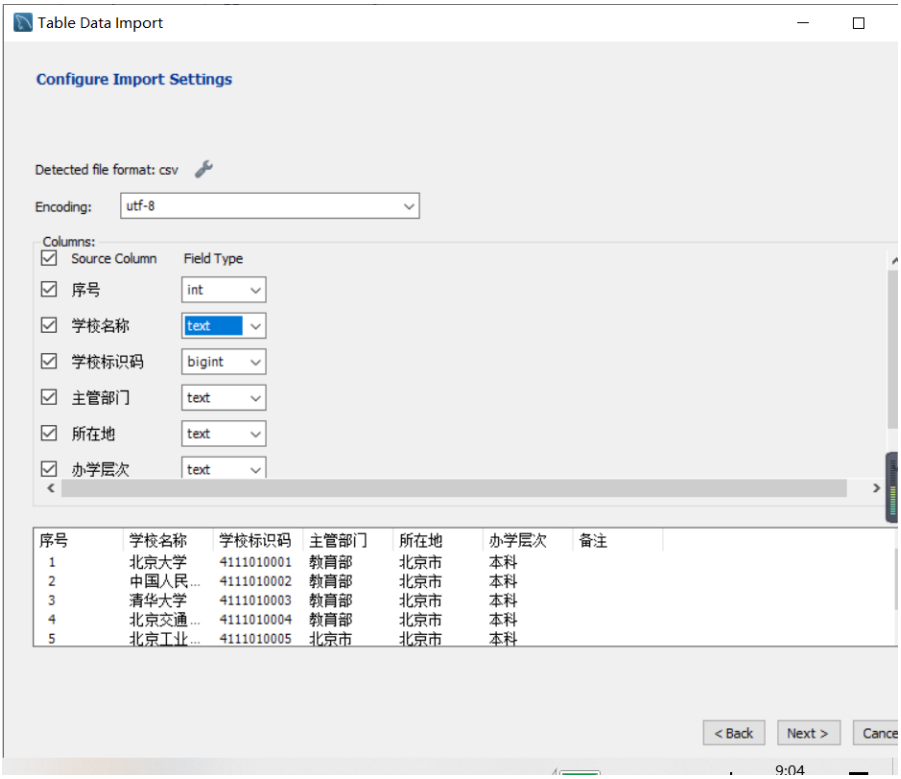


Table Data Import
Configure Import Settings
Detected file format: csv
Encoding: utf-8
Columns:
Source Column
Field Type
序号 int
学校名称 text
学校标识码 bigint
主管部门 text
所在地 text
办学层次 text
序号 学校名称 学校标识码 主管部门 所在地 办学层次 备注
1 北京大学 4111010001 教育部 北京市 本科
2 中国人民... 4111010002 教育部 北京市 本科
3 清华大学 4111010003 教育部 北京市 本科
4 北京交通... 4111010004 教育部 北京市 本科
5 北京工业... 4111010005 北京市 北京市 本科
< Back
Next >
Cance

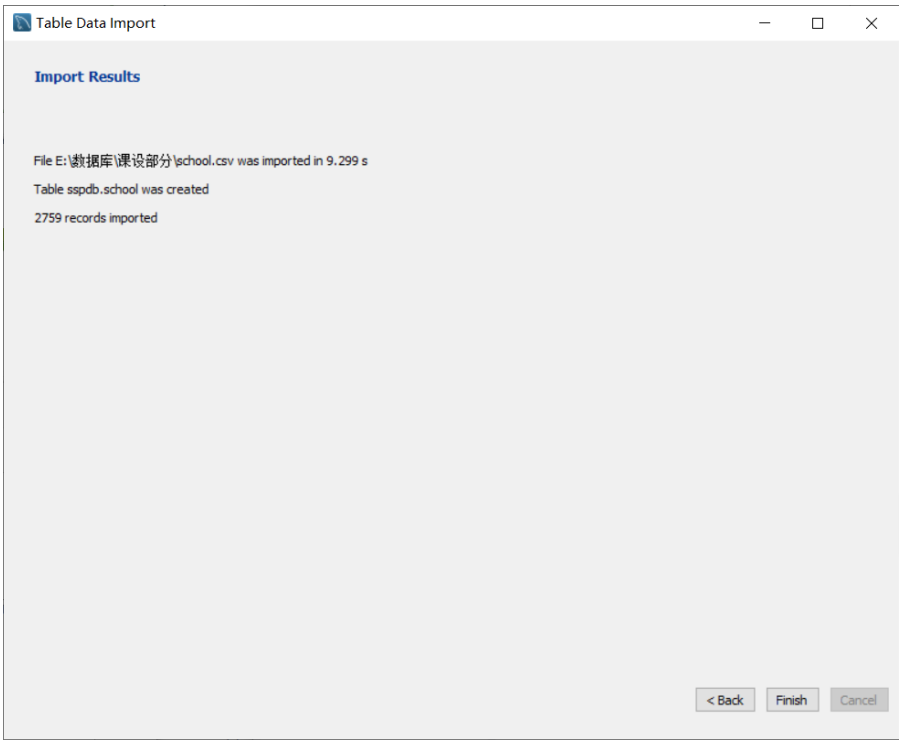
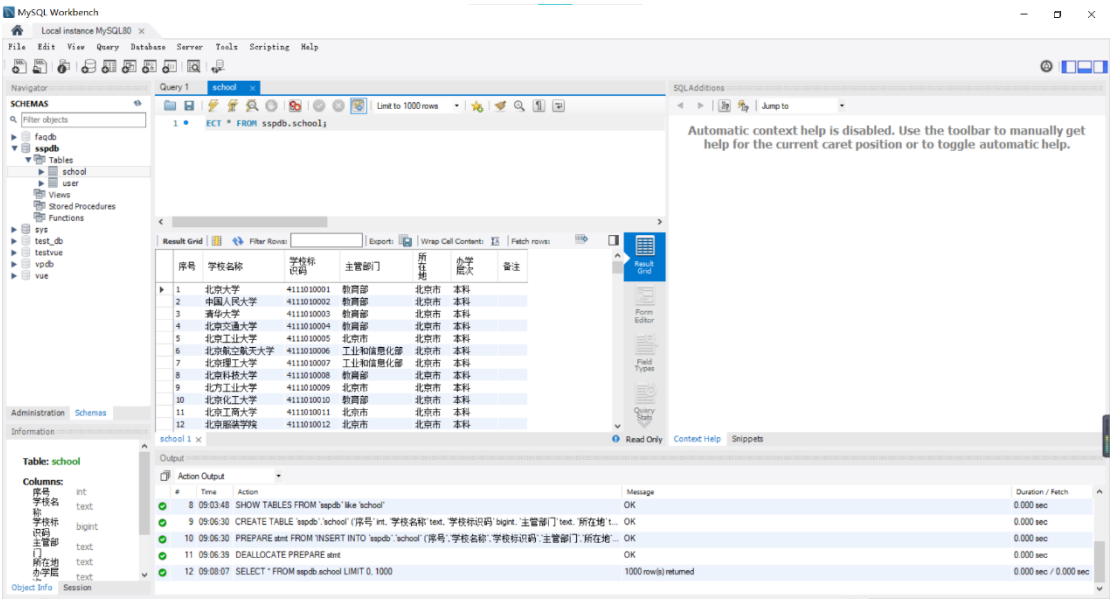


Table Data Import
Import Results
File E:\数据库\课设部分\school.csv was imported in 9.299 s
Table sspdb.school was created
2759 records imported
< Back
Finish
Cancel



以上就是数据库数据导入的操作，其他数据的导入均可以在网站上交互输入！

7.3.2 后端接口编写

1. 增加用户接口

功能：向 user 表中添加一个用户，输入的内容为 Uname 和 Psw

SQL 语句：`insert into user（Uname,Psw） values (?,?)`

实现代码：

```
// 增加用户接口
router.post('/addUser', (req, res) => {
  var params = req.body;
  var sql = 'insert into user(Uname,Psw) values (?, ?)';
  console.log(params);
  conn.query(sql, [params.username,params.password], function(err,
result) {
    if (err) {
      console.log(err);
    }
    if (result) {
      jsonWrite(res, result);
    }
  })
});
```

2. 增加管理员接口

功能：向 SchoolManager 表中添加一个管理员用户，输入内容为 Mname,Mpassword,Sid

SQL 语句：`insert into SchoolManager(Mname,Mpassword,Sid) values (?,?, ?)`

实现代码：

```
router.post('/addManager', (req, res) => {
```

```
  var params = req.body;
  var sql = 'insert into SchoolManager(Mname,Mpassword,Sid) values
(?,?, ?)';
  console.log(params);
  conn.query(sql, [params.Mname,params.Mpassword,params.Sid],
function(err, result) {
    if (err) {
      console.log(err);
    }
    if (result) {
      jsonWrite(res, result);
    }
  })
});
```

3.增加通知接口

功能：向 Noti 表添加一条通知，输入内容为 Nheader，Ncontent，Sid

SQL 语句：`insert into noti (Nheader,Ncontent,Sid) values (?,?, ?)`

实现代码：

```
router.post('/addNotice', (req, res) => {
  var params = req.body;
  var sql = 'insert into noti (Nheader,Ncontent,Sid) values (?,?, ?)';
  console.log(params);
  conn.query(sql, [params.Nheader,params.Ncontent,params.Sid],
function(err, result) {
    if (err) {
      console.log(err);
    }
    if (result) {
      jsonWrite(res, result);
    }
  })
});
```

4.查询用户接口

功能：在 user 表中按照 Uname 和 Psw 的值进行查询。

SQL 语句：`select * from user where Uname = ? and Psw = ?`

实现代码：

```
router.post('/queryUser', (req, res) => {
    var params = req.body;
    var sql = " select * from user where Uname = ? and Psw = ?";
    console.log(params);
    conn.query(sql, [params.username,params.password], function(err,
result) {
```

```
      if (err) {
        console.log(err);
      }
      if (result) {
        console.log(result.length);
        if(result.length === 0 ){
          console.log("不存在该用户");
          jsonWrite(res, result);
        }
        else {
          console.log("存在该用户");
          jsonWrite(res, result);
        }
      }
    })
  });
```

5.查询管理员用户

功能：在 schoolmanager 表中按照 Sid、Mname 和 Mpassword 的值进行查询。

SQL 语句：`select * from SchoolManager where Mname = ? and Mpassword = ? and Sid=?`

实现代码：

```
router.post('/queryManager', (req, res) => {
    var params = req.body;
    var sql = " select * from SchoolManager where Mname = ? and
Mpassword = ? and Sid=?";
    console.log(params);
    conn.query(sql, [params.Mname,params.Mpassword,params.Sid],
function(err, result) {
      if (err) {
        console.log(err);
      }
      if (result) {
        console.log(result.length);
        if(result.length === 0 ){
          console.log("不存在该用户");
          jsonWrite(res, result);
        }
        else {
          console.log("存在该用户");
          jsonWrite(res, result);
        }
      }
    })
```

```
});
```

6.删除通知接口
功能:在 noti 表中删除一条 Nid 为特定值的通知
SQL 语句：`delete from noti where Nid = ?`
实现代码：

```
router.post('/deleteNotice', (req, res) => {
    var params = req.body;
    var sql = " delete from noti where Nid = ?";
    console.log(params);
    conn.query(sql, [params.Nid], function(err, result) {
      if (err) {
        console.log(err);
      }
      if (result) {
        console.log(res);
        jsonWrite(res, result);
      }
    })
  });
```

7.查询学校接口 1
功能：在 school 表中按照学校名称进行模糊查询
SQL 语句：`select * from school where xxmc like ?`
实现代码：

```
router.post('/searchSchool',(req,res)=>{
  var params=req.body;
  var name="%"+params.search+"%";
  var sql='select * from school where xxmc like ?';
  console.log(params);
  conn.query(sql,name,function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

8.查询学校接口 2
功能：在 school 表中按照 Sid 进行
SQL 语句：`select * from school where Sid = ?`
实现代码：

```
router.post('/searchSchool2',(req,res)=>{
  var params=req.body;
  var sql='select * from school where Sid = ?';
  console.log(params);
  conn.query(sql,params.Sid,function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

9.获取留言板接口

功能：在 Comment 表中按照 Sid 进行查询

SQL 语句：`select * from comment where Sid = ?`

实现代码：

```
router.post('/getComments',(req,res)=>{
  var params=req.body;
  var sql='select * from comment where Sid = ?';
  console.log(params);
  conn.query(sql,params.Sid,function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

10.添加留言接口

功能：添加一条留言到 comment 表中，添加的信息有 Sid,Uid,Uname，Ccontent，LikeCount

SQL 语句：`insert into comment(Sid,Uid,Uname,Ccontent,LikeCount) values (?, ?,?,?,?)`

实现代码：

```
router.post('/addComment',(req,res)=>{
  var params=req.body;
  var sql = 'insert into comment(Sid,Uid,Uname,Ccontent,LikeCount)
values (?, ?,?,?,?)';
  console.log(params);
  conn.query(sql,[params.Sid,params.Uid,params.Uname,params.Ccontent,par
ams.LikeCount],function(err,result){
```

```
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

11.筛选学校接口

功能：通过一系列筛选条件在 School 表中进行筛选，每一项都是模糊查找，如果这一项为空则进行一番操作忽略这个筛选条件。

SQL 语句：`select * from school where '+name1+' like ? and '+name2+' like ? and '+name3+' like ? and '+name4+' like ? and '+name5+' like ?`

实现代码：

```
router.post('/filterSchool',(req,res)=>{
  var params=req.body;
  var Xxmc='';
  var Xxbsm='';
  var Zgbm='';
  var Szd='';
  var Bxcc='';
  var name1='\'\'';
  var name2='\'\'';
  var name3='\'\'';
  var name4='\'\'';
  var name5='\'\'';
  if(params.Xxmc!=""){
      Xxmc="%"+params.Xxmc+"%";
      name1='Xxmc';
  }
 if(params.Xxbsm!=""){
      Xxbsm="%"+params.Xxbsm+"%";
      name2='Xxbsm';
  }
  if(params.Zgbm!=""){
      Zgbm="%"+params.Zgbm+"%";
      name3='Zgbm';
  }
  if(params.Szd!=""){
      Szd="%"+params.Szd+"%";
      name4='Szd';
  }
  if(params.Bxcc!=""){
```

```
      Bxcc="%"+params.Bxcc+"%";
      name5='Bxcc';
  }

  var sql=
  'select * from school where '+name1+' like ? and '+name2+' like ? and
'+name3+' like ? and '+name4+' like ? and '+name5+' like ?';
  console.log(params);
  conn.query(sql,[Xxmc,Xxbsm,Zgbm,Szd,Bxcc],function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

12.获取通知列表接口
功能：在 noti 表中获取 Sid 为特定值的项
SQL 语句：`select * from noti where Sid = ?`
实现代码：

```
router.post('/getNotices',(req,res)=>{
  var params=req.body;
  var sql='select * from noti where Sid = ?';
  console.log(params);
  conn.query(sql,params.Sid,function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

13.获取一条通知接口
功能：通过特定的 Nid 获取对应的通知信息，并且因为 Nid 为主键，所以应当只有一条
SQL 语句：`select * from noti where Nid = ?`
实现代码：

```
router.post('/getoneNotice',(req,res)=>{
  var params=req.body;
  var sql='select * from noti where Nid = ?';
  console.log(params);
```

```
  conn.query(sql,params.Nid,function(err,result){
    if(err){
      console.log(err);
    }
    if(result){
      jsonWrite(res,result);
    }
  })
})
```

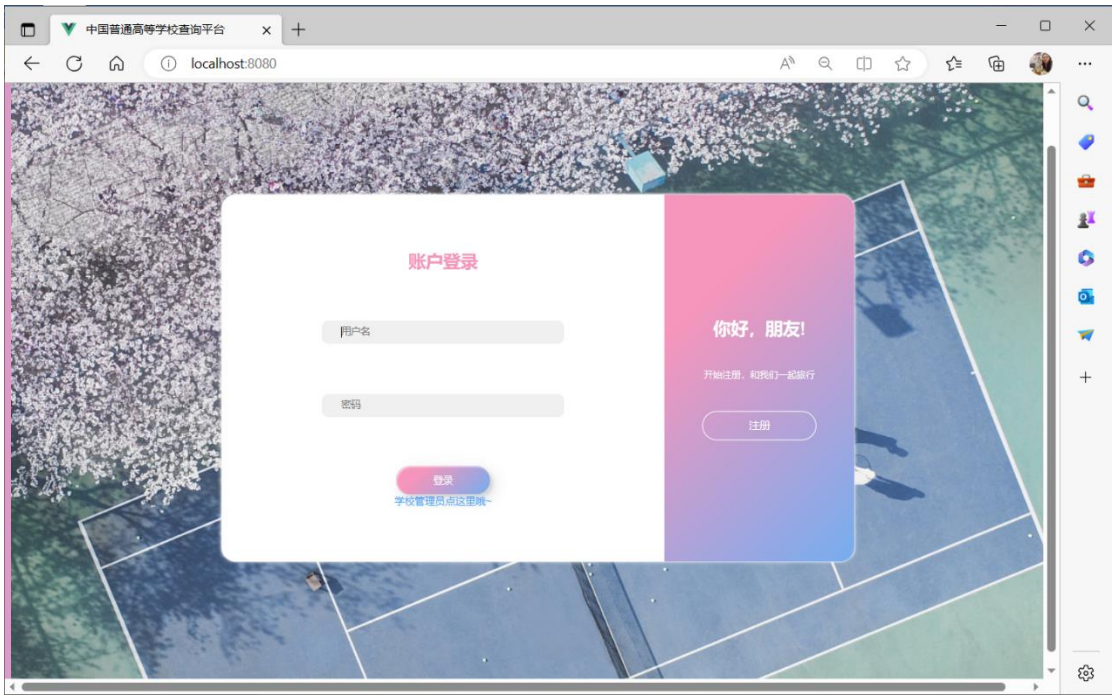
7.3.2 具体页面编写

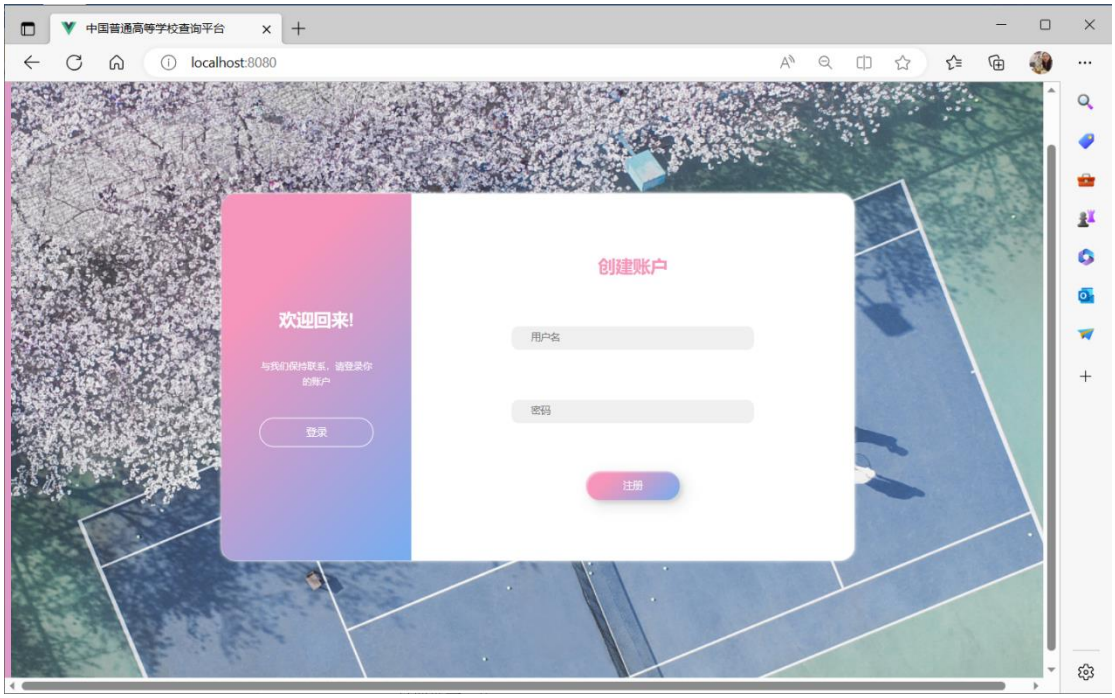
1.登录注册页面 Login-register

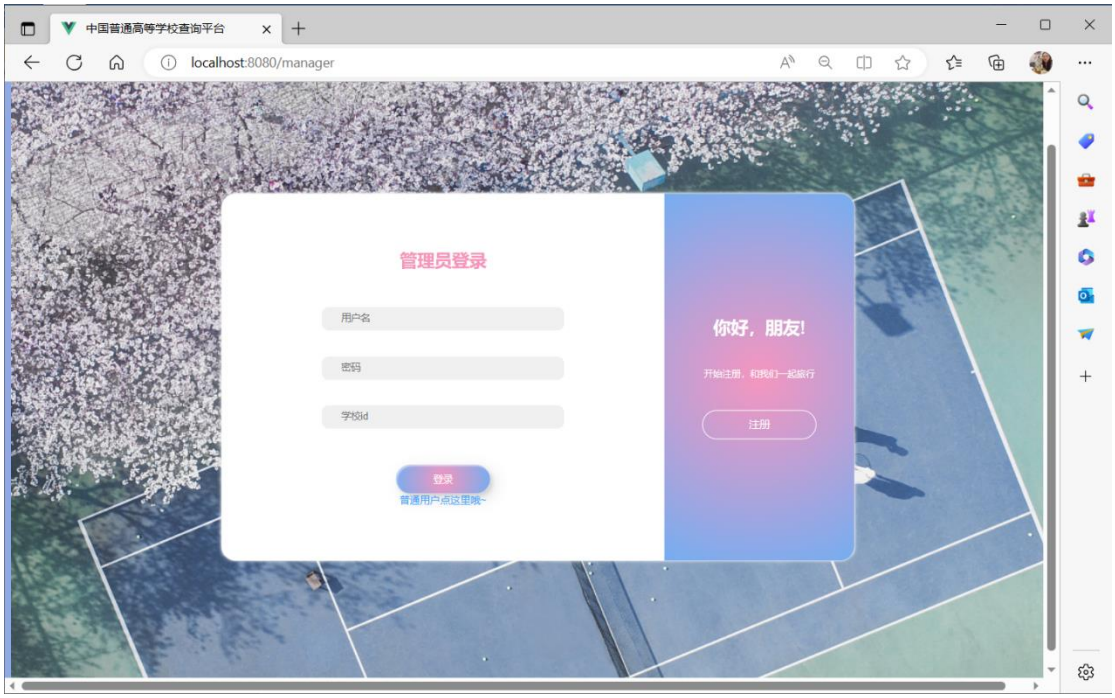
描述：普通用户和管理员均可以在此页面中进行注册或登录

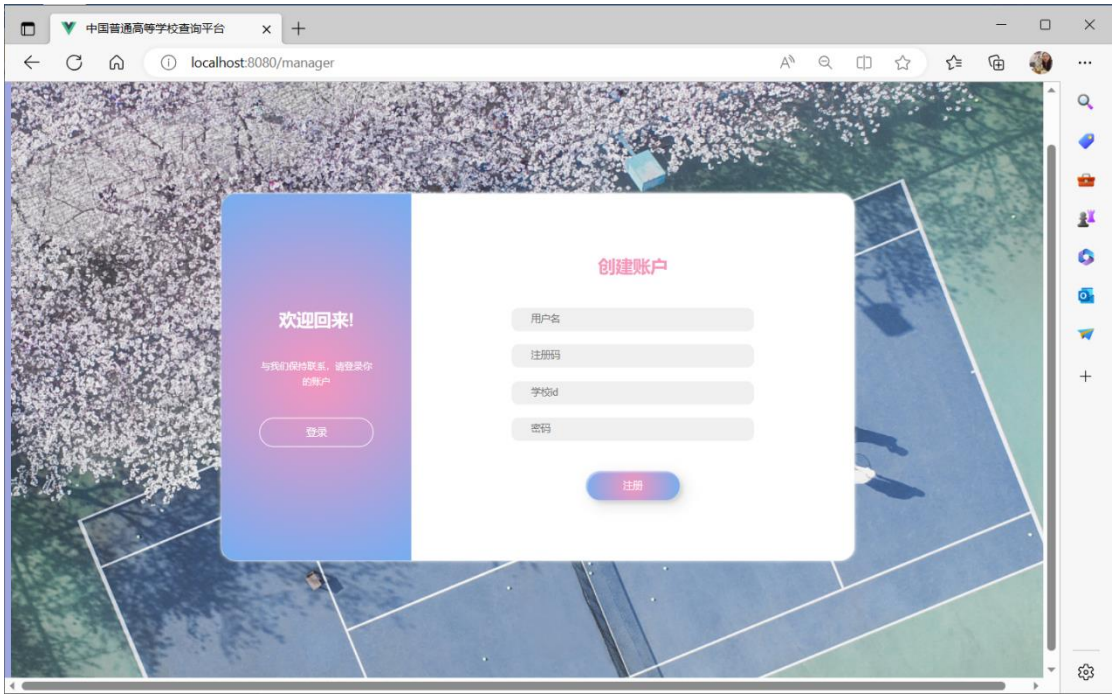
功能：在“登录”状态下，输入用户名，密码，（Sid），点击登录即可登录；在“注册”状态下，输入用户名，密码，（Sid，注册码），点击注册即可完成注册。

页面展示：




中国普通高等学校查询平台
localhost:8080
欢迎回来!
与我们保持联系，请登录你的账户
登录
创建账户
用户名
密码
注册


中国普通高等学校查询平台
localhost:8080/manager
管理员登录
用户名
密码
学校id
登录
普通用户点这里哦~
你好，朋友!
开始注册，和我们一起旅行
注册

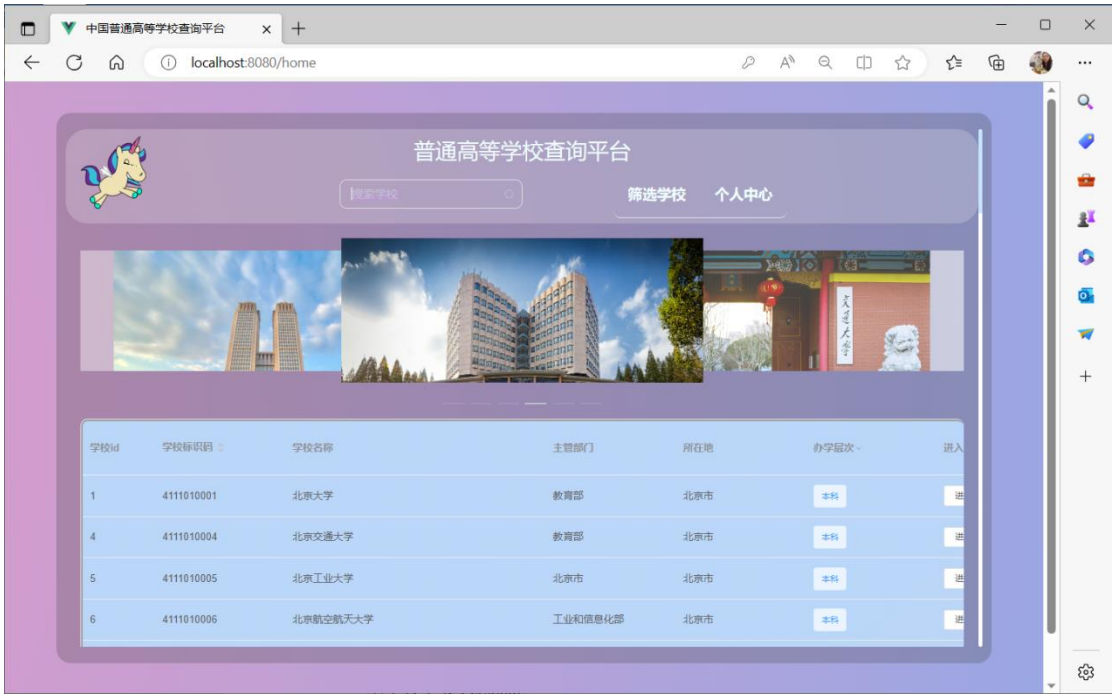


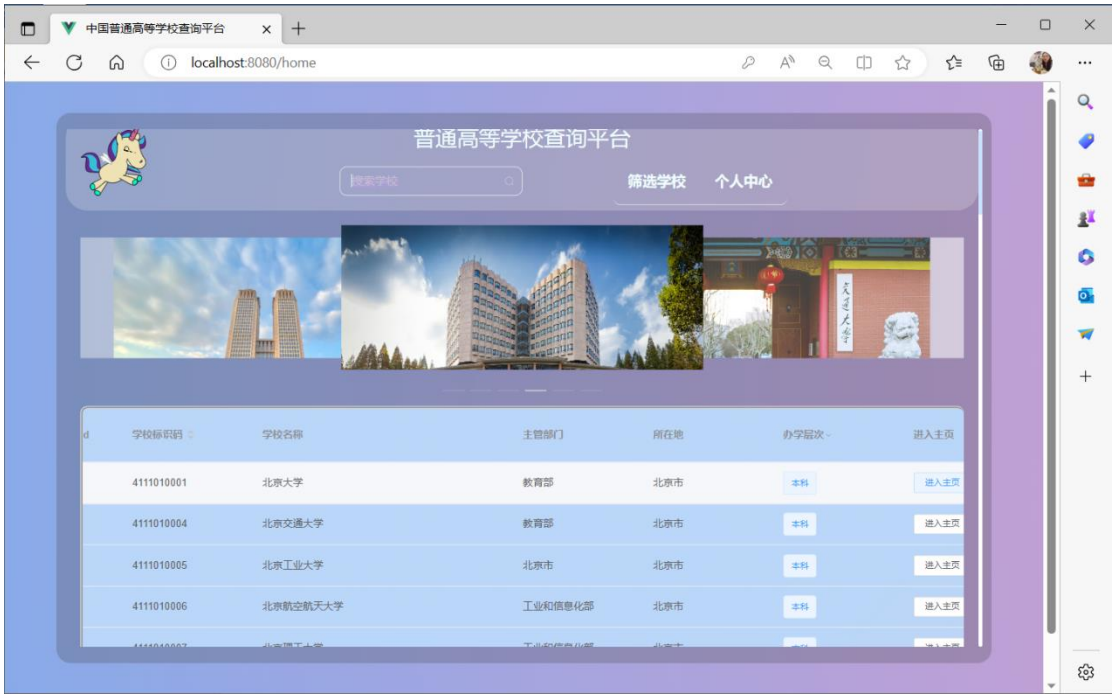
2.主页 homepage

描述：用户可以在此页面中模糊搜索学校。

功能：用户在上方的搜索栏输入模糊搜索的内容，回车进行搜索。下方的表格中会显示搜索的结果。在下方的表格中，点击进入主页按钮可进入相应学校的界面。

页面展示：



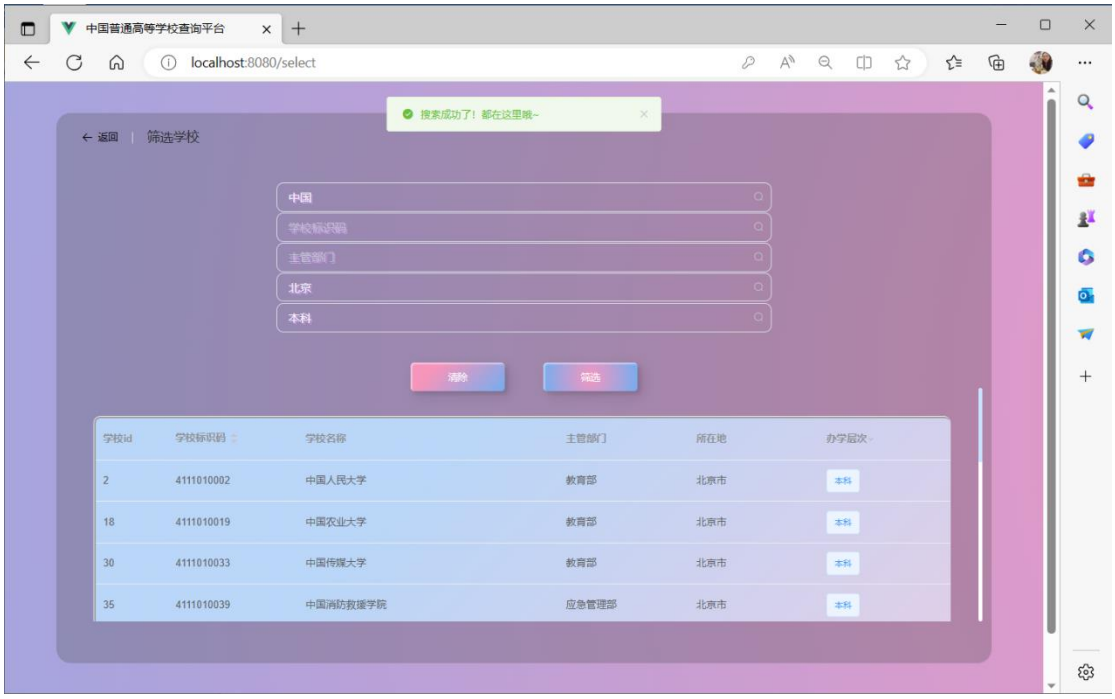


3.筛选页面 Select

描述：用户进入到筛选页面中，可以根据多项进行模糊查询，不是每一项都必须要输入内容。

功能：输入想要筛选的条件，点击筛选按钮进行搜索。下方的表格中会显示搜索的结果。在下方的表格中，点击进入主页按钮可进入相应学校的界面。

页面展示：



4.学校主页 School

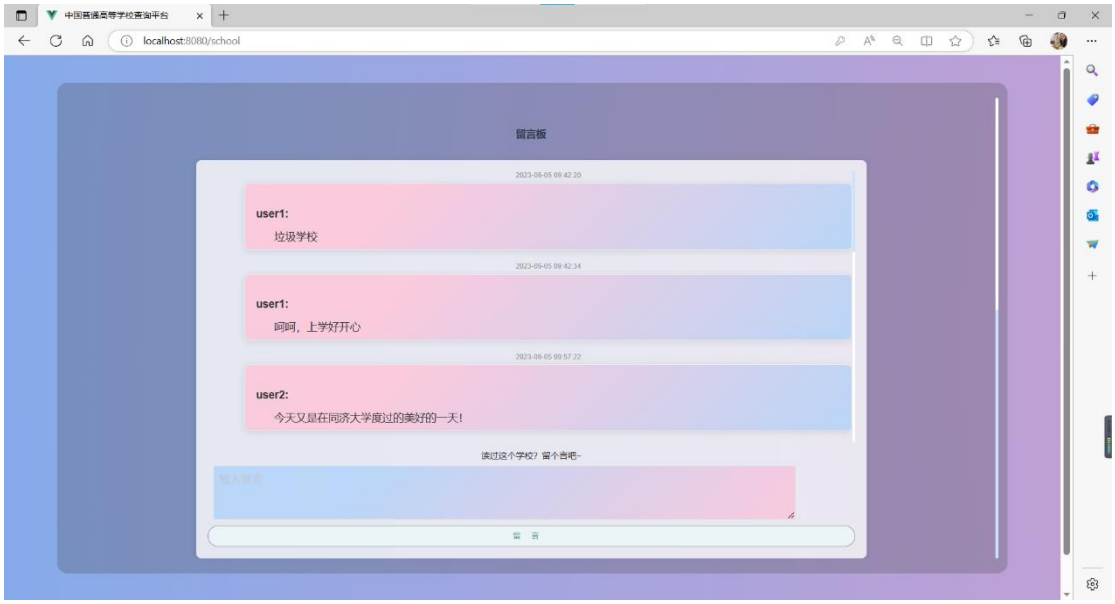
描述：这个页面是学校的主页，功能比较多。

功能：这个页面是管理员和用户都可以访问的页面，但是两种模式下的权限是不一样的。管理员在此页面中可以添加通知，删除通知，查看留言板，查看学校基本信息，但是不可以

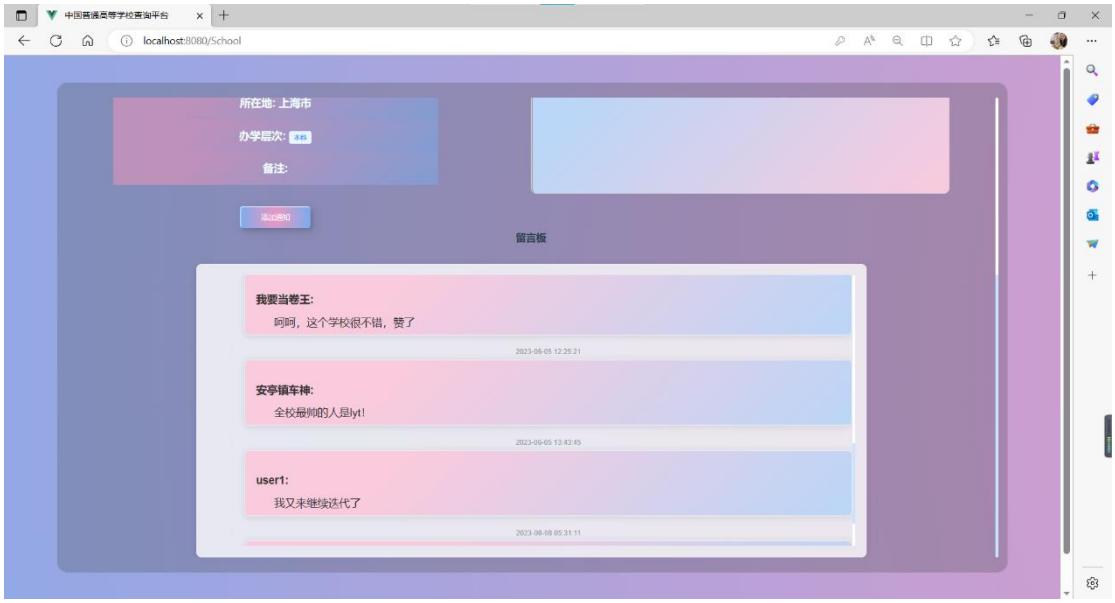
进行留言；普通用户在此页面中可以查看通知，查看学校基本信息、查看留言板并可以留言，但是不可以改变通知。

页面展示：

普通用户版：



管理员版：

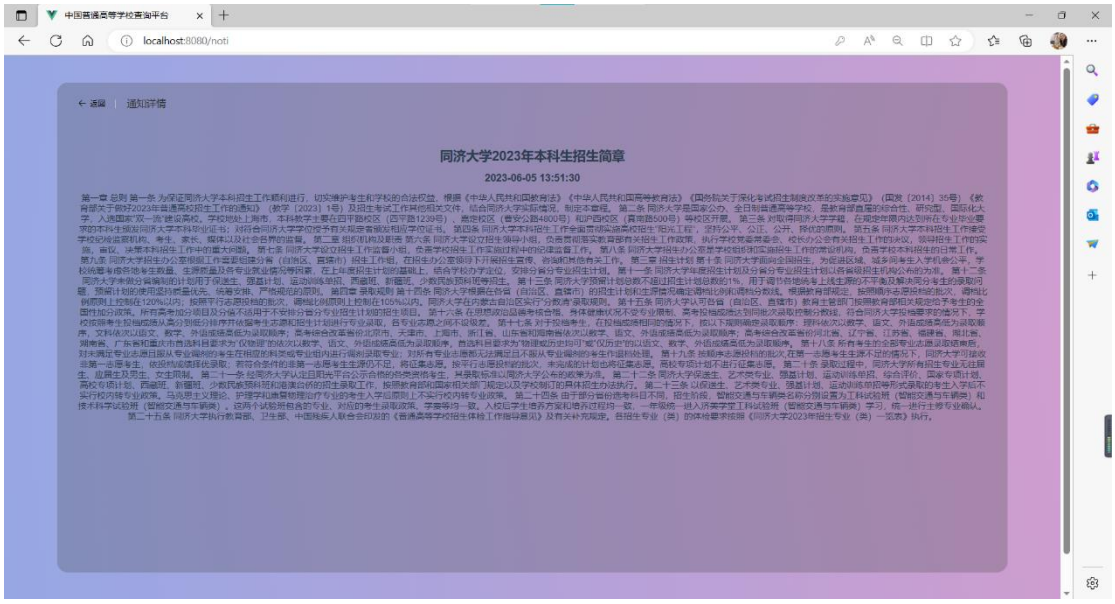


5.学校通知 Noti

描述：在学校主页可以点击通知来查看通知详情

功能：查看

页面展示：

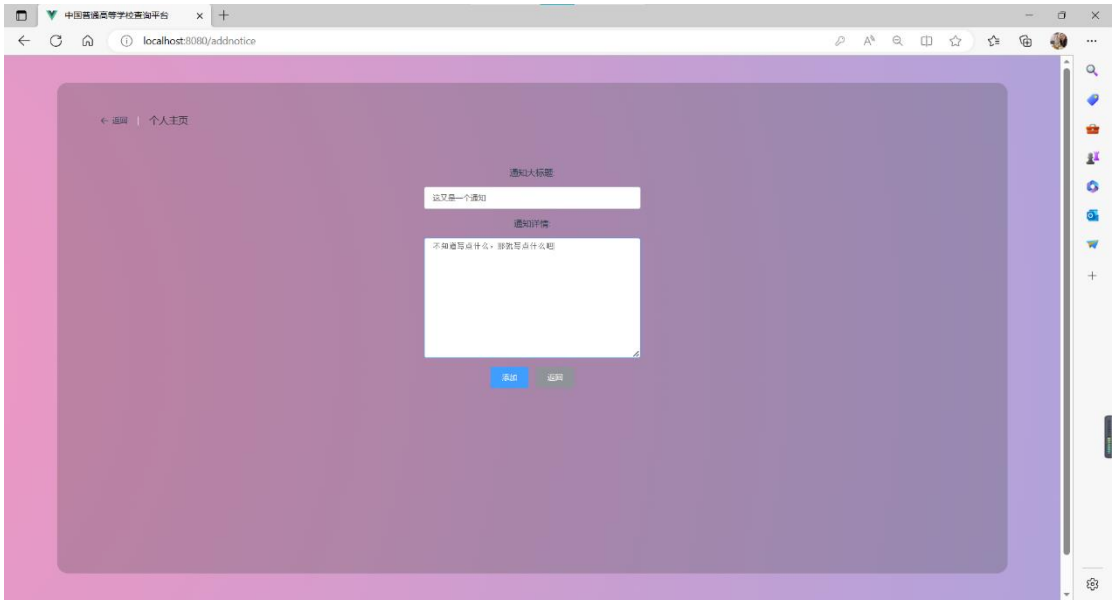


6.增加通知 AddNoti

描述：管理员在学校主页点击添加通知后跳转到的界面，可以输入添加的通知。

功能：管理员输入通知标题，通知内容完成通知的添加。

页面展示：

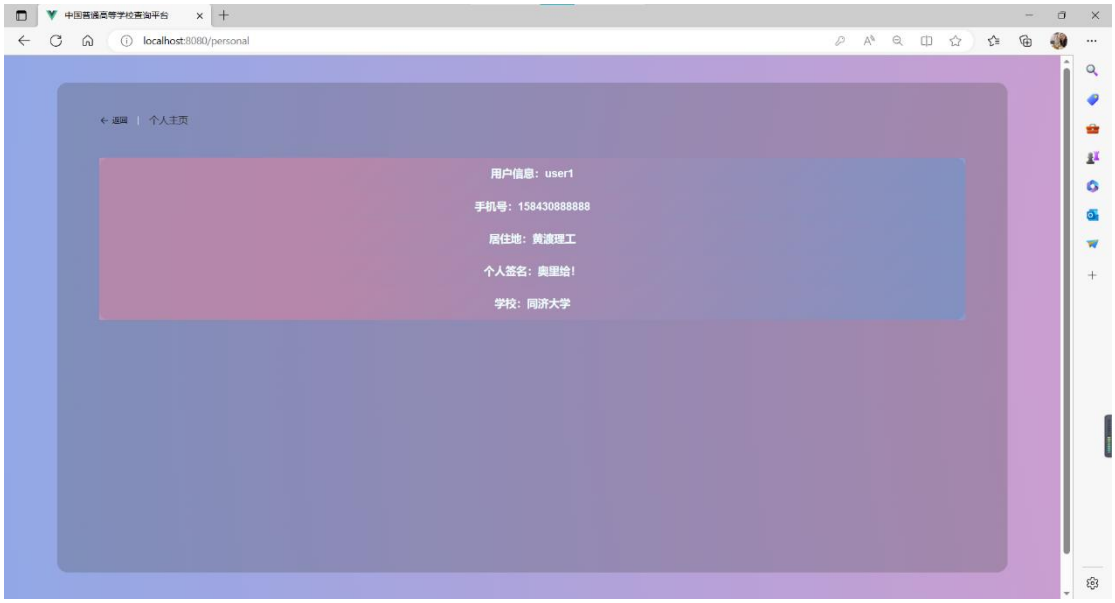


7.个人主页 Personal

描述：在主页点击个人主页可进入此页面，主要是个人信息的展示。

功能：查看个人信息

页面展示：



7.4 功能测试

7.4.1 数据库性能测试

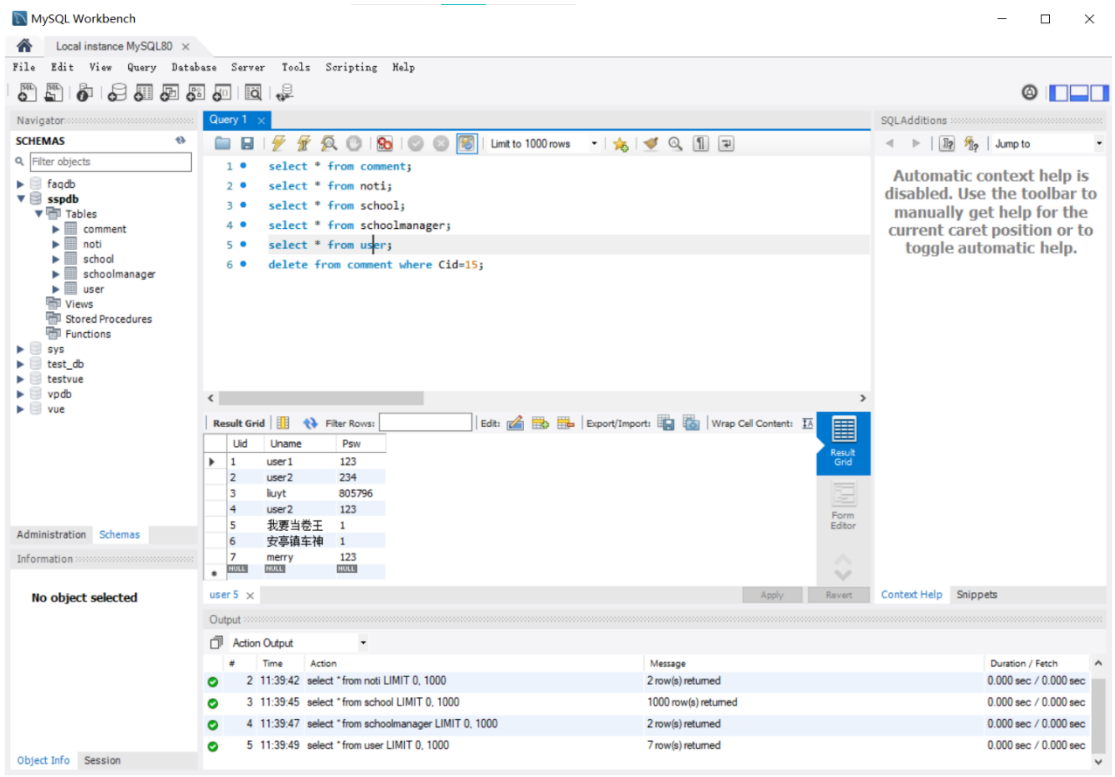
在数据库测试中，主要测试了以下内容

（1）数据库的账号管理

（2）数据库的表单建立

（3）数据的插入、删除、查询与修改

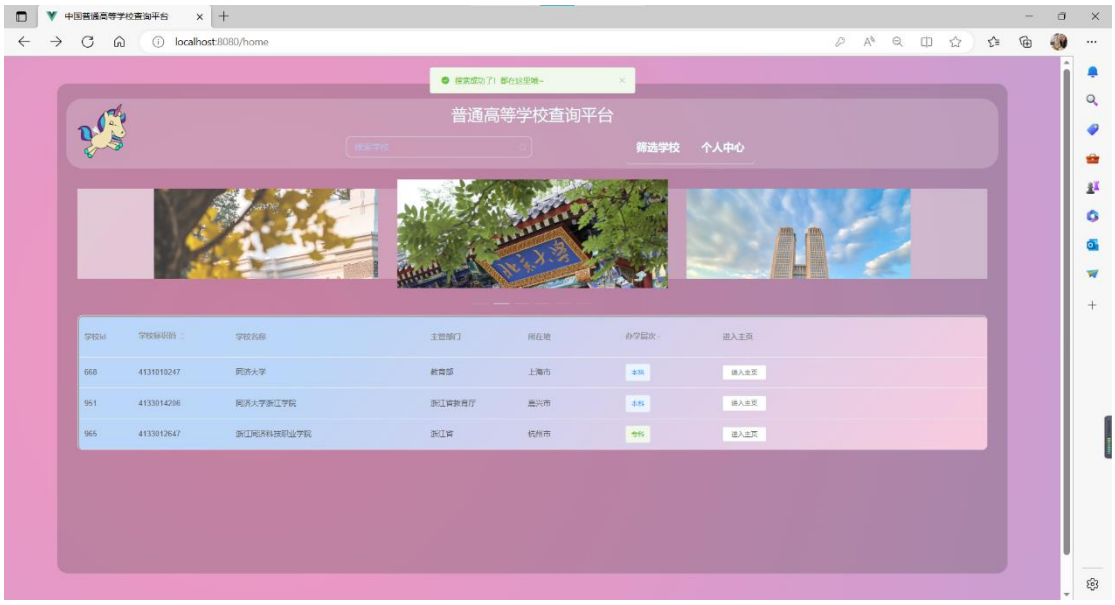
均成功完成，未发现问题。



7.4.2 系统综合测试

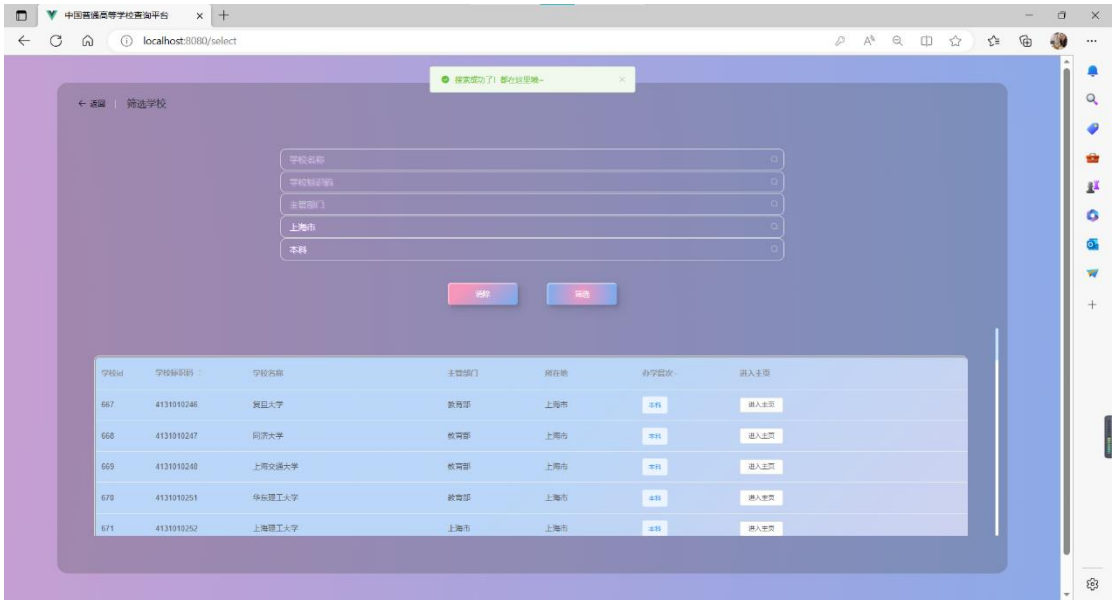
（1）普通用户查找学校

输入想搜的东西即可模糊查找，测试无误。



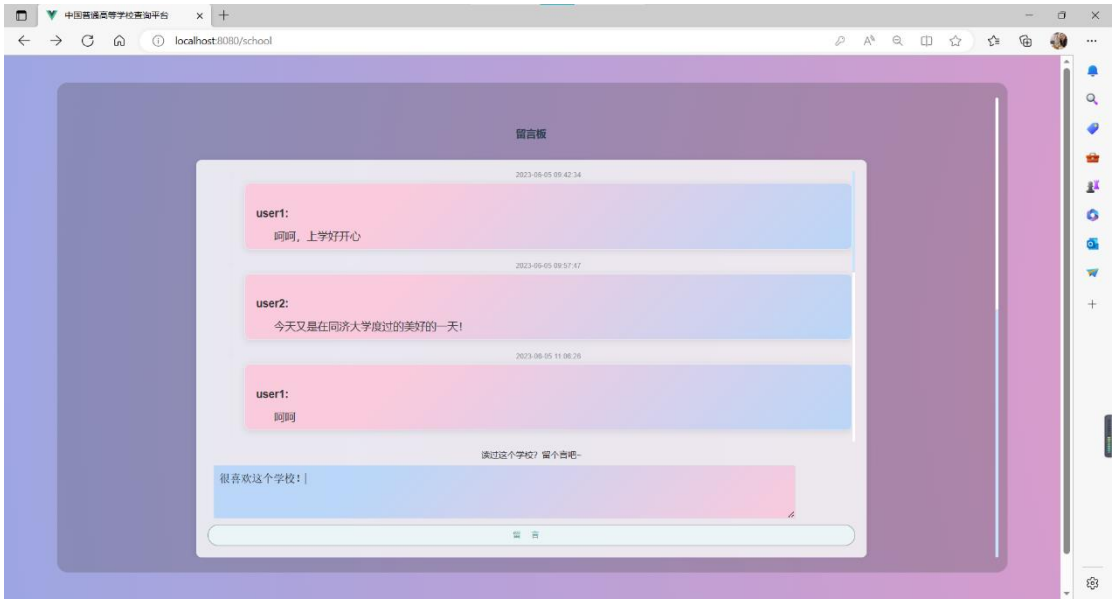
（2）普通用户筛选学校

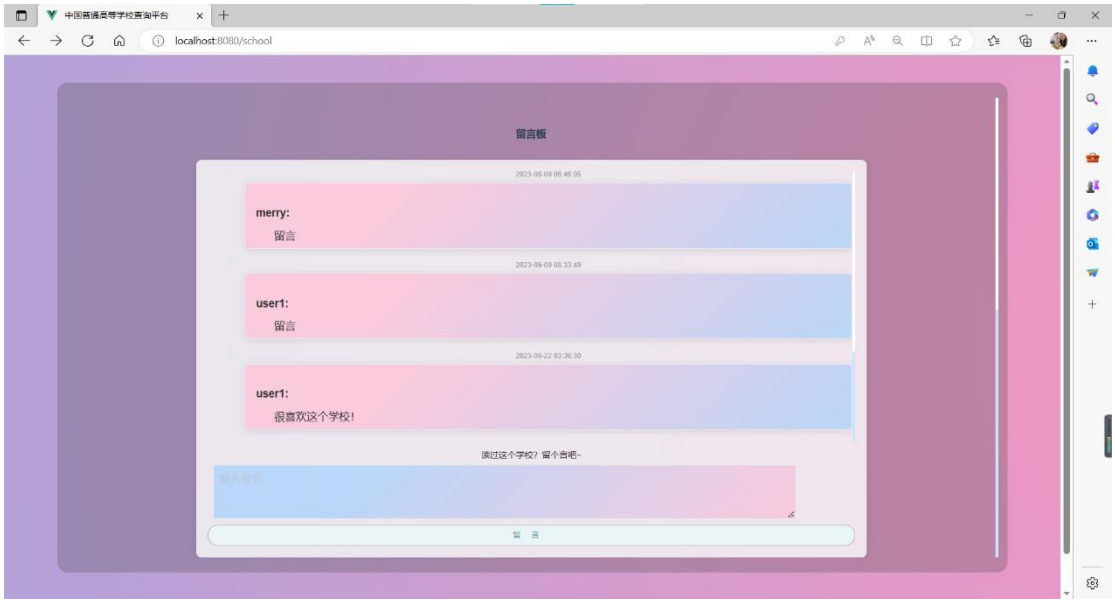
输入想筛选的条件即可筛选，测试无误。



（3）普通用户评论学校

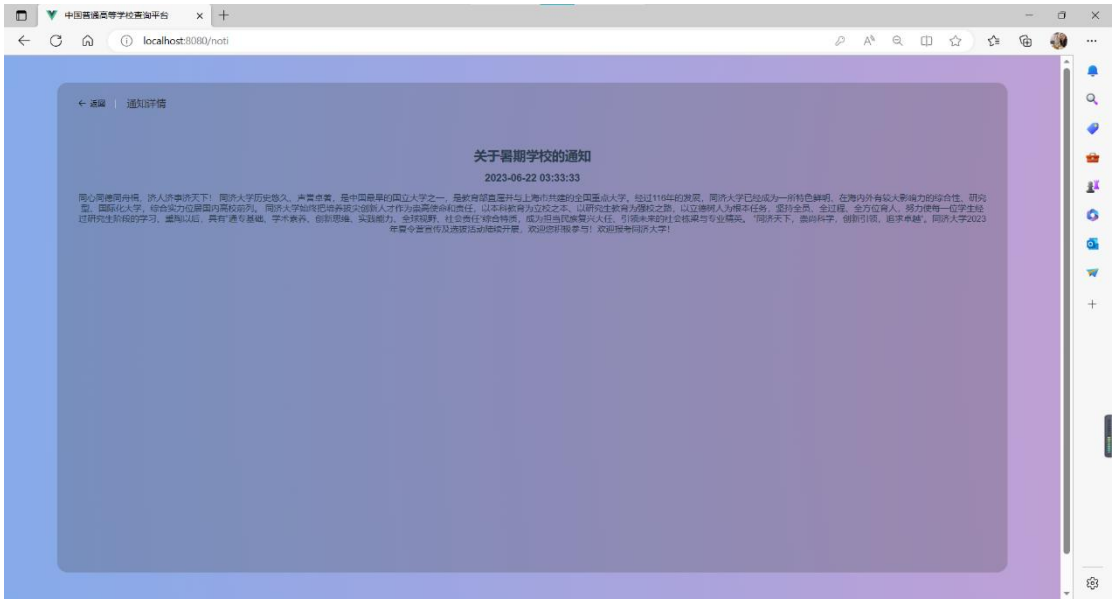
输入评论内容即可评论，测试无误。





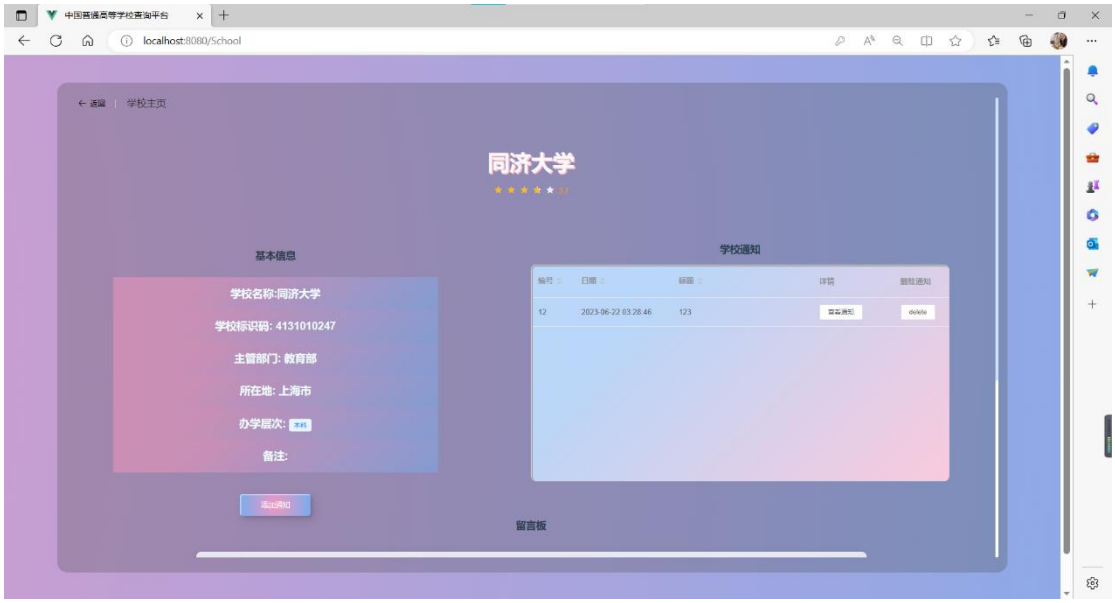
（4）普通用户查看通知

点击查看通知即可查看通知，测试无误。



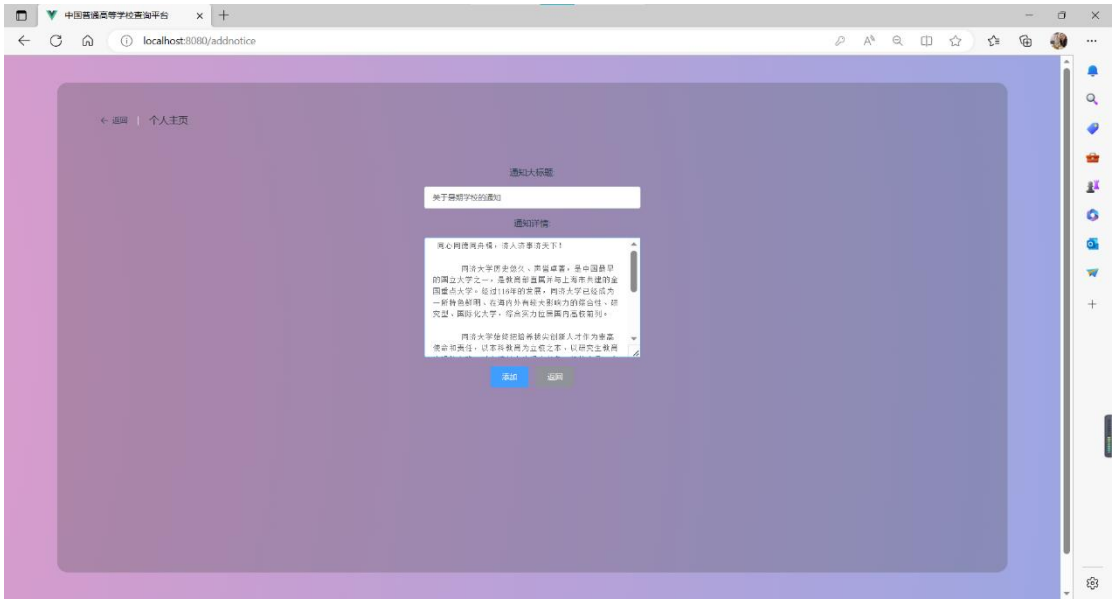
（5）管理员登录学校页面

输入账号密码及学校号即可正常登录，测试无误。



（6）管理员添加通知

输入通知标题和通知内容即可添加通知，测试无误。



八.系统运行维护

8.1 运行环境

对于硬件方面的需求，由于本应用的实现功能较为简单，承载的数据量小，而且使用的开发模块均较为轻量级，因此对硬件配置的需求较低，可以在大多数主机上正常运行。

对于软件方面的需求，需要配置较为复杂的环境，主要是安装 MYSQL 的服务器端，以及基于 JavaScript 的许多依赖包。后者使用了 package.json 进行描述，可以使用 npm 等包管理工具快速安装。如需部署到服务器上运行，则可以通过 docker 形式。

本项目的代码部分主要分为三项部分：前端程序、后端程序与数据库程序。开发过程中，三个模块在同一台主机上以不同端口通信，在实际应用时，前端模块应当在用户主机上运行，后端模块应当在远程服务器上运行。

8.2 相关维护接口

由于本项目清晰的实现了前后端分离，接口部分的代码清晰易懂，具有很强的可读性。数据库系统的登录使用了 root 用户身份进行管理，在应用层面除了普通账号之外还有管理员账号，便于实现内容与功能的管理。

8.3 可维护性分析

本项目采用了 Vue、Express 和 MySQL 完成，具有较高的可维护性，主要体现在以下几个方面：

1.模块化设计思想：Vue 框架本身就是一个支持组件化开发的框架，可以将页面拆分为多个组件，方便维护和复用。同时，Express 也采用了模块化设计思想，将应用程序拆分成多个模块，可以降低代码的复杂度，提高可维护性。

2.分层架构设计：采用前端-后端-服务器的分层架构设计，可以将不同的功能模块拆分到不同的层次，降低模块之间的耦合性，减少代码的依赖关系，从而提高代码的可维护性。规范化的编码风格和注释：在开发过程中，采用规范化的编码风格和注释，可以提高代码的可读性和可维护性。

3.数据库的规范化设计：MySQL 是一种关系型数据库，采用规范化设计可以降低数据冗余和数据不一致的风险，提高数据的一致性和完整性，从而提高代码的可维护性。

4.开源社区的支持：Vue、Express 和 MySQL 都是开源软件，有庞大的开源社区支持，可以及时修复漏洞和提供技术支持，保证了代码的可靠性和稳定性，提高了代码的可维护性。

综上所述，本项目具有较高的可维护性。

九.总结

在这次的实验中，我利用了 MySQL+Vue+express 实现了这样的一个普通高等学校信息交流平台。在实现这个项目的过程中，我有一些心得体会。

首先，良好的前期设计是一切的基础。在一开始就把功能设计好的话，会让整个系统更完整，功能设计更合理。也能避免多次的返工。因为修改起来其实很麻烦，尤其是对数据库的更改，如果不在一开始就设置好，就还要在表中更改字段等等。

其次，对于界面的前端设计要统一，可以采用全局 css 的方式，这样实际上在使得界面更美观一致的同时，也节省了时间。

最后就是我对这个项目的体会，在前期调研的过程中，我发现，网上的各种信息确实存在不太统一的问题，在搜索过程中需要对每个学校逐个搜索，很麻烦。所以这样一个集合的平台，可以为人们提供更方便的服务。但同时，一个重要问题是，我觉得相比于我这样自己为每个学校在我的平台创建一个发布通知的地方，不如在平台上给出该学校发布通知的官网网址。但鉴于数据库太过庞大，我不可能靠自己的力量添加每个学校的网址到数据库里，因此未能实现。但是我觉得这里是一个值得改进的地方。

目前我能实现的平台不过如此，但是等到未来有能力去实现更好的平台的时候，我觉得如果能为全国人民提供一个公开透明的教育信息平台是一件很有意义的事情。希望未来有机会可以实现这件事情。

参考文献

【1】Vue.js 教程：https://www.runoob.com/vue3/vue3-tutorial.html
【2】Vue.js 项目实战教程：https://www.vue3js.cn/vue3js-tutorial
【3】Express 教程：https://www.runoob.com/nodejs/nodejs-express-framework.html
【4】JavaScript 基础教程：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript
【5】HTML 和 CSS 教程：https://www.w3school.com.cn/
【6】中国高等学校名单：全国高等学校名单（截至 2022 年 5 月 31 日）_服务信息_中国政府网 (www.gov.cn)