湖南铁路科技职业技术学院

毕业设计成果

课　　题：基于.Net 和 HTML5 的《CAC 电脑装机小屋》项目设计与实现

专　　业：软件技术

班　　级：2004

学生姓名：胡健

所属学院：铁道工程与信息学院

指导教师：黄曾雄

湖南铁路科技职业技术学院教务处监

目 录

第一章项目描述及要求

“CAC 装机小屋”，是利用计算机技术对电脑硬件的获取，了解电脑硬件知识，讨论相关知识的一款网页应用程序。

众所周知，讨论研究电脑硬件的受众很小，关注电脑硬件发展、性能、性价比的人群比较少，使用“装机小屋”能为不了解电脑硬件相关知识的人群提供一个信息、硬件获取平台，能友好讨论有关电脑硬件的话题。

本系统的开发环境是在 Visual Studio 2019 下，采用 ASP.NET、BOOTSTRAP、JQUERY、HTML 等技术进行开发。其后台数据库主要采用 SQL SERVER 2018 并使用三层架构实现采用的这几种开发技术，极大的提升了工作速率。

在该系统中主要包含的模块有：登录、注册、查看、电脑硬件论坛、管理员管理的数据的增删改。它为用户提供了一个信息获取平台，大大减少了用户获取信息的时间，一体化的操作服务，让系统统计和查询变得格外便捷。

本系统目前的主要对象是想要获取电脑硬件信息的人群，在计算机网络，数据库和先进的开发平台上，利用现有的软件，配置一定的硬件，分析和设计一个具有开放体系结构的、易扩充的、易维护的的信息传播平台，为用户提供充分的信息、快捷的信息查询和友好的交流，减少不必要的时间浪费，提高电脑硬件信息获取效率。

第二章项目概要设计

2.1 应用程序架构用例图

2.1.1 登录功能用例图

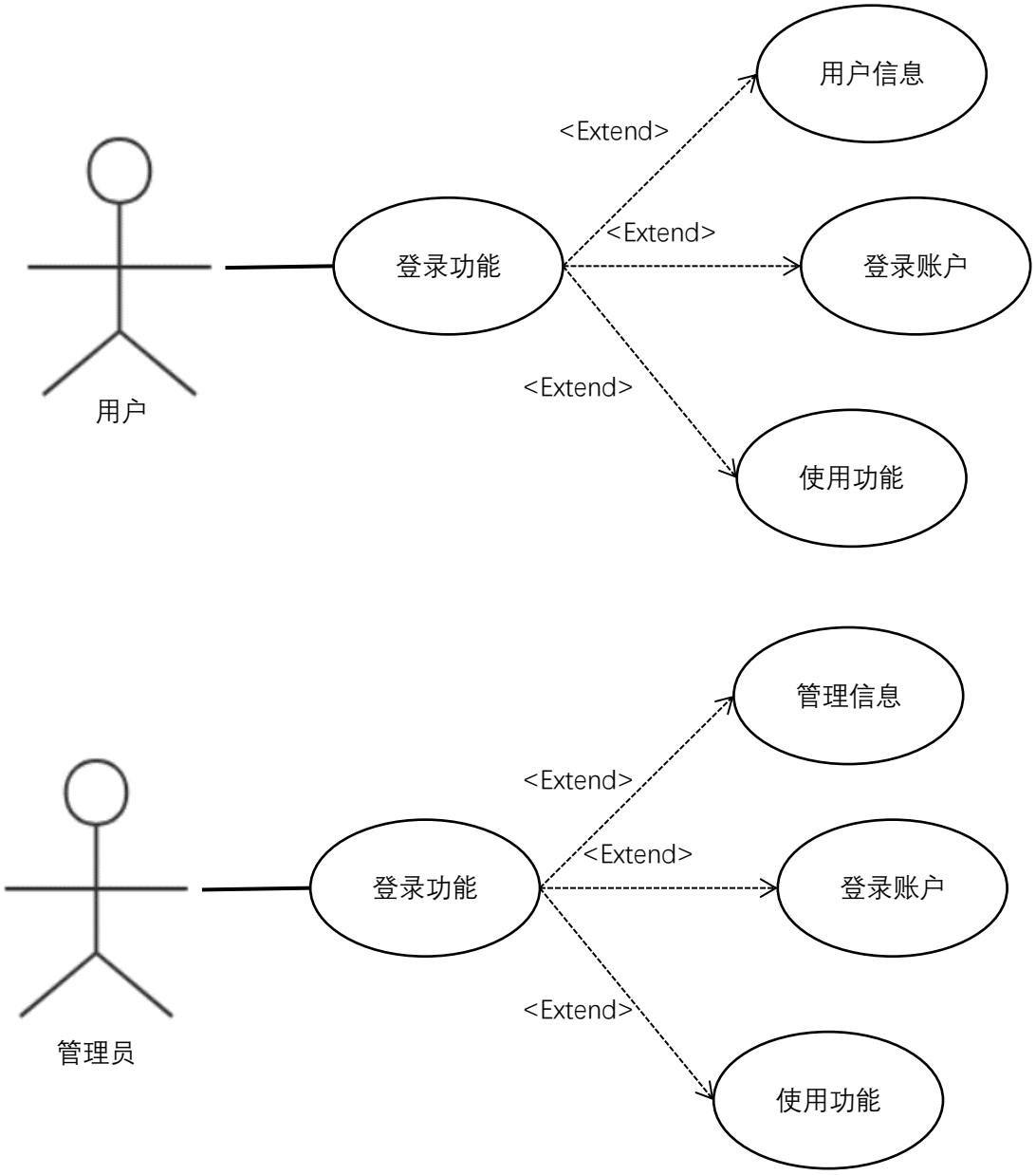

图 2.1 登录功能用例图

2.1.2 注册功能用例图

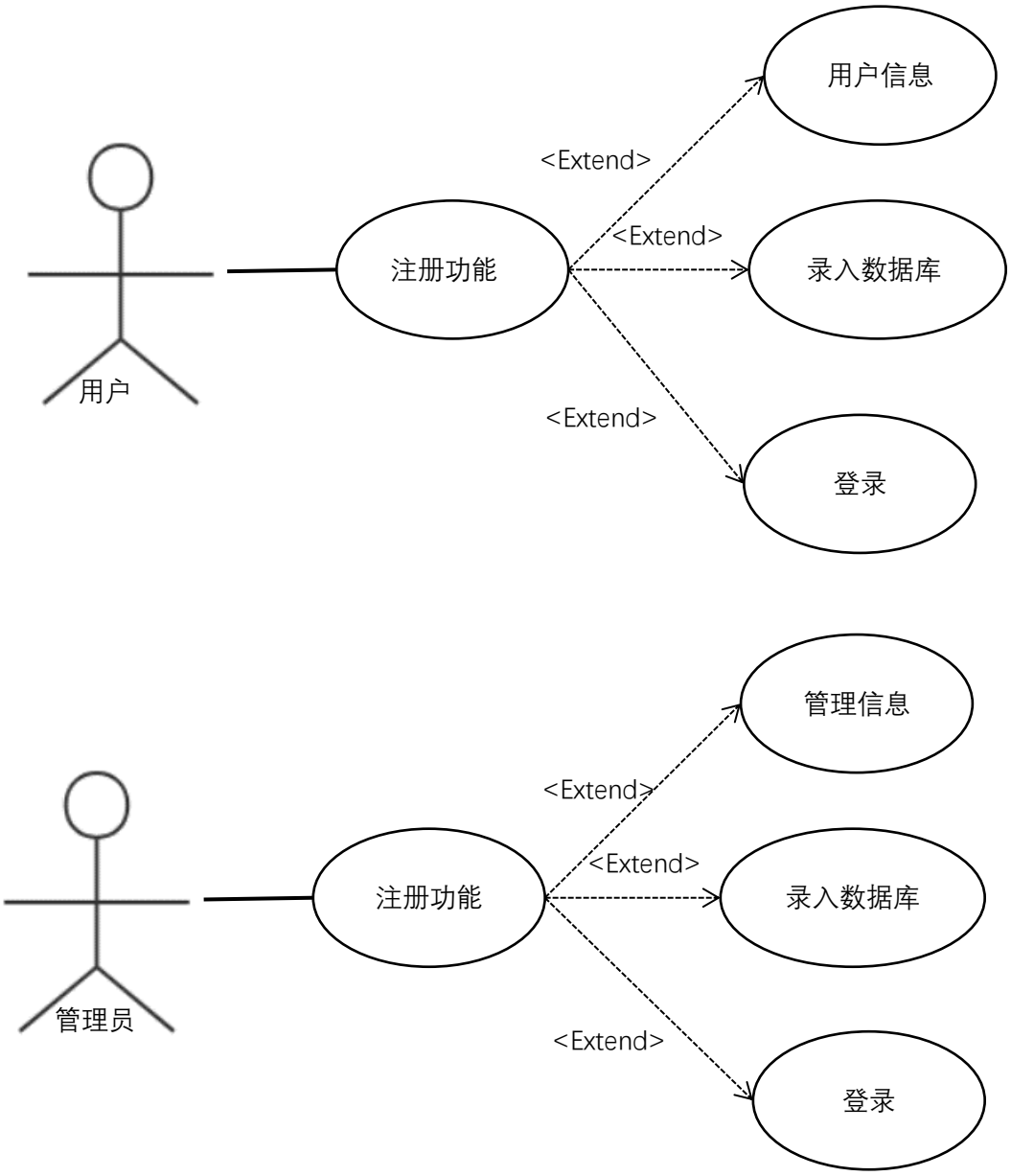

图 2.2 注册功能用例图

2.1.3 首页功能用例图

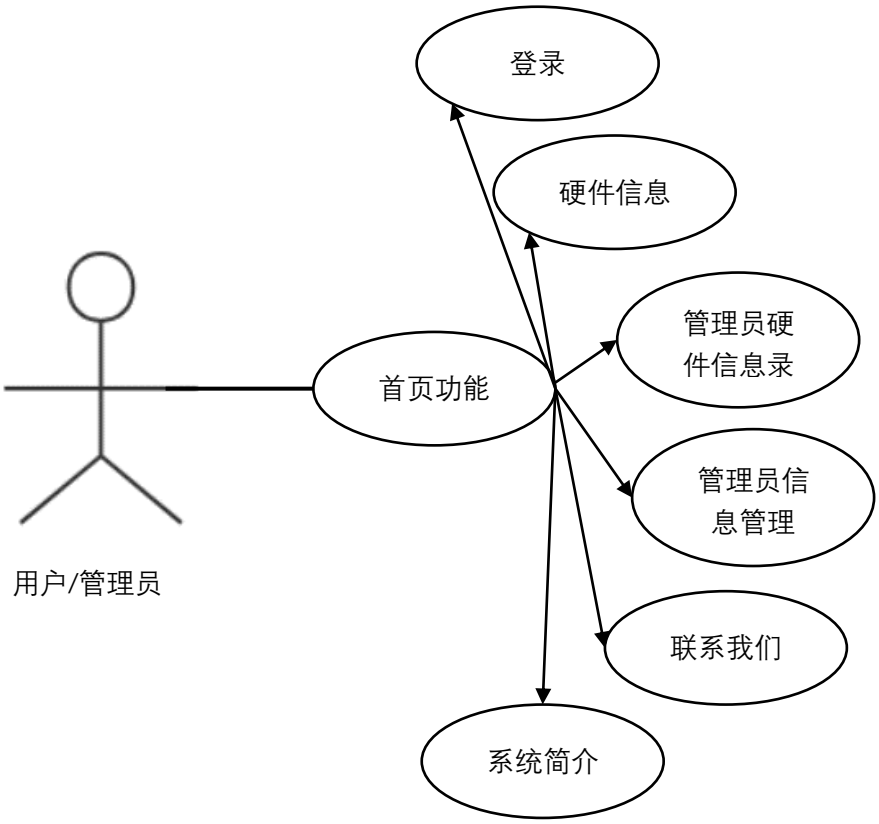

图 2.3 首页功能用例图

2.1.4 论坛功能用例图

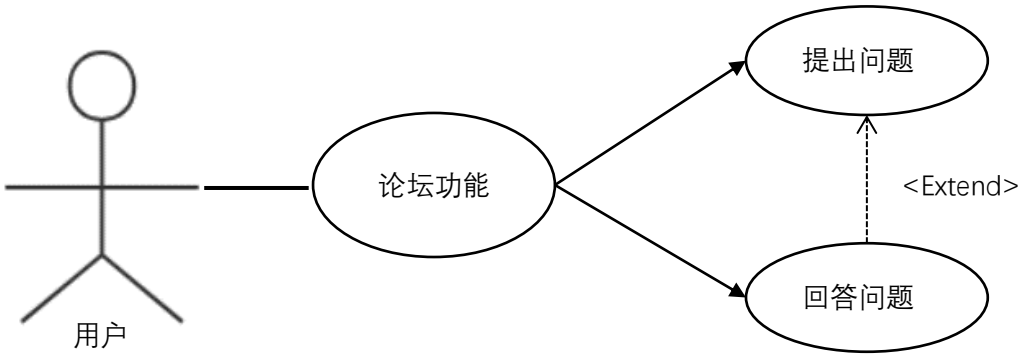

图 2.4 论坛功能用例图

2.1.5 信息管理功能用例图

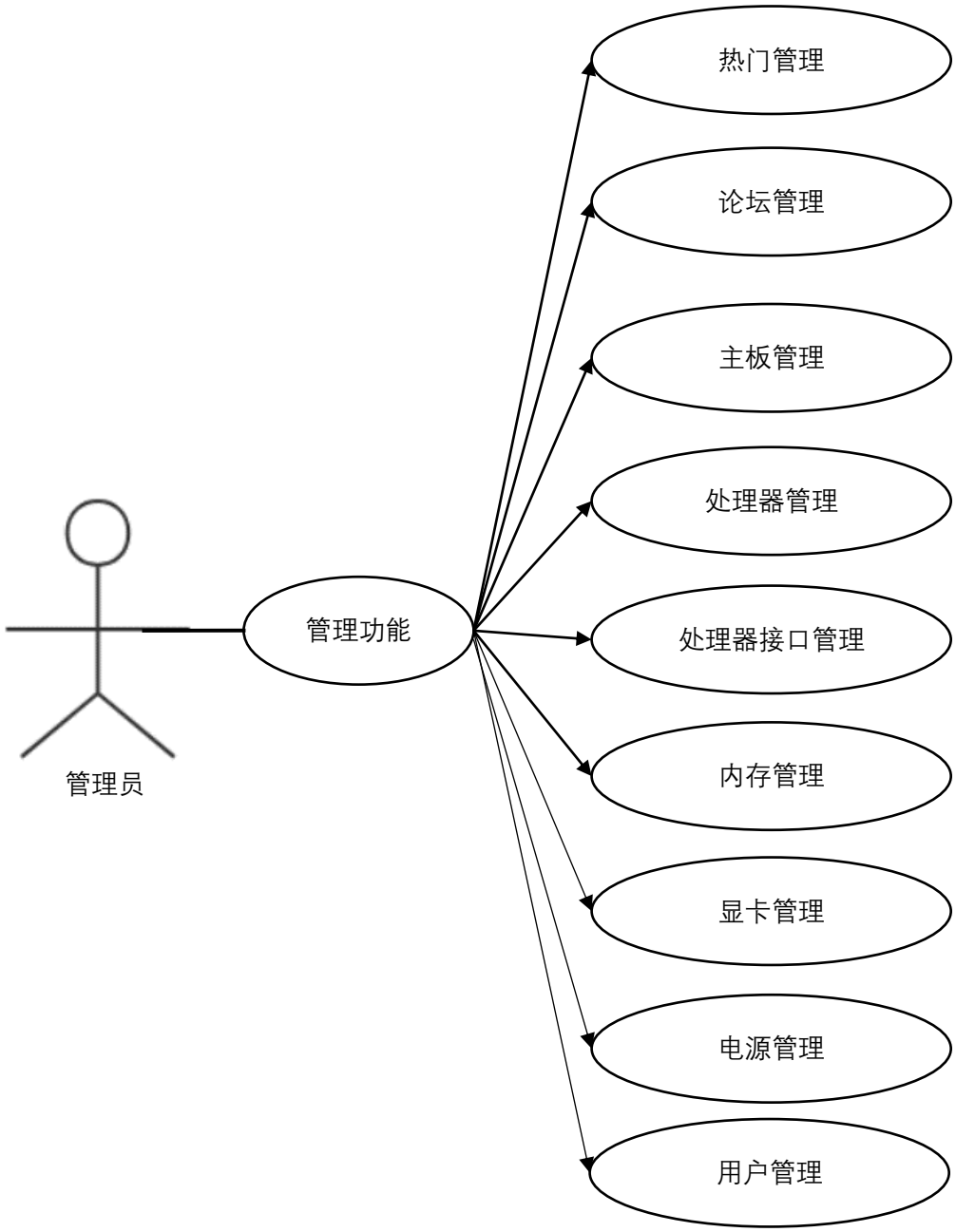

图 2.5 信息管理功能用例图

2.2 数据库设计

2.2.1 数据库表结构的关系

“CAC 电脑装机小屋”的各个表之间有主外键的联系，通过数据库表关系图能够很好的体现这种关系，如下图 2.6“CAC 电脑装机小屋”表关系图所示。

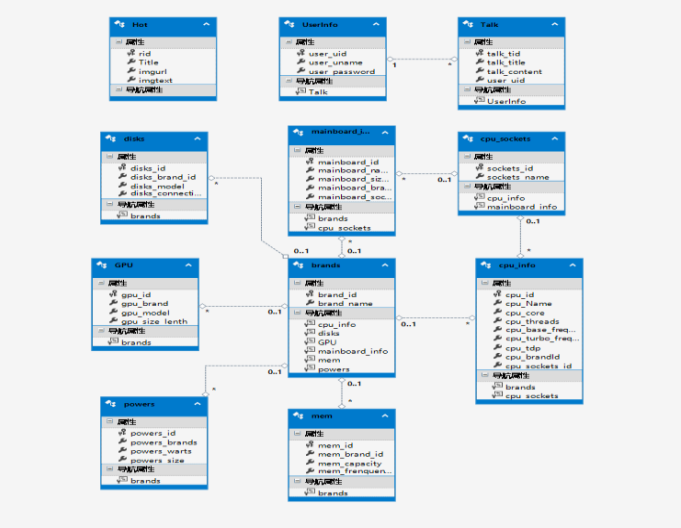

图 2.6 数据库表结构关系图

2.2.2 数据库 E-R 图设计

在设计“CAC 电脑装机小屋”使最先做的便是数据库表的设计，而 E-R 图是最好体现实体与属性最好的方式，如下图 2.7“CAC 电脑装机小屋”E-R 图所示。

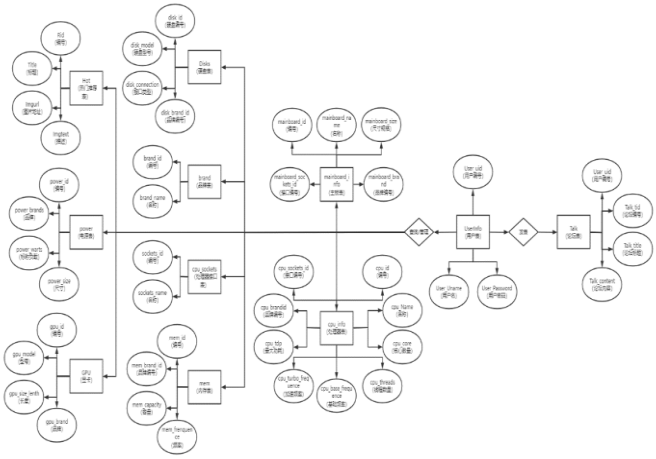

图 2.7 数据库表结构 E-R 图

2.2.3 数据库用户表的创建

设计用户表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.1 所示。

表 2.1 用户表（UserInfo）结构

字段显示	字段名	数据类型	默认值	备注说明
用户序号	user_uid	int		主键自动增长列
用户账号	user_uname	varchar(50)		不能为空

用户密码	user_password	varchar(30)		不能为空

2.2.4 数据库论坛表的创建

设计论坛表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.2 所示。

表 2.2 论坛表（Talk）结构

字段显示	字段名	数据类型	默认值	备注说明
讨论序号	talk_tid	int		主键自动增长列
讨论标题	talk_title	varchar(50)		不能为空
讨论内容	talk_content	text		不能为空
发言用户	user_uid	int		外键，与 UserInfo 表关联

2.2.5 数据库所有品牌表的创建

设计所有品牌表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.3 所示。

表 2.3 所有品牌表（brands）结构

字段显示	字段名	数据类型	默认值	备注说明
品牌编号	brand_id	int		主键自动增长列
品牌名称	brand_name	varchar(20)		不能为空

2.2.6 数据库处理器接口表的创建

设计处理器接口表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.4 所示。

表 2.4 处理器接口表（cpu_sockets）结构

字段显示	字段名	数据类型	默认值	备注说明
接口编号	sockets_id	int		主键自动增长列
接口名称	sockets_name	Varchar(20)		不能为空

2.2.7 数据库处理器信息表的创建

设计处理器信息表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.5 所示。

表 2.5 处理器信息表（cpu_info）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	cpu_id	Int		主键自动增长列
名称	cpu_Name	varchar (20)		不能为空
核心数量	cpu_core	Int		不能为空

线程数量	cpu_threads	int		不能为空
基础频率	cpu_base_frequency	numeric(4,2)		不能为空
加速频率	cpu_turbo_frequency	numeric(4,2)		不能为空
热设计功耗	cpu_tdp	int		不能为空
品牌编号	cpu_brandId	int		外键，与 brands 表关联
接口编号	cpu_sockets_id	int		外键，与 cpu_sockets 表关联

2.2.8 数据库主板信息表的创建

设计主板信息表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.6 所示。

表 2.6 主板信息表（mainboard_info）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	mainboard_id	int		主键自动增长列
型号	mainboard_name	varchar(30)		不能为空
尺寸类型	mainboard_sizeType	varchar(20)		不能为空
品牌编号	mainboard_brand	int		外键，与 brands 表关联
接口编号	mainboard_sockets_id	int		外键，与 cpu_sockets 表关联

2.2.9 数据库内存表的创建

设计内存表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.7 所示。

表 2.7 内存表（mem）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	mem_id	int		主键自动增长列
容量	mem_capacity	int		不能为空
频率	mem_frenquence	int		不能为空
品牌编号	mem_brand_id	int		外键，与 brads 表关联

2.2.10 数据库显卡表的创建

设计显卡表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.8 所示。

表 2.8 显卡表（GPU）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	gpu_id	int		主键自动增长列
型号	gpu_model	varchar(50)		不能为空

尺寸长度	gpu_size_lenth	int		不能为空
品牌编号	gpu_brand	int		外键，与 brands 表关联

2.2.11 数据库硬盘表的创建

设计硬盘表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.9 所示。

表 2.9 硬盘表（disks）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	disks_id	int		主键自动增长列
型号	disks_model	varchar(50)		不能为空
接口类型	disks_connection	varchar(20)		不能为空
品牌编号	disks_brand_id	int		外键，与 brands 表关联

2.2.12 数据库电源表的创建

设计电源表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.10 所示。

表 2.10 电源表（powers）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	powers_id	int		主键自动增长列
最大功率	powers_warts	int		不能为空
尺寸	powers_size	varchar(20)		不能为空
品牌编号	powers_brands	int		外键，与 brands 表关联

2.2.13 数据库热门表的创建

设计热门表的数据库结构，确定建表的数据名称，新建完成后，创建适当约束，添加相关数据。如表 2.11 所示。

表 2.11 热门表（Hot）结构

字段显示	字段名	数据类型	默认值	备注说明
编号	rid	int		主键自动增长列
标题	Title	nvarchar(20)		不能为空
图片	imgurl	text		不能为空
描述	imgtext	text		不能为空

第三章项目详细设计

3.1 系统总体设计

“装机小屋”的模块有用户登录模块、管理员登录模块、用户注册模块、管理员注册模块、首页模块、硬件信息展示模块，硬件信息论坛模块，硬件信息录入模块、管理员管理模块、联系我们模块和系统简介模块。

登录模块：主要是为了更好更安全的使用系统。

注册模块：是为了更好的管理系统。

首页模块：展现了近段时间的热门电脑硬件信息。

电脑硬件信息查询模块：对市场上所有硬件的信息进行统计、整理并展示。

硬件信息编辑模块：是管理员才有权限使用的模块，让管理员对硬件信息进行录入、修改和删除。

电脑硬件论坛模块：用户对相关硬件进行讨论交流，提出问题以及回答问题。

联系我们模块：是为了更好的增加用户体验，当用户遇到问题时可通过此模块来联络系统的维护人员来解决问题。

“装机小屋”的设计主要是面向想要了解电脑硬件信息的人群。本系统的核心功能是利用硬件信息展示模块向用户提供电脑硬件信息。其次就是利用硬件信息论坛模块向用户提供硬件讨论平台，方便用户交流相关硬件的使用的场景、性能表现等非纸面参数。此系统的的开发主要采用 ASP.NET、SQL SERVER、HTML、Bootstrap 等技术。

“装机小屋”总体设计如图 3.1 所示：

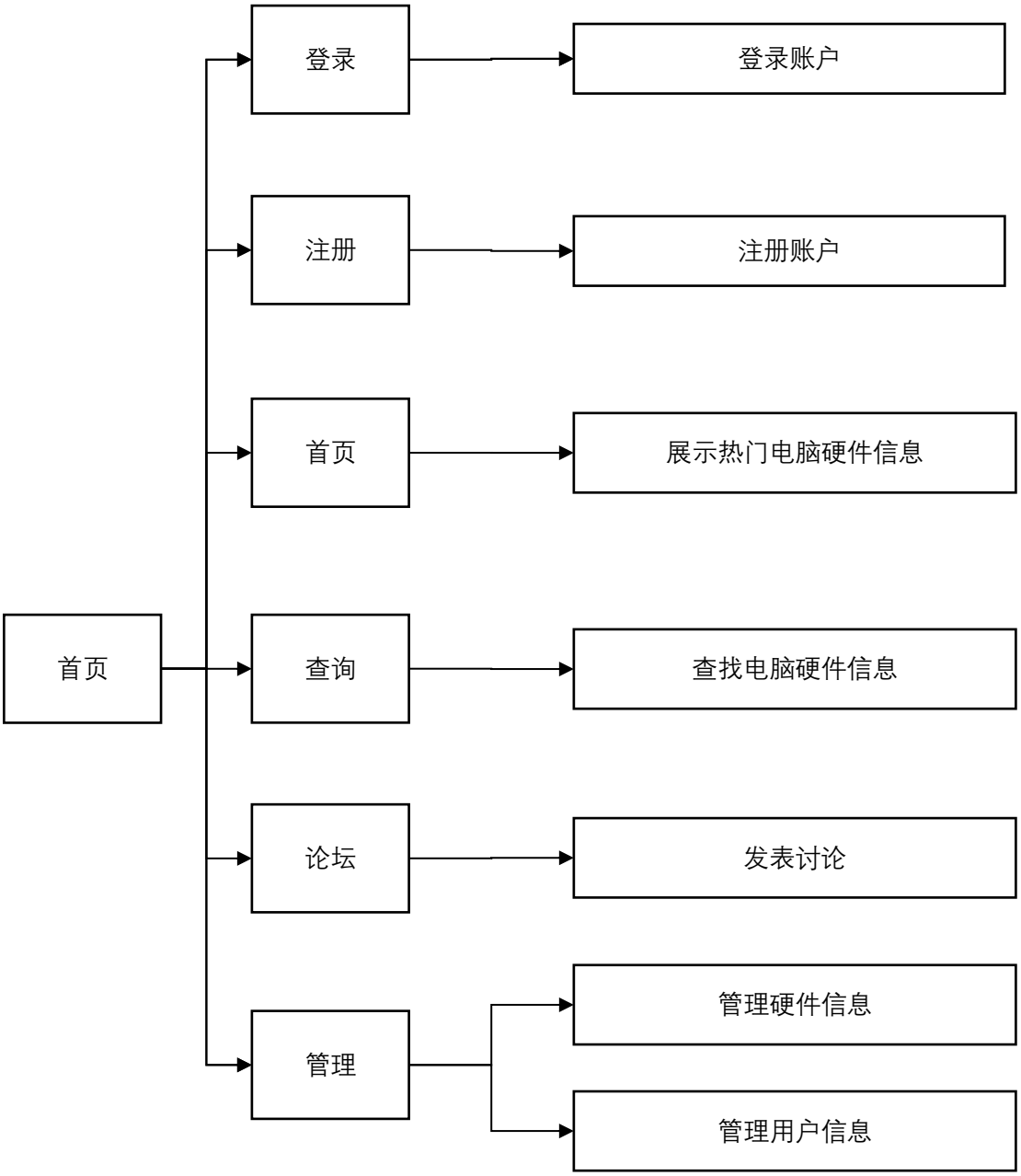

图 3.1“装机小屋”总体设计图

3.2 登录模块设计

3.2.1 登录功能介绍

为了保障整个系统的安全性，在用户想要进行一些账户交易等操作时，必须先成功登陆进系统。用户登录需要输入账户和密码，在用户输入完账号密码后，点击登录按钮，系统会将用户的账号密码与数据库匹对。数据匹配成功后提示用户登录成功，否则系统将会提示登录失败。

登录界面分为用户登录界面和管理员登录界面，依靠数据库中的信息来判断是否为管理员还是用户。界面简洁大方，让人看着非常舒服。可以通过管理员界面直接跳转到用户界面，也可以通过用户界面直接跳转到管理员界面。为了防止恶意破解密码、刷票、论坛灌水、刷页，界面中使用了验证码来进行判断登录。为方便用户们使用用户登录界面和管理员登录界面都使用了忘记密码跳转。为了照顾用户体验感登录界面使用了返回首页跳转。用户登录、管理员登录界面使用了三层架构 UI 层、BLL 层、BLL 层，便于维护和管理。

3.2.2 登录流程图

系统将角色划分为用户登录和管理员登录两个角色。用户主要是查看电脑硬件信息，而管理员则主要是管理电脑硬件信息，登录功能流程图如图 3.2 所示。

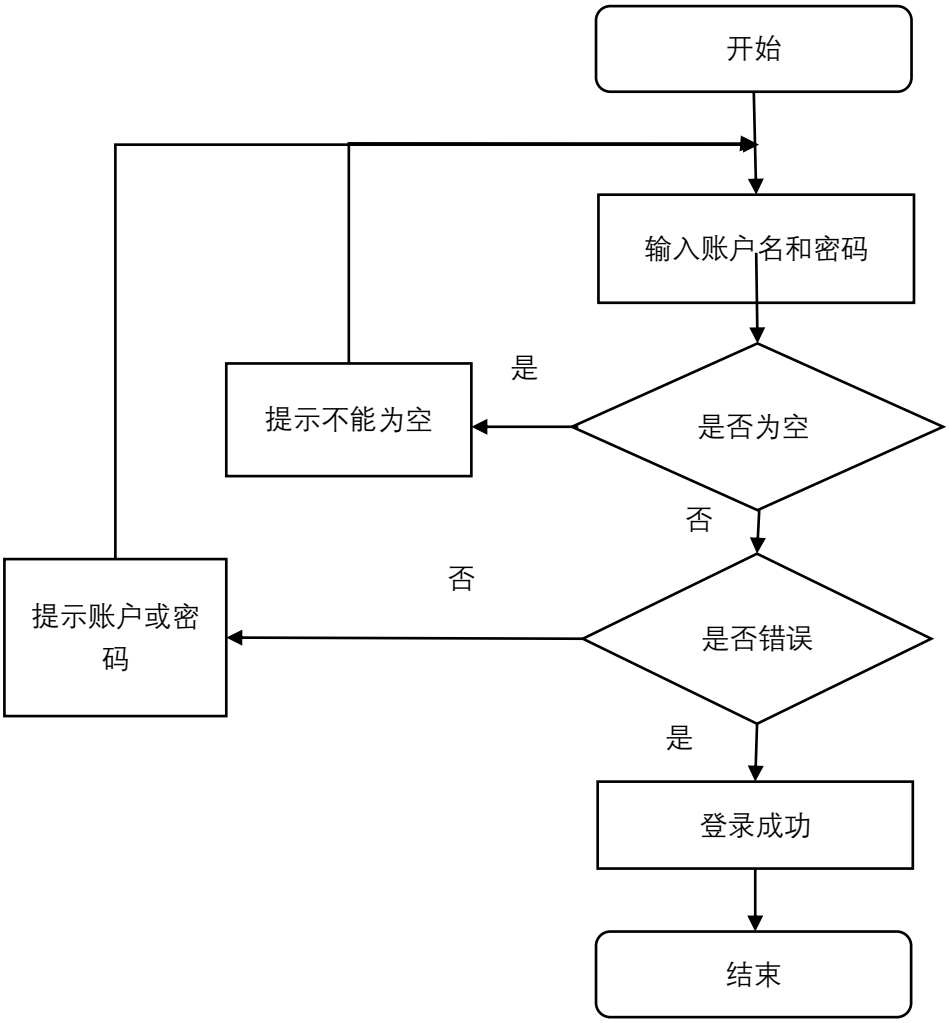

图 3.2 登录流程图

3.2.3 登录效果图

单击右上角登录按钮，弹出登录窗口，登录效果如图 3.3 所示：

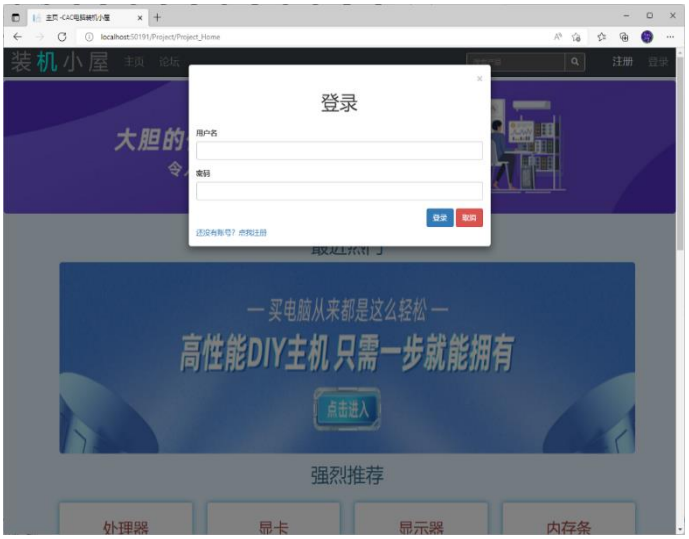

图 3.3 登录效果图

3.2.4 登录核心源代码

```
public ActionResult Project_Login(UserInfo m)
    {
        var i = db.UserInfo.Where(x => x.user_uname == m.user_uname && x.user_password == m.user_password).ToArray();
        if (i.Count() > 0)
        {
            Session["UserID"] = i[0].user_uid;
            TempData["msg"] = "登陆成功！";
            return RedirectToAction("Project_Home");
        }
        else
        {
            TempData["msg"] = "用户名或密码输入有误！";
            return RedirectToAction("Project_Home");
        }
    }
```

3.3 注册模块设计

3.3.1 注册功能介绍

用户需要进行一些账户交易等操作时，必须先登陆到系统中，而没有账户的用户则需要进行注册。注册需要用户填入账号、密码等相关信息，且两次输入的新密码必须完全一致，否则将会给出失败提示。在用户输入完账号密码后，点击注册按钮，系统自动将用户输入的账户名与数据库进行匹对。如果没有重复，即可成功注册。

3.3.2 注册流程图

用户注册。用户只查看数据信息，注册功能流程图如下所示。

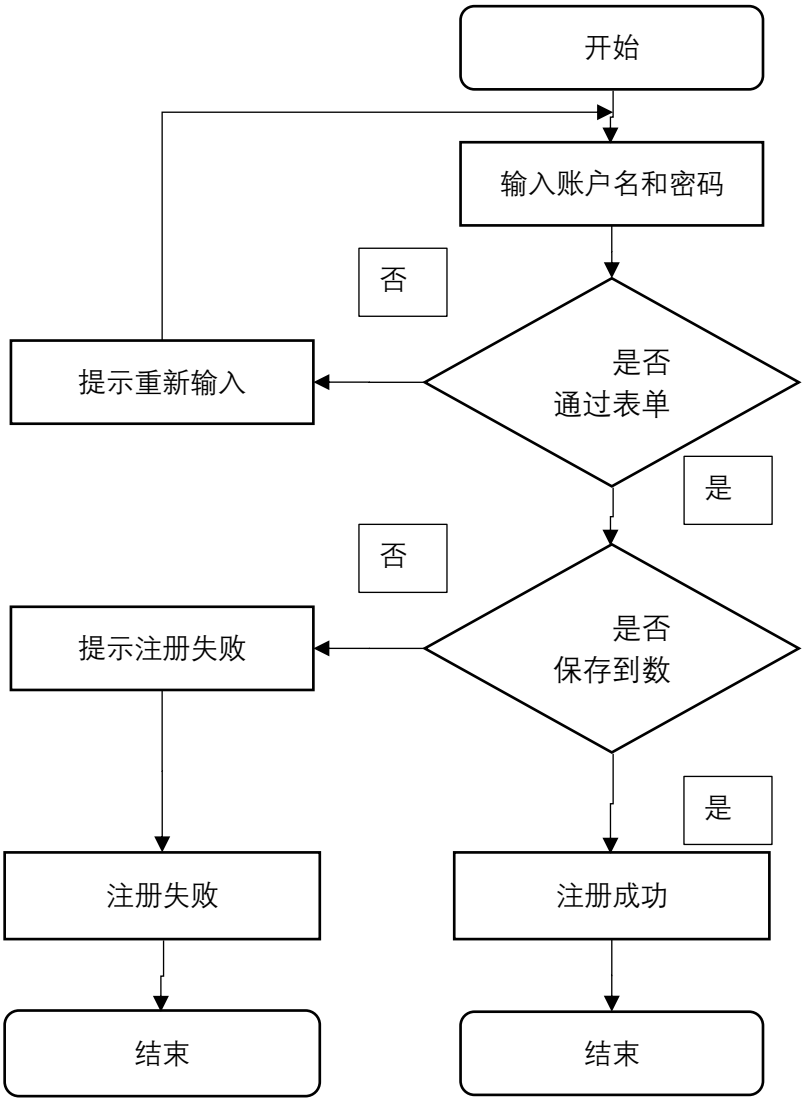

图 3.4 注册功能流程图

3.3.3 注册效果图

单击首页右上角注册按钮，弹出注册窗口，注册效果如图 3.5 所示：

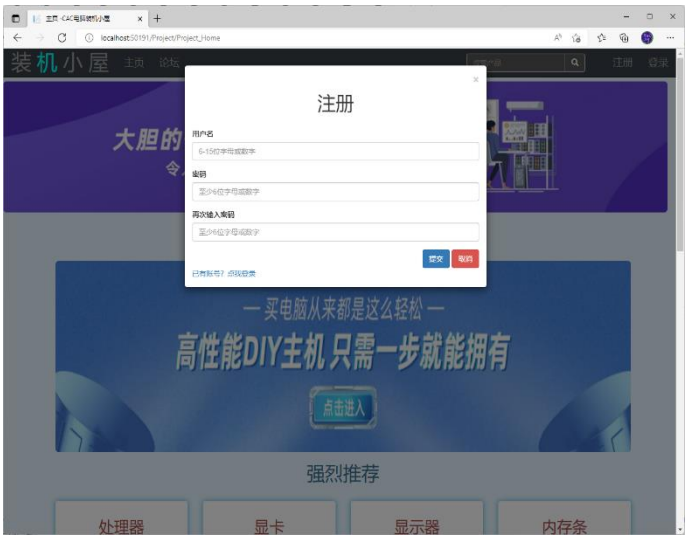

图 3.5 注册效果图

3.3.4 注册核心源代码

```
public ActionResult Project_Regist(UserInfo m)
    {
        db.UserInfo.Add(m);
        if (db.SaveChanges() > 0)
        {
            TempData["msg"] = "注册成功";
            return RedirectToAction("Project_Home");
        }
        TempData["msg"] = "注册失败";
        return RedirectToAction("Project_Home");
    }
```

3.4 首页模块设计

3.4.1 首页功能介绍

这是用户使用的主页面，导航栏部分有用户可以使用的功能清单。点击各个功能，网页自动跳转到各个功能的子菜单。每个功能都有各自对应的子菜单。首页的内容十分丰富，轮播图、线上服务、热门产品展示与最新咨询等模块合理搭配，吸引用户，用户能够选择自己感兴趣的

内容直接跳转到相应页面。该界面分为三大块：头部、中间部分、和尾部。头部是一个 carousel 轮播图，给用户提供最推荐信息。中间部分由上下两部分组成，上部分是热门硬件信息推荐之一。而下半部分是全部的电脑硬件推荐。尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.4.2 首页流程图

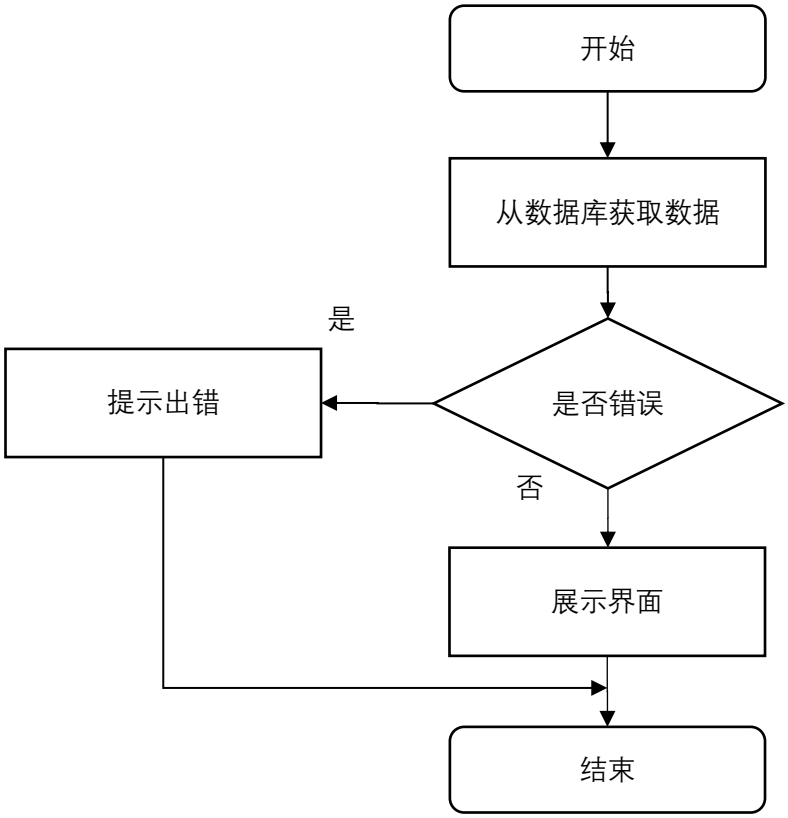

图 3.6 首页流程图

3.4.3 首页效果图

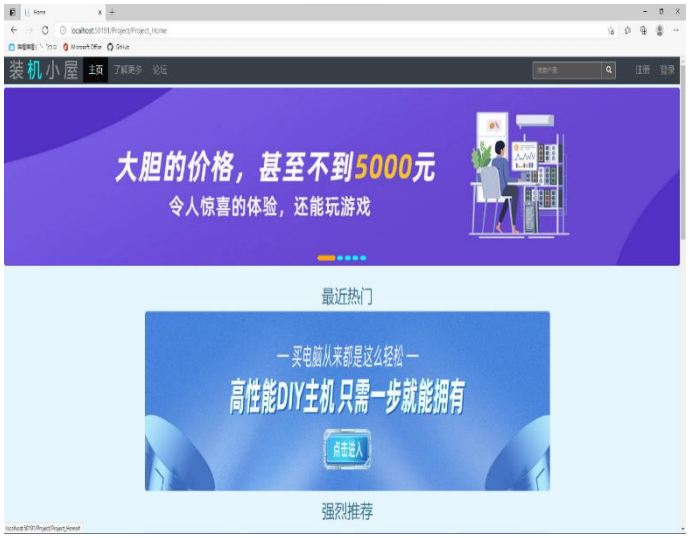

图 3.7 首页效果图

3.4.4 首页核心源代码

```
public ActionResult Project_Home()
        {
            ViewBag.Title = "主页";
            ViewBag.UserID = Session["UserID"] == null ? 0 : Session["UserID"];
            return View(db.Hot.ToList());
        }
```

3.5 了解更多模块设计

3.5.1 了解更多功能介绍

这是主页面的拓展页面，展示了主页推荐的详细的信息，用户使用的副面，导航栏部分有用户可以使用的功能清单。该界面分为三大块：头部、中间部分、和尾部。头部是一个导航栏，给用户提供功能选择。中间部分分为左右两个部分，左边展示了电脑硬件的图片，右边部分电脑硬件信息的描述文字。尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.5.2 了解更多流程图

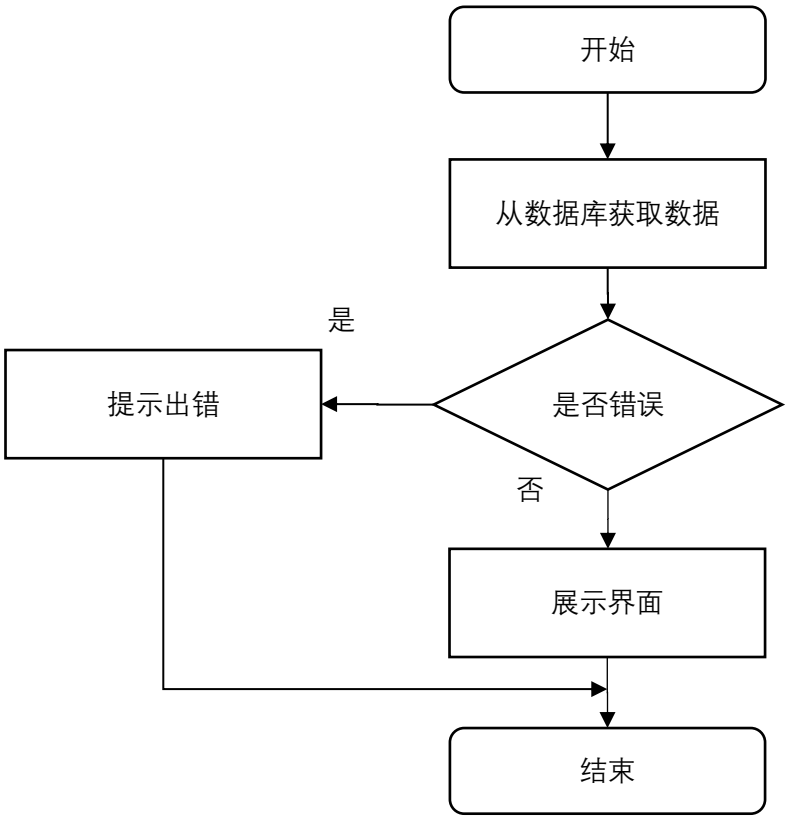

图 3.8 了解更多流程图

3.5.3 了解更多效果图

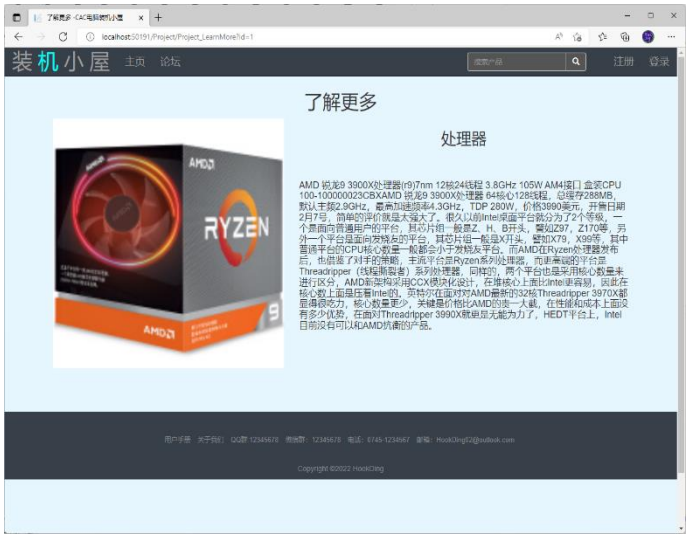

图 3.9 了解更多效果图

3.5.4 了解更多核心源代码

```
public ActionResult Project_LearnMore(int id = 0)
```

```
{
    ViewBag.Title = "了解更多";
    ViewBag.UserID = Session["UserID"] == null ? 0 : Session["UserID"];
    var b = new Hot();
    if (id != 0) {b = db.Hot.Find(id); }
    else{ b.rid = db.Hot.Count() + 1; }
    return View(b);
}
```

3.6 电脑硬件论坛模块模块设计

3.6.1 电脑硬件论坛模块功能介绍

论坛模块时给用户提供的一个发表讨论的模块，此模块主要包含了一个展示所有讨论的板块和发表讨论的表单，并且通过表单提交讨论内容并发表到平台。

3.6.2 电脑硬件论坛模块流程图

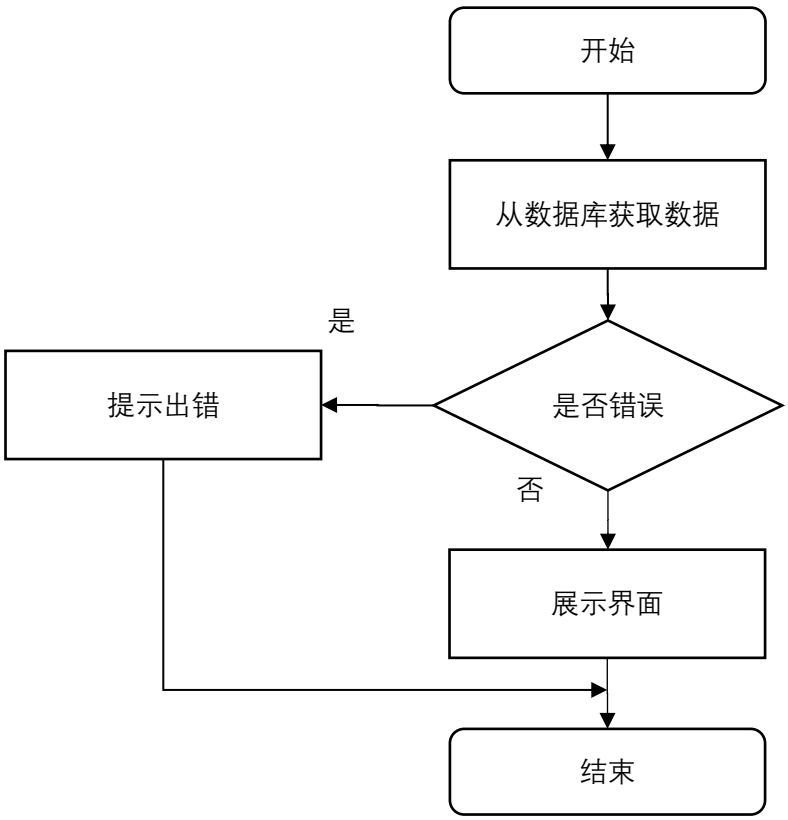

图 3.10 电脑硬件论坛模块流程图

3.6.3 电脑硬件论坛模块效果图

图 3.11 电脑硬件论坛模块效果图

3.6.4 电脑硬件论坛模块核心源代码

```
public ActionResult Project_Talk(int id = 0)
        {
            ViewBag.Title = "论坛";
            ViewBag.UserID = Session["UserID"] == null ? 0 : Session["UserID"];
            return View(db.Talk.ToList());
        }
```

3.7 管理首页模块设计

3.7.1 管理首页模块功能介绍

这个页面主要是管理员使用的页面，该页面分为上中下三个部分，上部分是导航栏，向管理员提供管理选择，中间部分是一个巨幕，展示文字欢迎界面；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.7.2 管理首页模块流程图

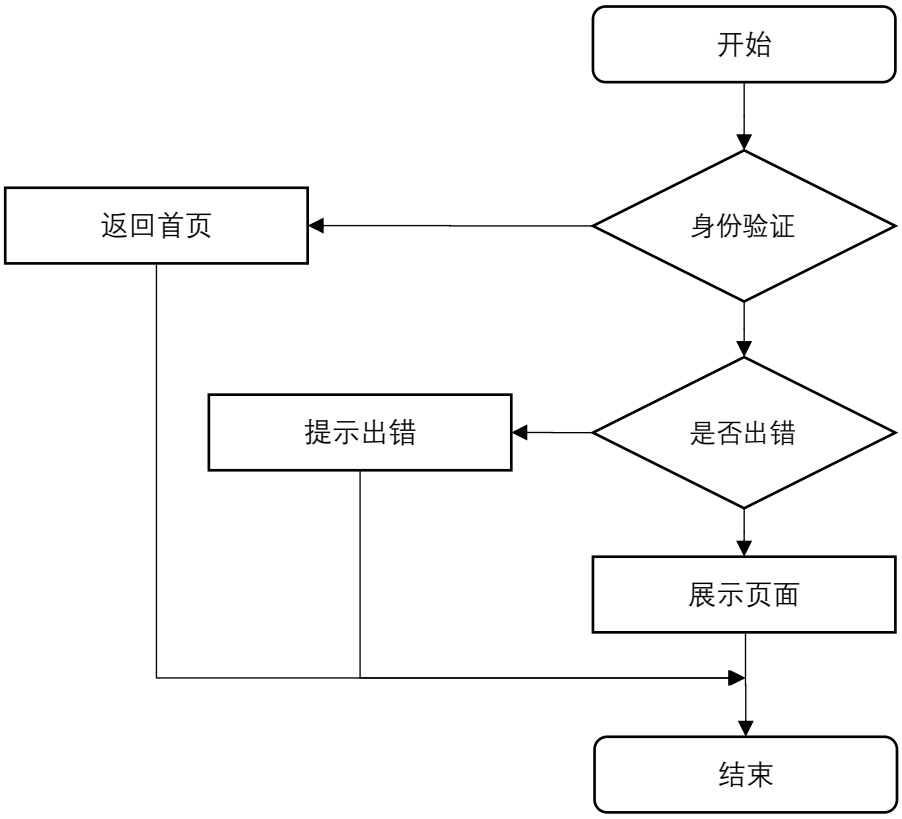

图 3.12 管理首页模块流程图

3.7.3 管理首页模块效果图

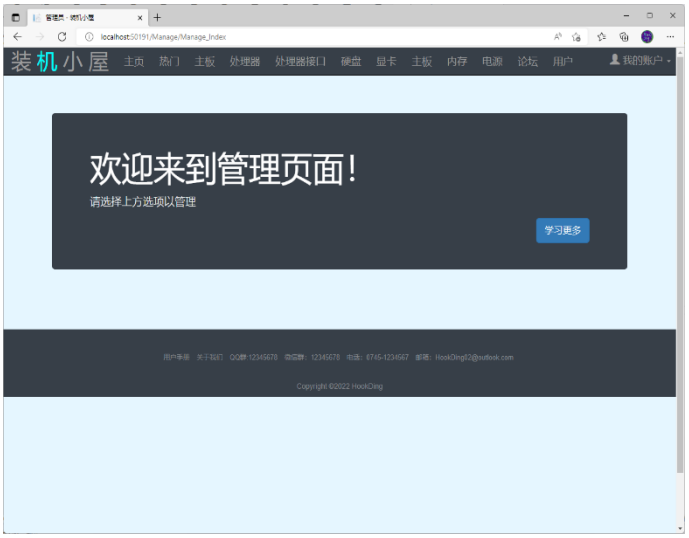

图 3.13 管理首页模块效果图

3.7.4 管理首页模块核心源代码

```
@model List<MVC_GP.Models.Hot>
@{
    ViewBag.Title = "管理员";
    Layout = "~/Views/Shared/_LayoutManager.cshtml";
}
<br /><br /><br />
<div class="container">
    <div class="row">
        <div class="col-md-12 ">
            <div class="jumbotron bg-black" style="color:white;">
                <div class="container">
                    <h1>欢迎来到管理页面！</h1>
                    <p>请选择上方选项以管理</p>
                    <p>
                        <a class="btn btn-primary btn-lg pull-right" role="button" href="">
                            学习更多
                        </a>
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
```

3.8 热门管理模块设计

3.8.1 热门管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏，向管理员提供管理选择，中间部分主要内容，由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.8.2 热门管理模块流程图

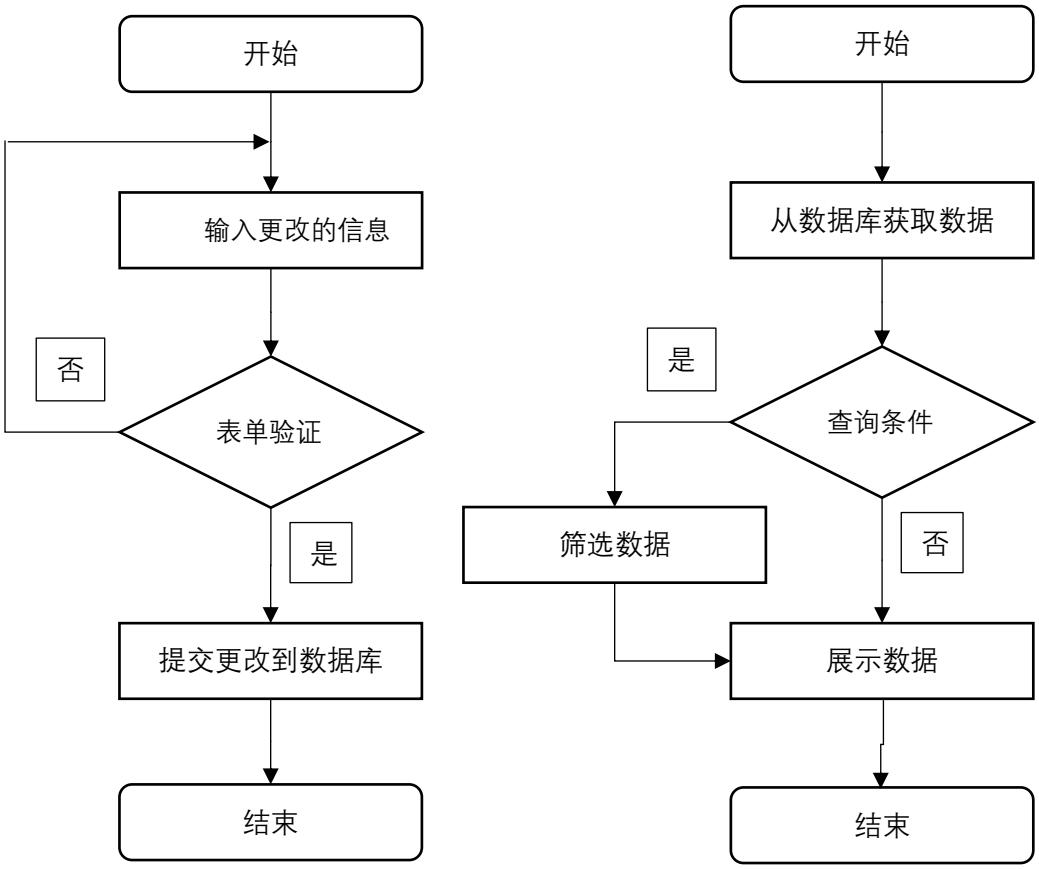

图 3.14 热门管理模块流程图

3.8.3 热门管理模块效果图

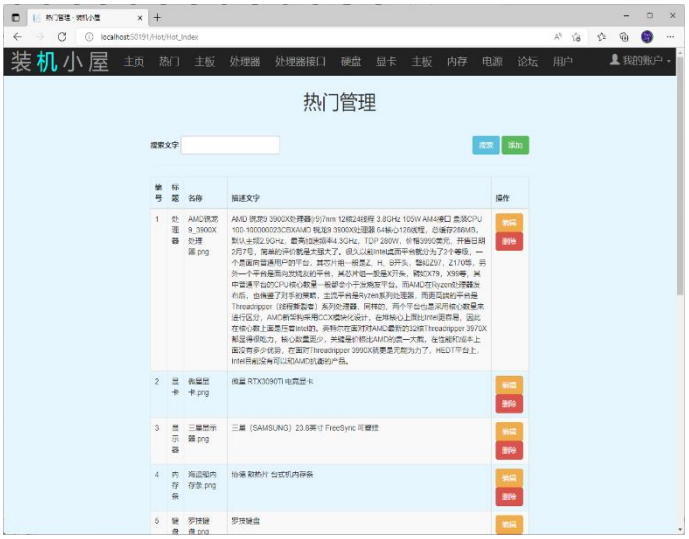

图 3.15 热门管理模块效果图

3.8.4 热门管理模块核心源代码

```
public class HotController : Controller
   { ProjectDBEntities db = new ProjectDBEntities();
       // Hot_Index
       public ActionResult Hot_Index(string imgtext)
       { ViewBag.Title = "热门管理";
           ViewBag.imgtext = imgtext;
           var b = db.Hot
               .Where(t => string.IsNullOrEmpty(imgtext) || t.imgtext.Contains(imgtext))
               .ToList();
           return View(b);
       }
       public ActionResult Hot_Edit(int id=0)
       {var m = new Hot();
           if (id == 0) {//添加
               ViewBag.Title = "添加热门";
               var g = db.Hot.Count();
               m = new Hot { rid = g + 1 };
           }else{//编辑
               ViewBag.Title = "修改热门";
               m = db.Hot.Find(id); }
           return View(m);
       }
       // DoEdit
       [HttpPost]
       public ActionResult Hot_Edit(Hot m)
       {
           if (m.rid > db.Hot.Count()){ //添加
               db.Hot.Add(m);
               db.SaveChanges();
           }else{//编辑
```

```
            db.Entry(m).State = System.Data.Entity.EntityState.Modified;
            db.SaveChanges();}
        return RedirectToAction("Hot_Index");
    }
    // Delete
    public ActionResult Hot_Delete(int id=0)
    {
        if (id != 0) {
            db.Hot.Remove(db.Hot.Find(id));
            db.SaveChanges();}
        return RedirectToAction("Hot_Index");
    }
}
```

3.9 主板管理模块设计

3.9.1 主板管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏，向管理员提供管理选择，中间部分主要内容，由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.9.2 主板管理模块流程图

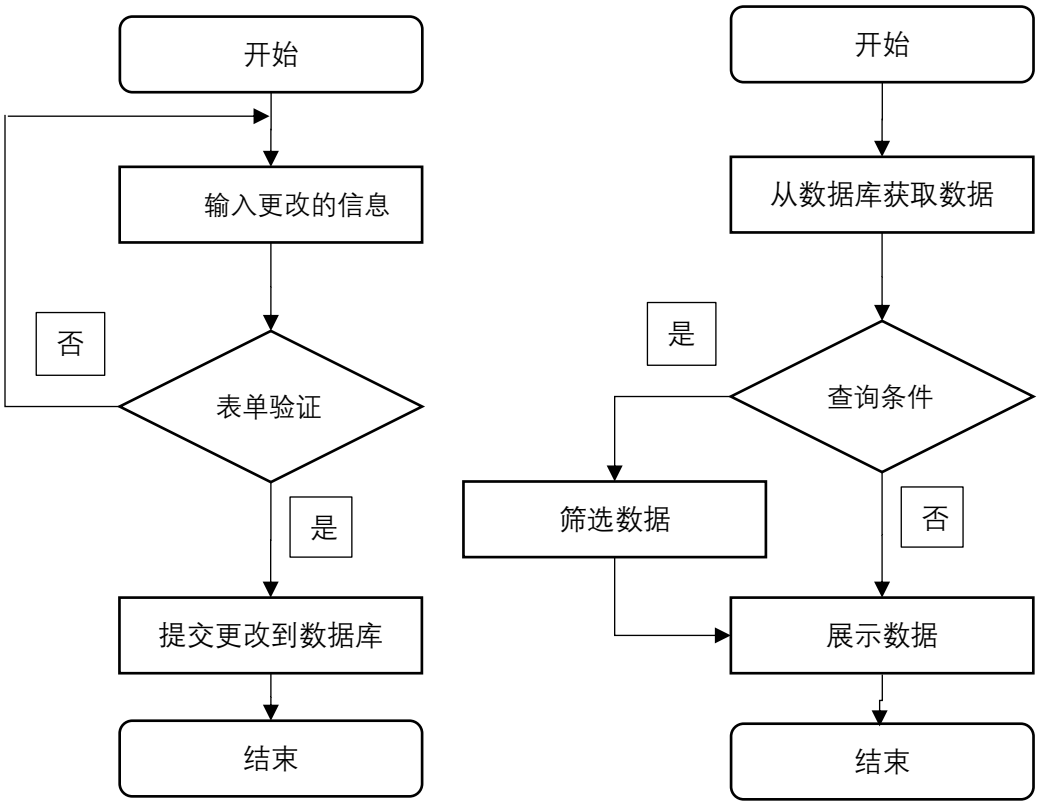

图 3.16 主板管理模块流程图

3.9.3 主板管理模块效果图

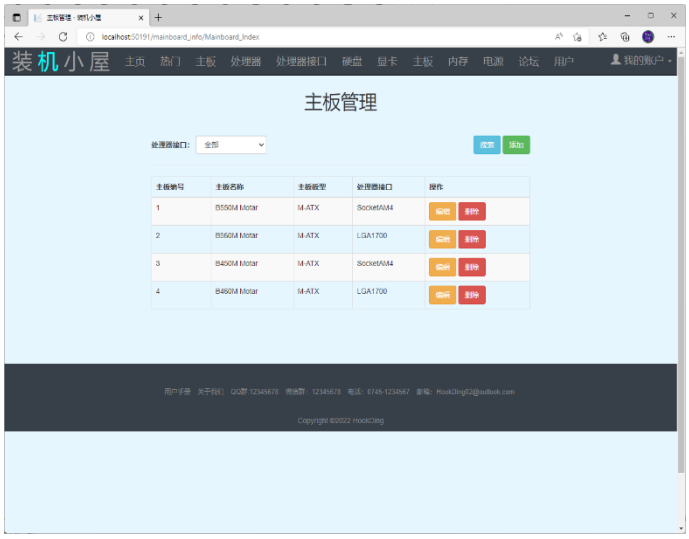

图 3.17 主板管理模块效果图

3.9.4 主板管理模块核心源代码

```
public class mainboard_infoController : Controller
```

```
{
    ProjectDBEntities db = new ProjectDBEntities();
    // Mainboard_Index
    public ActionResult Mainboard_Index(int mainboard_sockets_id = 0)
    {ViewBag.Title = "主板管理";
        ViewBag.mainboard_sockets_id  =  new  SelectList(db.cpu_sockets.ToList(),"sockets_id",
        "sockets_name", mainboard_sockets_id);
        var b = db.mainboard_info
            .Where(x=>mainboard_sockets_id==0|| x.mainboard_sockets_id == mainboard_sockets_id)
            .ToList();
        return View(b); }
    // Edit
    public ActionResult Mainboard_Edit(int id = 0)
    {var m = new mainboard_info();
      if (id == 0) {//添加
            ViewBag.Title = "添加主板";
            var g = db.disks.Count();
            m = new mainboard_info { mainboard_id = g + 1 , mainboard_sockets_id =0};}
        else{//编辑
            ViewBag.Title = "修改主板";
            m = db.mainboard_info.Find(id); }
        ViewBag.mainboard_sockets_id = new SelectList(db.cpu_sockets.ToList(), "sockets_id",
        "sockets_name", m.mainboard_sockets_id);
        return View(m);
    }
    // DoEdit
    [HttpPost]
    public ActionResult Mainboard_Edit(mainboard_info m)
    {
        if (m.mainboard_id > db.mainboard_info.Count()){//添加
            db.mainboard_info.Add(m);
```

```
            db.SaveChanges();
        }
        else{//编辑
            db.Entry(m).State = System.Data.Entity.EntityState.Modified;
            db.SaveChanges();}
       return RedirectToAction("Mainboard_Index");
    }
    // Delete
    public ActionResult Mainboard_Delete(int id = 0)
    {
        if (id != 0) {
            db.mainboard_info.Remove(db.mainboard_info.Find(id));
            db.SaveChanges();}
        return RedirectToAction("Mainboard_Index");
    }
}
```

3.10 处理器管理模块设计

3.10.1 处理器管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏，向管理员提供管理选择，中间部分主要内容，由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.10.2 处理器管理模块流程图

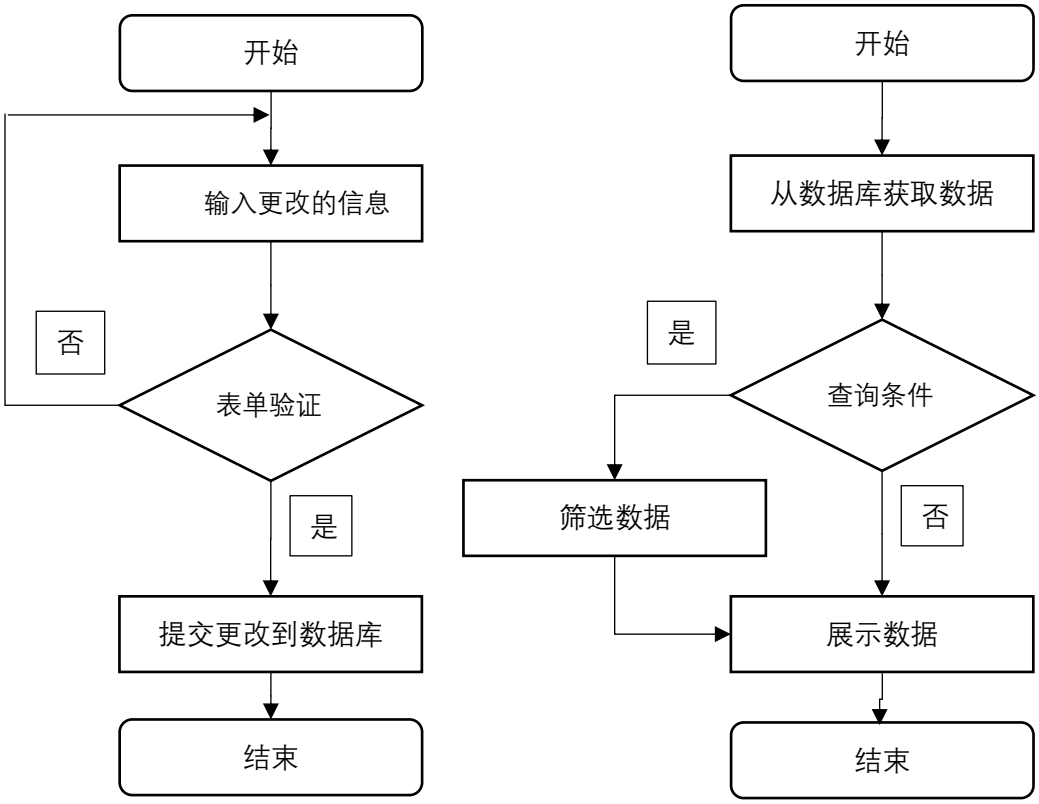

图 3.18 处理器管理模块流程图

3.10.3 处理器管理模块效果图

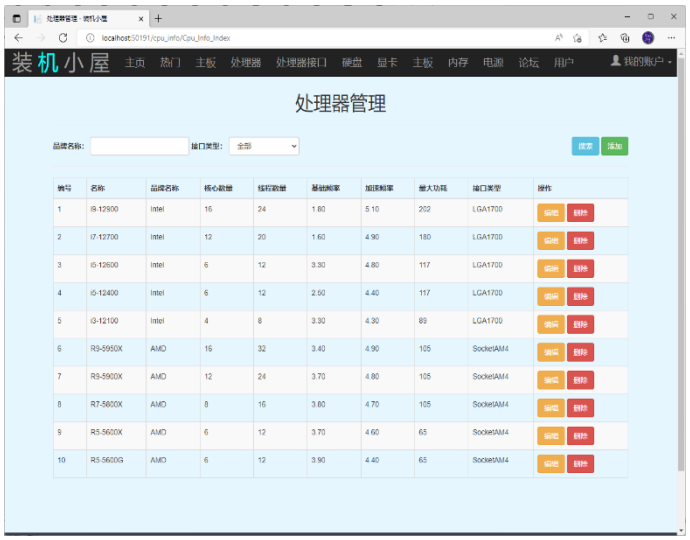

图 3.19 处理器管理模块效果图

3.10.4 处理器管理模块核心源代码

```
public class cpu_infoController : Controller
```

```
{
    ProjectDBEntities db = new ProjectDBEntities();
    public ActionResult Cpu_Info_Index(string brand_name,int cpu_sockets_id=0)
    {
        ViewBag.Title = "处理器管理";
        ViewBag.brand_name = brand_name;
        ViewBag.cpu_sockets_id=new SelectList(db.cpu_sockets.ToList(),
        "sockets_id","sockets_name", cpu_sockets_id);
        var b = db.cpu_info
        .Where(t=>string.IsNullOrEmpty(brand_name)||t.brands.brand_name.Contains(brand_na
        me)).Where(t=> cpu_sockets_id == 0||t.cpu_sockets_id== cpu_sockets_id).ToList();
        return View(b);
    }
    public ActionResult Cpu_Info_Edit(int id=0)
    {
        var m = new cpu_info();
        if (id == 0) {//新建
            ViewBag.Title = "添加处理器";
            //var g = db.cpu_info.Count();
            //m = new cpu_info { cpu_id = g + 1 };
        }else{//修改
            ViewBag.Title = "修改处理器";
            m = db.cpu_info.Find(id);
        }
        ViewBag.cpu_brandId = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
        m.cpu_brandId);
        ViewBag.cpu_sockets_id=new        SelectList(db.cpu_sockets.ToList(),    "sockets_id",
        "sockets_name", m.cpu_sockets_id);
        return View(m);
    }
    [HttpPost]
```

```
    public ActionResult Cpu_Info_Edit(cpu_info m)
    {
        try{
            if (m.cpu_id == 0) { //新建
                db.cpu_info.Add(m);
                db.SaveChanges();
            }else{//修改
                db.Entry(m).State = System.Data.Entity.EntityState.Modified;
                db.SaveChanges();
            }
            return RedirectToAction("Cpu_Info_Index");
        }catch{
            ViewBag.Erroy = "出错！";
            return RedirectToAction("Cpu_Info_Edit");
        }
    }

    public ActionResult Cpu_Info_Delete(int id=0)
    {
        try{
            if (id != 0) {
                db.cpu_info.Remove(db.cpu_info.Find(id));
                db.SaveChanges();}
            return RedirectToAction("Cpu_Info_Index");
        }catch{
            ViewBag.Erroy = "删除失败！";
            return RedirectToAction("Cpu_Info_Index");
        }
    }
}
```

3.11 处理器接口管理模块设计

3.11.1 处理器接口管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.11.2 处理器接口管理模块流程图

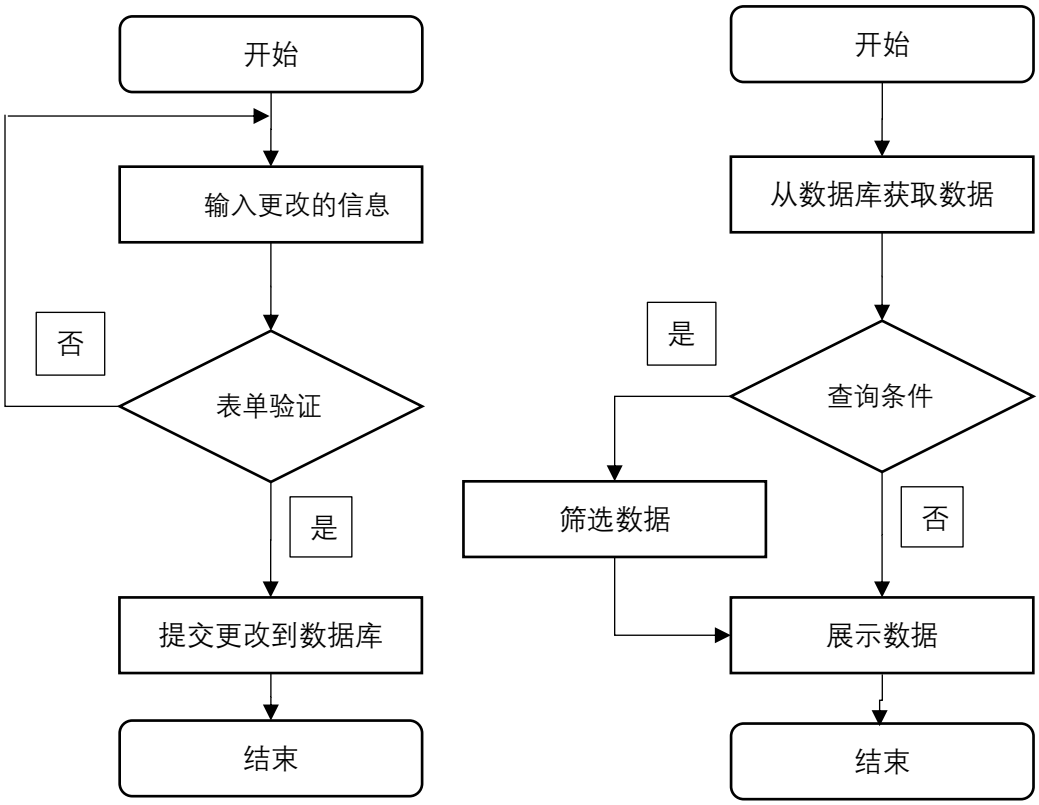

图 3.20 处理器接口管理模块流程图

3.11.3 处理器接口管理模块效果图

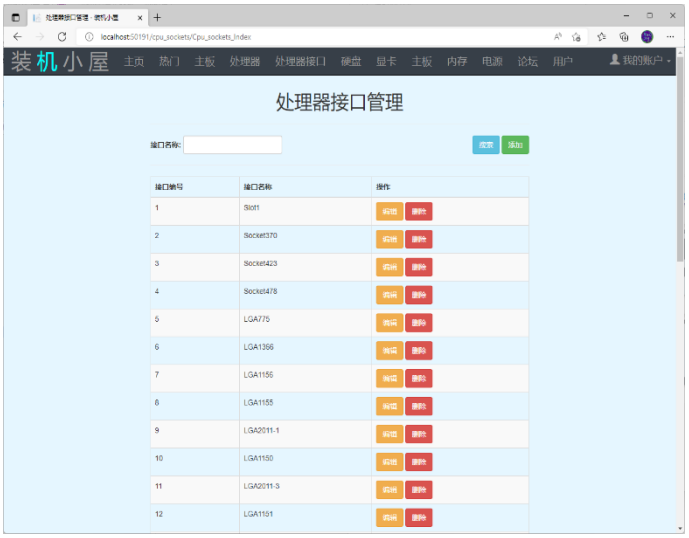

图 3.21 处理器接口管理模块效果图

3.11.4 处理器接口管理模块核心源代码

```
public class cpu_socketsController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        public ActionResult Cpu_sockets_Index(string sockets_name)
        {
            ViewBag.Title = "处理器接口管理";
            ViewBag.Sockets_name = sockets_name;
            var b = db.cpu_sockets
              .Where(t=>string.IsNullOrEmpty(sockets_name)||t.sockets_name.Contains(sockets_name))
                .ToList();
            return View(b);
        }
        public ActionResult Cpu_sockets_Edit(int id=0)
        {
            var m = new cpu_sockets();
            if (id == 0) {//新建
                ViewBag.Title = "添加处理器接口";
                var g = db.cpu_sockets.Count();
```

```
            m = new cpu_sockets { sockets_id = g + 1 };
        }else{//修改
            ViewBag.Title = "修改处理器接口";
            m = db.cpu_sockets.Find(id); }
        return View(m);
    }
    [HttpPost]
    public ActionResult Cpu_sockets_Edit(cpu_sockets m)
    {
        if (m.sockets_id == 0) {//新建
            db.cpu_sockets.Add(m);
            db.SaveChanges();
        }else{//修改
            db.Entry(m).State = System.Data.Entity.EntityState.Modified;
            db.SaveChanges();}
        return RedirectToAction("Cpu_sockets_Index");
    }
    public ActionResult Cpu_sockets_Delete(int id, FormCollection collection)
    {
        try{
            if (id != 0) {
                db.cpu_sockets.Remove(db.cpu_sockets.Find(id));
                db.SaveChanges();}
            return RedirectToAction("Cpu_sockets_Index");
        }catch{
            ViewBag.Erroy = "删除失败！";
            return RedirectToAction("Cpu_sockets_Index");
        }
    }
}
```

3.12 硬盘管理模块设计

3.12.1 硬盘管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.12.2 硬盘管理模块流程图

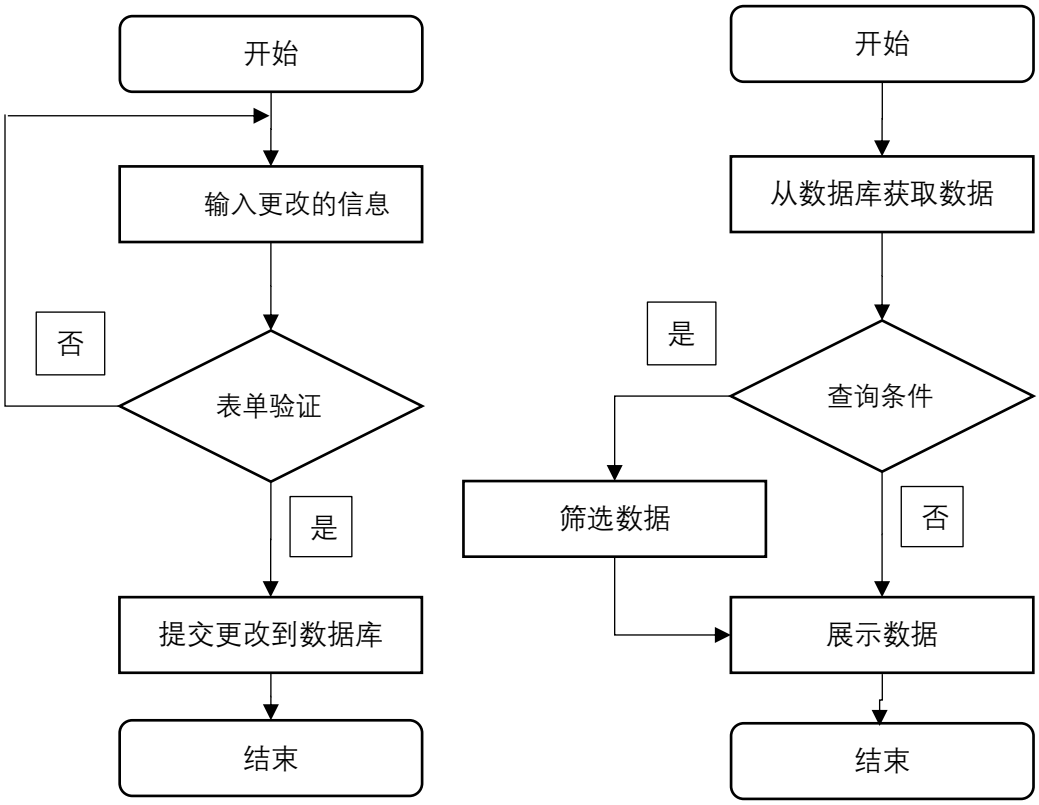

图 3.22 硬盘管理模块流程图

3.12.3 硬盘管理模块效果图

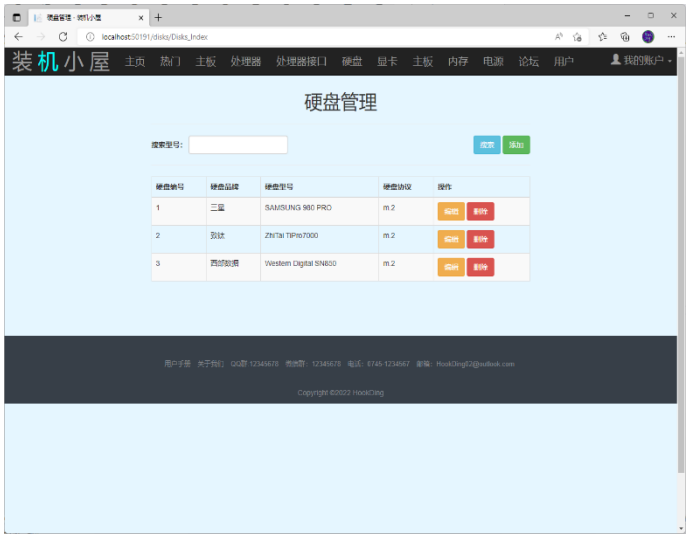

图 3.23 硬盘管理模块效果图

3.12.4 硬盘管理模块核心源代码

```
public class GPUController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // GPU_Index
        public ActionResult GPU_Index(string gpu_model)
        {
            ViewBag.Title = "显卡管理";
            ViewBag.gpu_model = gpu_model;
            var b = db.GPU
                .Where(t => string.IsNullOrEmpty(gpu_model) || t.gpu_model.Contains(gpu_model))
                .ToList();
            return View(b);
        }
        // Edit
        public ActionResult GPU_Edit(int id=0)
        {
            var m = new GPU();
            if (id == 0) {//添加
```

```
            ViewBag.Title = "添加显卡";
            var g = db.disks.Count();
            m = new GPU { gpu_id = g + 1 };
        }else{
            //编辑
            ViewBag.Title = "修改显卡";
            m = db.GPU.Find(id);
        }
        ViewBag.gpu_brand = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
        m.gpu_brand);
        return View(m);
    }
    // DoEdit
    [HttpPost]
    public ActionResult GPU_Edit(GPU m)
    {
        if (m.gpu_id > db.GPU.Count()){ //添加
            db.GPU.Add(m);
            db.SaveChanges();
        }else{ //编辑
            db.Entry(m).State = System.Data.Entity.EntityState.Modified;
            db.SaveChanges();
        }
        return RedirectToAction("GPU_Index");
    }
    // Delete
    public ActionResult GPU_Delete(int id, FormCollection collection)
    {
        try{
            if (id != 0) {
                db.GPU.Remove(db.GPU.Find(id));
```

```
                db.SaveChanges();}
            return RedirectToAction("GPU_Index");
        }catch{
            return RedirectToAction("GPU_Index");
        }
    }
}
```

3.13 显卡管理模块设计

3.13.1 显卡管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.13.2 显卡管理模块流程图

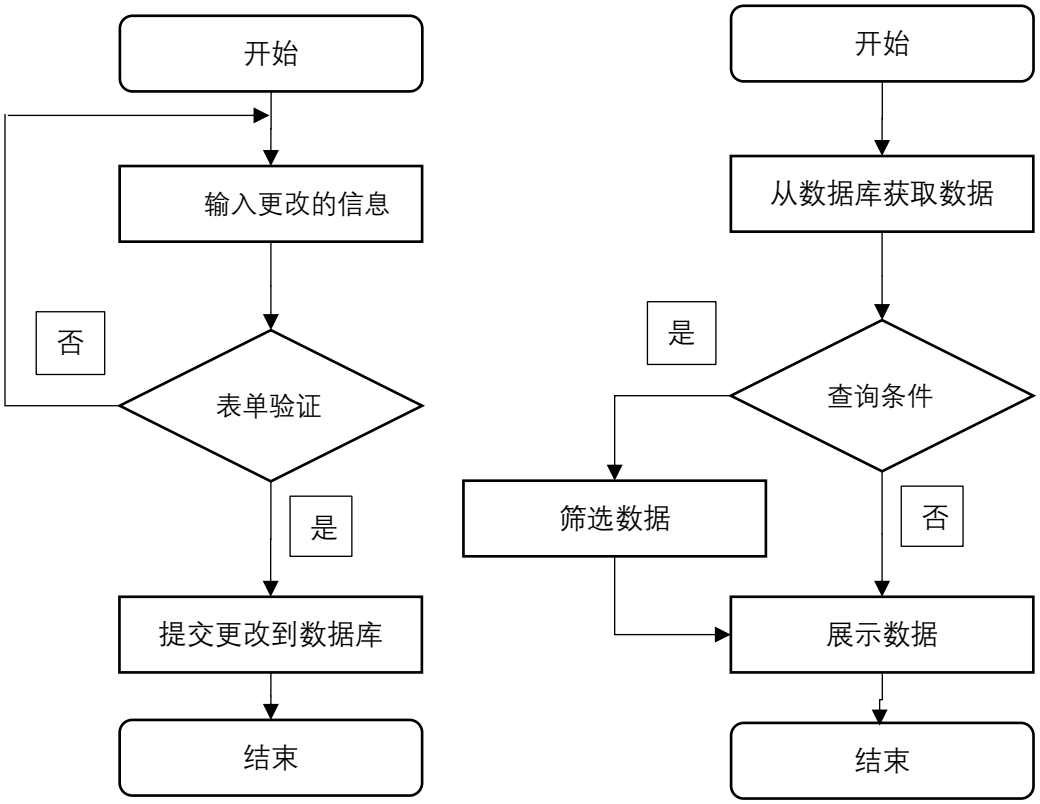

图 3.24 显卡管理模块流程图

3.13.3 显卡管理模块效果图

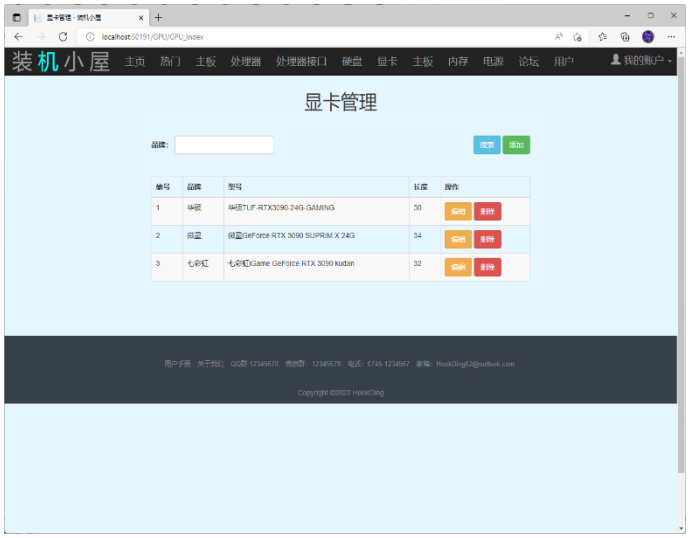

图 3.25 显卡管理模块效果图

3.13.4 显卡管理模块核心源代码

```
public class GPUController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // GPU_Index
        public ActionResult GPU_Index(string gpu_model)
        {
            ViewBag.Title = "显卡管理";
            ViewBag.gpu_model = gpu_model;
            var b = db.GPU
                .Where(t => string.IsNullOrEmpty(gpu_model) || t.gpu_model.Contains(gpu_model))
                .ToList();
            return View(b);
        }
        // Edit
        public ActionResult GPU_Edit(int id=0)
        {
            var m = new GPU();
            if (id == 0) {//添加
```

```
        ViewBag.Title = "添加显卡";
        var g = db.disks.Count();
        m = new GPU { gpu_id = g + 1 };
    }else{//编辑
        ViewBag.Title = "修改显卡";
        m = db.GPU.Find(id);
    }
    ViewBag.gpu_brand = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
    m.gpu_brand);
    return View(m);
}
// DoEdit
[HttpPost]
public ActionResult GPU_Edit(GPU m)
{
    if (m.gpu_id > db.GPU.Count()){//添加
        db.GPU.Add(m);
        db.SaveChanges();
    }else{//编辑
        db.Entry(m).State = System.Data.Entity.EntityState.Modified;
        db.SaveChanges();
    }
    return RedirectToAction("GPU_Index");
}
// Delete
public ActionResult GPU_Delete(int id, FormCollection collection)
{
    try{
        if (id != 0) {
            db.GPU.Remove(db.GPU.Find(id));
            db.SaveChanges();}
```

```
                return RedirectToAction("GPU_Index");
            } catch{
                return RedirectToAction("GPU_Index");
            }
        }
    }
```

3.14 内存管理模块设计

3.14.1 内存管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.14.2 内存管理模块流程图

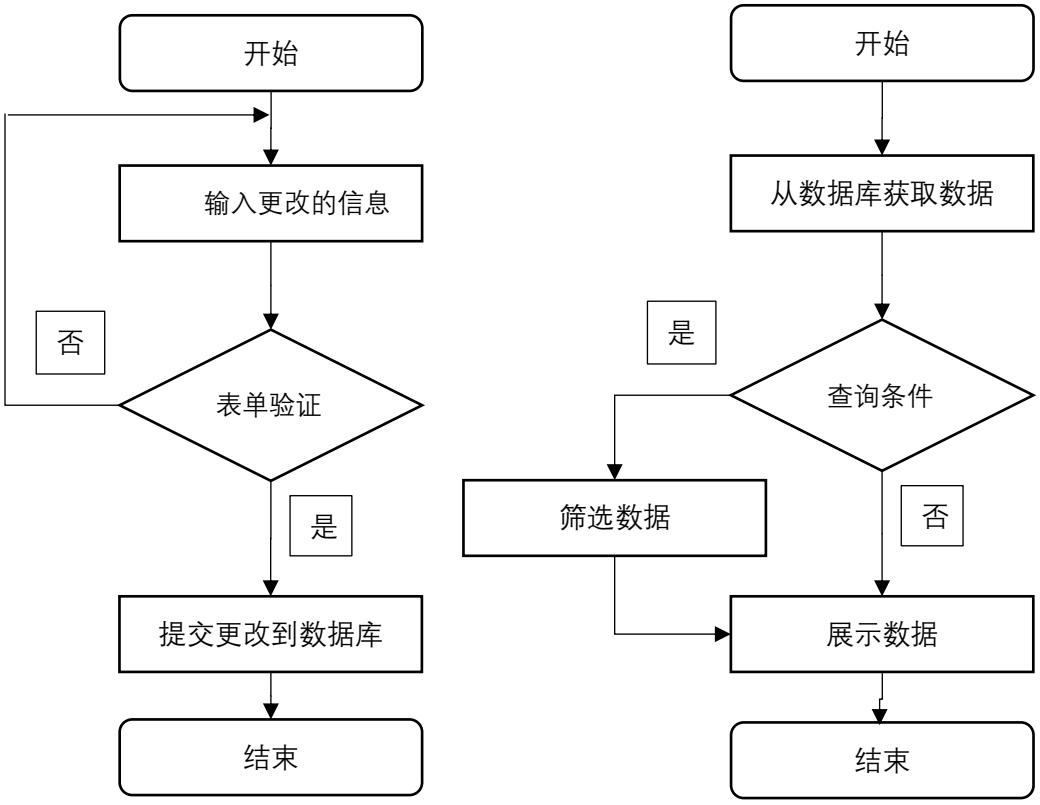

图 3.26 内存管理模块流程图

3.14.3 内存管理模块效果图

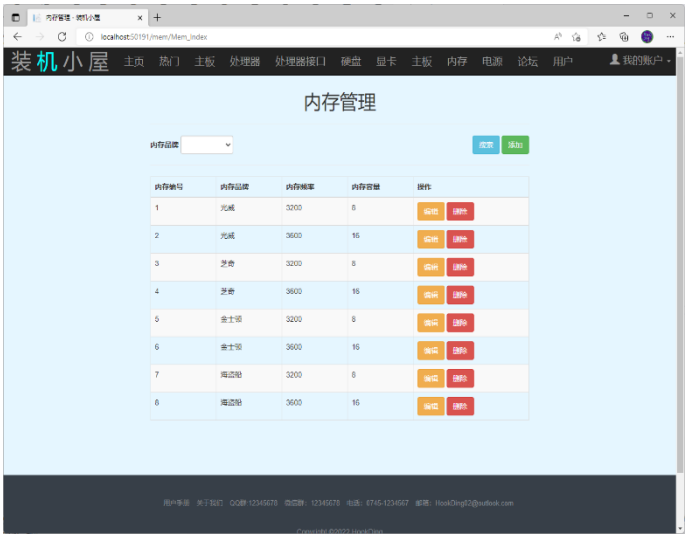

图 3.27 内存管理模块效果图

3.14.4 内存管理模块核心源代码

```
public class memController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // Mem_Index
        public ActionResult Mem_Index(int mem_brand_id=0)
        {
            ViewBag.Title = "内存管理";
            ViewBag.mem_brand_id = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
            mem_brand_id);
            var b = db.mem
                .Where(x => mem_brand_id==0|| x.mem_brand_id == mem_brand_id)
                .ToList();
            return View(b);
        }
        // Edit
        public ActionResult Mem_Edit(int id=0)
        {
            var m = new mem();
```

```
    if (id == 0) {
        ViewBag.Title = "添加内存";
        var g = db.mem.Count();
        m = new mem { mem_id = g + 1 };
    }else{
        ViewBag.Title = "修改内存";
        m = db.mem.Find(id);
    }
    ViewBag.mem_brand_id = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
    m.mem_brand_id);
    return View(m);
}

// DoEdit
[HttpPost]
public ActionResult Mem_Edit(mem m)
{
    if (m.mem_id > db.mem.Count()){ //添加
        db.mem.Add(m);
        db.SaveChanges();
    }else{ //编辑
        db.Entry(m).State = System.Data.Entity.EntityState.Modified;
        db.SaveChanges();
    }
    return RedirectToAction("Mem_Index");
}
// Delete
public ActionResult Mem_Delete(int id=0)
{
    if (id != 0) {
        db.mem.Remove(db.mem.Find(id));
```

```
                db.SaveChanges();}
            return RedirectToAction("Mem_Index");
        }
    }
```

3.15 电源管理模块设计

3.15.1 电源管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成;下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.15.2 电源管理模块流程图

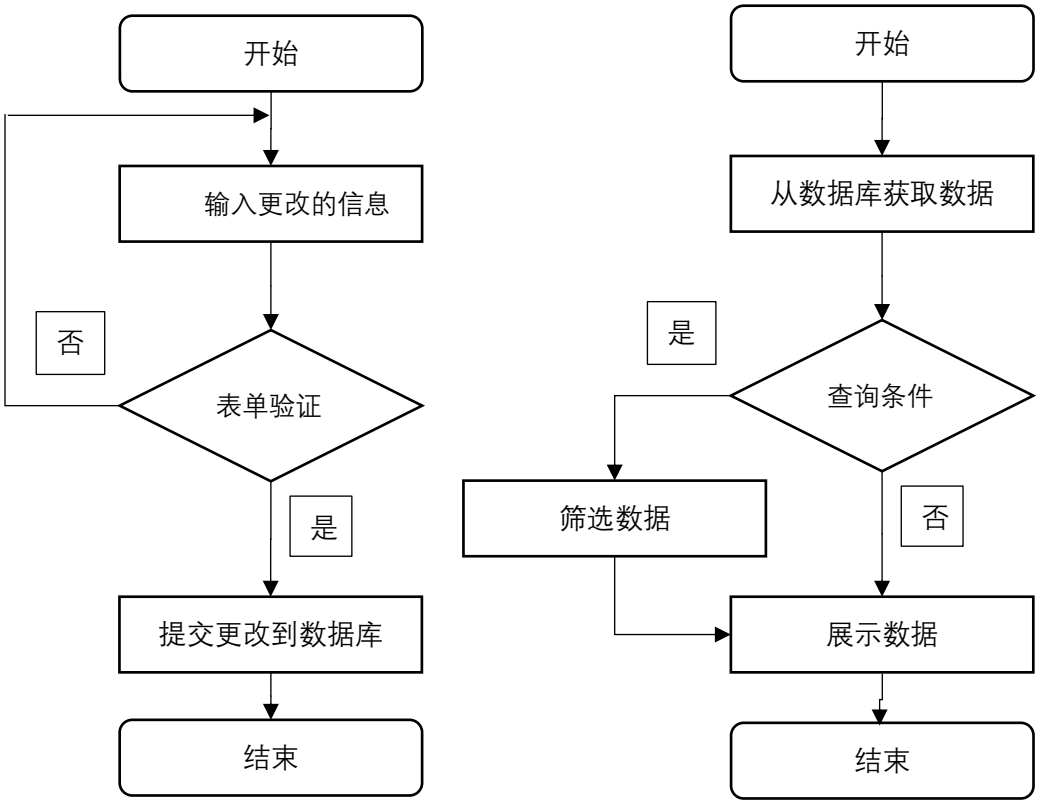

图 3.28 电源管理模块流程图

3.15.3 电源管理模块效果图

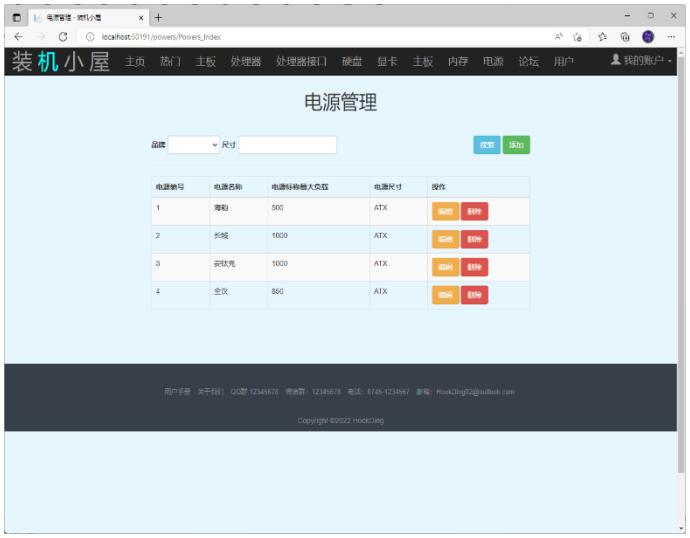

图 3.29 电源管理模块效果图

3.15.4 电源管理模块核心源代码

```
public class powersController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // Powers_Index
        public ActionResult Powers_Index(string powers_size, int powers_brands=0)
        {
            ViewBag.Title = "电源管理";
            ViewBag.Powers_size = powers_size;
            ViewBag.powers_brands = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
            powers_brands);
            var b = db.powers
                .Where(x=>string.IsNullOrEmpty(powers_size)|| x.powers_size.Contains(powers_size))
                .Where(x => powers_brands == 0 || x.powers_brands == powers_brands).ToList();
            return View(b);
        }

        // Edit
        public ActionResult Powers_Edit(int id=0)
```

```
{
    var m = new powers();
    if (id == 0) {//添加
        ViewBag.Title = "添加电源";
        var g = db.mem.Count();
        m = new powers { powers_id = g + 1 };
    }else{ //编辑
        ViewBag.Title = "修改电源";
        m = db.powers.Find(id);
    }
    ViewBag.powers_brands = new SelectList(db.brands.ToList(), "brand_id", "brand_name",
    m.powers_brands);
    return View(m);
}
// DoEdit
[HttpPost]
public ActionResult Powers_Edit(powers m)
{
    if (m.powers_id > db.powers.Count()){ //添加
        db.powers.Add(m);
        db.SaveChanges();
    }else{ //编辑
        db.Entry(m).State = System.Data.Entity.EntityState.Modified;
        db.SaveChanges();
    }
    return RedirectToAction("Powers_Index");
}
// Delete
public ActionResult Powers_Delete(int id=0)
{
    if (id != 0) {
```

```
            db.powers.Remove(db.powers.Find(id));
            db.SaveChanges();}
        return RedirectToAction("Powers_Index");
    }
}
```

3.16 论坛管理模块设计

3.16.1 论坛管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成；下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.16.2 论坛管理模块流程图

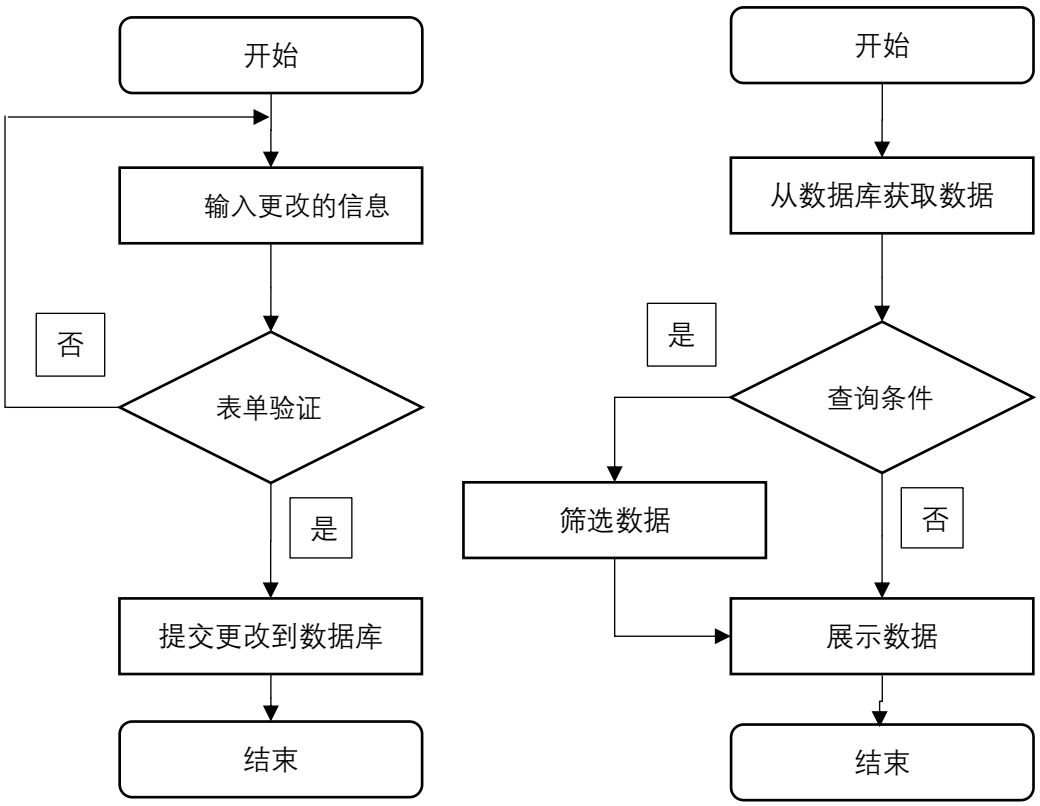

图 3.30 论坛管理模块流程图

3.16.3 论坛管理模块效果图

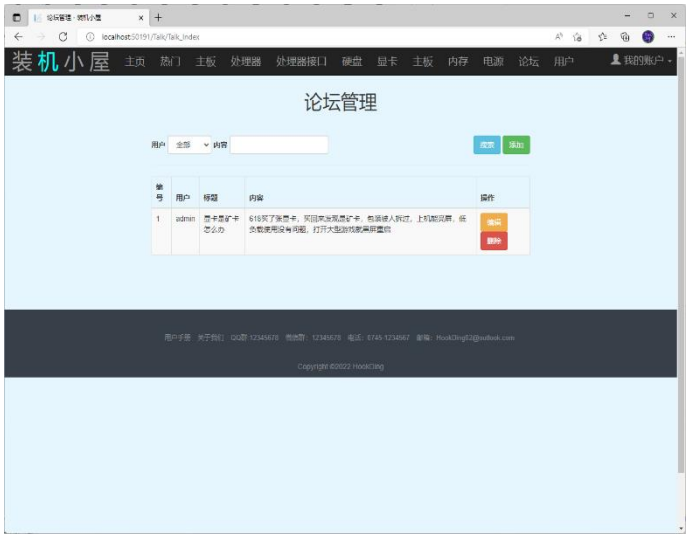

图 3.31 论坛管理模块效果图

3.16.4 论坛管理模块核心源代码

```
public class TalkController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // Talk_Index
        public ActionResult Talk_Index(string talk_content,int user_uid=0)
        {
            ViewBag.Title = "论坛管理";
            ViewBag.talk_content = talk_content;
            ViewBag.user_uid  =  new  SelectList(db.UserInfo.ToList(),  "user_uid",  "user_uname",
            user_uid);
            var b = db.Talk
            .Where(t=>string.IsNullOrEmpty(talk_content)||t.talk_content.Contains(talk_content))
             .Where(t => user_uid==0|| t.user_uid == user_uid) .ToList();
            return View(b);
        }
        // Edit
        public ActionResult Talk_Edit(int id=0)
        {
```

```
    var m = new Talk();
    if (id == 0) { //添加
        ViewBag.Title = "添加讨论";
        var g = db.Talk.Count();
        m = new Talk { talk_tid = g + 1 };
    } else{ //编辑
        ViewBag.Title = "修改讨论";
        m = db.Talk.Find(id);
    }
    ViewBag.user_uid = new SelectList(db.UserInfo.ToList(), "user_uid", "user_uname",
    m.user_uid);
    return View(m);
}
// DoEdit
[HttpPost]
public ActionResult Talk_Edit(Talk m)
{
    if (m.talk_tid > db.Talk.Count()){//添加
        db.Talk.Add(m);
        db.SaveChanges();
    }else{//编辑
        db.Entry(m).State = System.Data.Entity.EntityState.Modified;
        db.SaveChanges();
    }
    return RedirectToAction("Talk_Index");
}
// Delete
public ActionResult Talk_Delete(int id=0)
{
    if (id != 0) {
        db.Talk.Remove(db.Talk.Find(id));
```

```
            db.SaveChanges();}
        return RedirectToAction("Talk_Index");
    }
}
```

3.17 用户管理模块设计

3.17.1 用户管理模块功能介绍

这个页面主要是管理员使用的页面，需要管理员权限，该页面分为上中下三个部分，上部分是导航栏,向管理员提供管理选择,中间部分主要内容,由一个表格和一些导航按钮组成;下部分是尾部导航，尾部由一些地址信息和联系方式组成。界面简介美观，让人看上去非常舒适

3.17.2 用户管理模块流程图

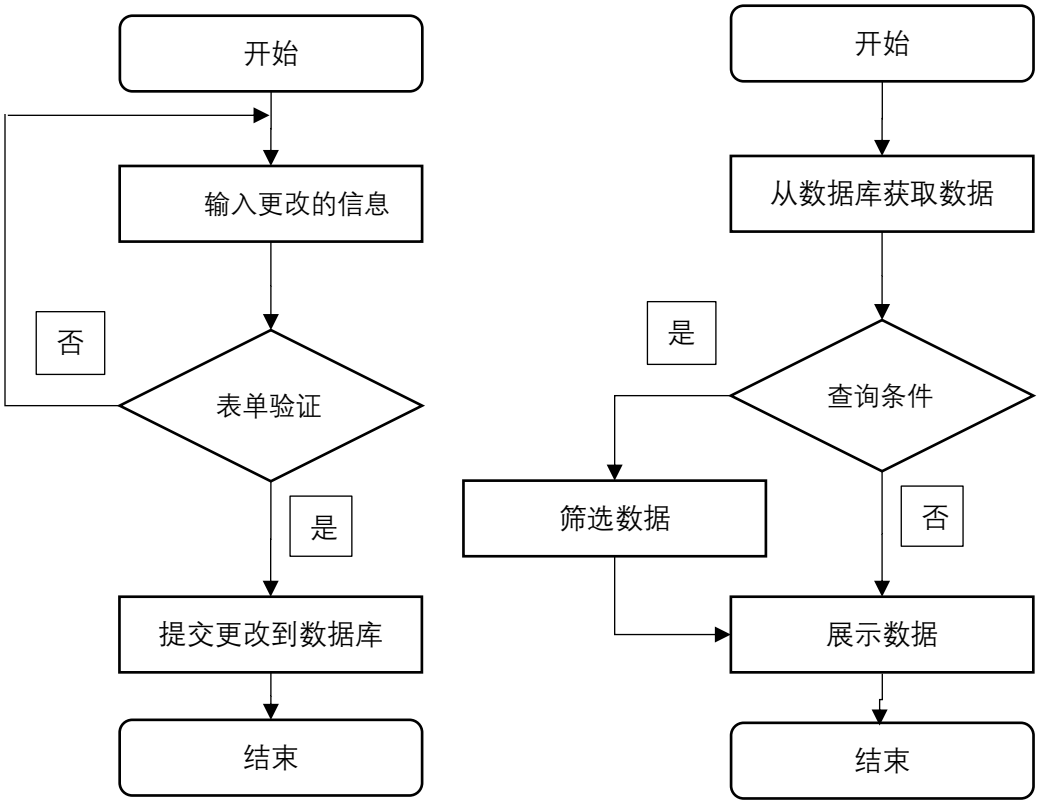

图 3.32 用户管理模块流程图

3.17.3 用户管理模块效果图

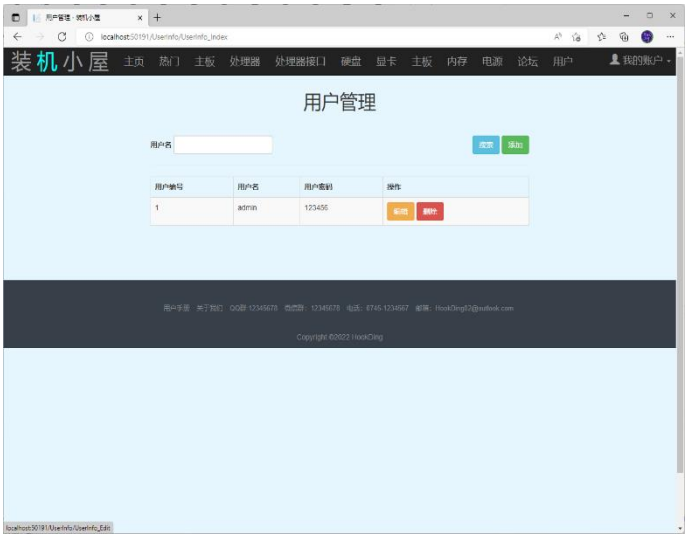

图 3.33 用户管理模块效果图

3.17.4 用户管理模块核心源代码

```
public class UserInfoController : Controller
    {
        ProjectDBEntities db = new ProjectDBEntities();
        // UserInfo_Index
        public ActionResult UserInfo_Index(string user_uname)
        {
            ViewBag.Title = "用户管理";
            ViewBag.user_uname = user_uname;
            var b = db.UserInfo
                .Where(t => string.IsNullOrEmpty(user_uname) || t.user_uname.Contains(user_uname))
                .ToList();
            return View(b);
        }
        // Edit
        public ActionResult UserInfo_Edit(int id=0)
        {
            var m = new UserInfo();
            if (id == 0) {
```

```
            ViewBag.Title = "添加用户";
            var g = db.UserInfo.Count();
            m = new UserInfo { user_uid = g + 1 };
        }else{
            ViewBag.Title = "修改用户";
            m = db.UserInfo.Find(id);
        }
        return View(m);
    }
    // DoEdit
    [HttpPost]
    public ActionResult UserInfo_Edit(UserInfo m)
    {
        if (m.user_uid > db.UserInfo.Count()){
            db.UserInfo.Add(m);
            db.SaveChanges();
        }else{
            db.Entry(m).State = System.Data.Entity.EntityState.Modified;
            db.SaveChanges();
        }
        return RedirectToAction("UserInfo_Index");
    }
    // Delete
    public ActionResult UserInfo_Delete(int id=0)
    {
        if (id != 0) {
            db.UserInfo.Remove(db.UserInfo.Find(id));
            db.SaveChanges();}
        return RedirectToAction("UserInfo_Index");
    }
}
```

第四章系统集成设计测试

4.1 单元测试

测试目的：

1） 提升网站的质量

2） 保障网站的安全

3） 降低网站的开发成本

4） 降低企业风险

5） 提高用户体验感

4.1.1 登录功能测试

登录功能测试是整个系统里是最简单且最重要的环节,我们需要通过编写几个测试用例,寻找未知的错误。登录是进入系统操作的第一步，也是安全性要求比较高的一个功能，我们可以根据需求考虑对该功能的测试，具体的登陆功能测试用例如表 4.1 所示。

表 4.1 登录测试用例

用例编号	测试用例	预期结果	实际结果
1	登录页面直接点击登录按钮	提示输入账户名或密码	与预期结果一致
2	登录页面只输入账户、密码	正常登录	与预期结果一致
3	登录输入正确的账户与错误的密码	提示账户或密码错误	与预期结果一致
4	登录输入错误的账户与正确的密码	提示账户或密码错误	与预期结果一致
5	登录页面只输入账户或密码	提示用户名或密码为空	与预期结果一致

4.1.2 注册功能测试

注册功能主要是为没有账户的游客提供服务，安全性要求相对来说比较高，可以参考边界值法来设计测试用例，具体的注册功能测试用例如表 4.2 所示。

表 4.2 注册功能测试用例

用例编号	测试用例	预期结果	实际结果
1	直接点击注册按钮	提示用户名或密码为空	与预期结果一致
2	正确输入账户、密码	正常登录	与预期结果一致
3	输入重复的账户	提示账户已存在	与预期结果一致
4	输入两次不同的密码	提示两次密码不一致	与预期结果一致
5	只输单独的入账户或密码	提示用户名或密码为空	与预期结果一致

4.1.3 首页功能测试

为了确保该功能与需求一致，该模块需要对首页的轮播效果，鼠标悬停效果，选项卡切换还有单击链接之后是否能跳转到相应页面等功能进行测试。具体的首页功能测试用例如表 4.3 所示。

表 4.3 首页测试用例

用例编号	测试用例	预期结果	实际结果
1	轮播切换	正常切换对应图片	与预期结果一致
2	鼠标悬停	正常显示	与预期结果一致
3	单击选项卡选项	正常切换对应内容	与预期结果一致
4	单击含有链接的部分	正常跳转到对应页面	与预期结果一致

4.1.4 了解更多功能测试

为了确保该功能与需求一致，该模块需要对了解更多页面的图片展示效果，文字展示效果，链接跳转效果进行测试。具体的功能测试用例如表 4.4 所示。

表 4.4 了解更多测试用例

用例编号	测试用例	预期结果	实际结果
1	图片展示	正常展示对应图片	与预期结果一致
2	文字展示	正常显示	与预期结果一致
4	单击含有链接的部分	正常跳转到对应页面	与预期结果一致

4.1.5 电脑硬件论坛功能测试

为了确保该功能与需求一致，该模块需要对电脑硬件论坛页面的评论内容展示效果，评论发布效果，单击链接之后是否能跳转到相应页面等功能进行测试。具体的功能测试用例如表 4.5 所示。

表 4.5 电脑硬件论坛测试用例

用例编号	测试用例	预期结果	实际结果
1	评论展示	正常展示所有信息	与预期结果一致
2	评论发布之后	页面刷新，展示新增内容	与预期结果一致
4	单击含有链接的部分	正常跳转到对应页面	与预期结果一致

4.1.6 管理首页功能测试

为了确保该功能与需求一致，该模块需要对管理首页的文字展示效果，导航栏跳转效果，链接跳转效果等功能进行测试，具体的功能测试用例如表 4.6 所示。

表 4.6 管理首页测试用例

用例编号	测试用例	预期结果	实际结果
1	轮播切换	正常切换对应图片	与预期结果一致

2	鼠标悬停	正常显示	与预期结果一致
3	单击选项卡选项	正常切换对应内容	与预期结果一致
4	单击含有链接的部分	正常跳转到对应页面	与预期结果一致

4.1.7 热门管理功能测试

为了确保该功能与需求一致，该模块需要对热门管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.7 所示。

表 4.7 热门管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.8 主板管理功能测试

为了确保该功能与需求一致，该模块需要对主板管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.8 所示。

表 4.8 主板管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.9 处理器管理功能测试

为了确保该功能与需求一致，该模块需要对处理器管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.9 所示。

表 4.9 处理器管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.10 处理器接口管理功能测试

为了确保该功能与需求一致，该模块需要对处理器接口管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.10 所示。

表 4.10 处理器接口管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.11 硬盘管理功能测试

为了确保该功能与需求一致，该模块需要对硬盘管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.11 所示。

表 4.11 硬盘管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.12 显卡管理功能测试

为了确保该功能与需求一致，该模块需要对显卡管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.12 所示。

表 4.12 显卡管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.13 内存管理功能测试

为了确保该功能与需求一致，该模块需要对内存管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.13 所示。

表 4.13 内存管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.14 电源管理功能测试

为了确保该功能与需求一致，该模块需要对电源管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.14 所示。

表 4.14 电源管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.15 论坛管理功能测试

为了确保该功能与需求一致，该模块需要对论坛管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.15 所示。

表 4.15 论坛管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.1.16 用户管理功能测试

为了确保该功能与需求一致，该模块需要对用户管理页面的文字展示效果，信息搜索效果，添加、编辑、删除效果等功能进行测试。具体的首页功能测试用例如表 4.16 所示。

表 4.16 用户管理测试用例

用例编号	测试用例	预期结果	实际结果
1	文字展示效果	正常展示所有文字	与预期结果一致
2	单击搜索按钮进行搜索	正常展示相关内容	与预期结果一致
3	单击含有链接的按钮	正常跳转到相应页面	与预期结果一致
4	单击删除按钮	弹出提示，确认后删除	与预期结果一致

4.2 性能测试

本次性能测试的环境与真实运行环境基本一致，都运行在同样的硬件和网络环境中，数据库则是真实环境数据，本系统采用标准的 B/S 结构，客户端通过浏览器访问整个系统。其中具体的硬件和软件环境如表 4.9。

表 4.9 软硬件环境

环境配置	应用服务器	数据库服务器	客户端
硬件配置	CPU:AMD Ryzen 5 5600G with Radeon Graphics 3.90 GHz Memory: 16.0 GB	CPU:AMD Ryzen 5 5600G with Radeon Graphics 3.90 GHz Memory: 16.0 GB	CPU:AMD Ryzen 5 5600G with Radeon Graphics 3.90 GHz Memory: 16.0 GB
软件配置	Windows 10：Microsoft Visual Studio 2019	Windows 10 : Microsoft SQL Server Management Studio 18	Windows 10:Google Chrome
网络配置	10MLAN	10MLAN	10MLAN

网络拓扑如图 4.1 所示：

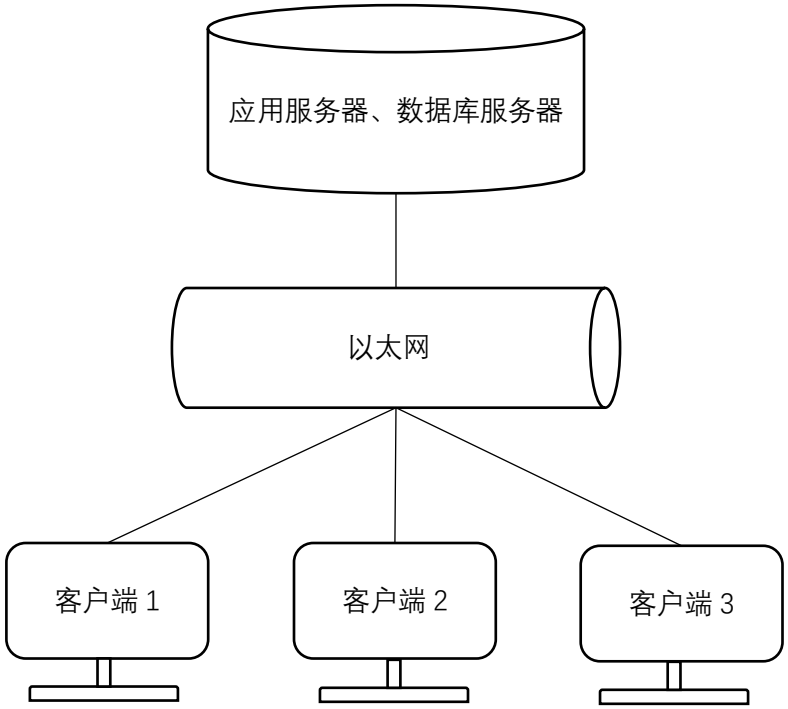

图 4.1 网络拓扑图

获取系统在正式环境下使用登录注册功能的最大平均并发用户数，也就是负载测试。系统每日的用户登录数为 900 人,每个用户登录花费时间取 9 秒,一天当中系统被使用的时间为 9 小时，估算出当前实际环境平均并发用户数为 0.3 人。测试结果需满足：用户成功率 100%；应用服务器、数据库服务器的 CPU 利用率小于百分之九十；应用服务器与数据库服务器的内存利用率小于百分之八十；平均响应时间小于 9 秒(3-5-8 原则)。我们使用登录与注册功能分别做出两个测试。

4.2.1 测试一

测试功能：登录功能

测试场景：初始 100 个用户登录,每 5 分钟增加 100 个用户登录，增加到 500 个用户登录时，开始每 10 秒退出 50 个用户。

测试目的：使用阶梯性增加用户动了的测试,能获得系统在各个负载用户下大概的性能指标。

测试结果：随着运行时间的增加,系统空闲内存呈现下降趋势，表明登录功能很可能存在内存泄露。

4.2.2 测试二

测试功能：注册功能

测试场景：初始 50 个用户开始注册,每 5 分钟增加 100 个用户注册,逐渐增加用户注册进行测试，每次测试的时间取 5 分钟，以便找出峰值用户数。

测试目的：找出峰值用户注册的数量。

测试结果：读写数据库 CPU 在 50 个用户成功运行时已达到百分之九十左右,如果想要要从硬件上调优，需要更改数据库服务器 CPU。

结论

这次毕业设计让我更加熟悉了从理论到实践的跨越。从当初的查阅图书，到现在的网站成功运行，这中间有很多值得回味的地方。

这次的设计，从选题到实现，几乎都是自己独立完成的。从前台网页设计的实现，到后台代码的编辑，我用到的软件主要有 Visual Studio 2019、Microsoft SQL Server Management Studio、HBuilderX、Word、Visio 等，开发了这个简单的在线信息网站。在系统的开发过程中，以前感觉很抽象的课程变得清晰起来，强烈地感觉到理论课程在实践中的重要性。

CAC 电脑装机小屋是现代从网络获取信息的一种途径，在互联网高速发展的今天，越来越多的人们从互联网中获取信息，互联网已和书本一样，成为了重要的信息获取来源，而本应用的设计，为想要获取电脑硬件相关信息的人们提供了一个全面的平台，在这里能够获取到大部分互联网上的零散信息，在这个应用中能够系统全面的了解到电脑的零部件的信息。

本次毕业设计项目设计模块主要是由六大模块：登录模块、注册模块、首页模块、查询模块、论坛模块、管理模块构成，从而形成一个完整的应用程序，方便用户的使用，也方便管理员对应用的管理，可以收集大量关键可靠的数据。

企业决策层分析这些数据，做出合理决策，及时调整，使之能够更好的遵循市场的销售规律，适应市场变化，从而让企业能够在激烈的行业竞争中占据一席之地。

在设计项目的过程中，因为知识点的薄弱，导致在开发时，许多功能都有不少“缺漏点”，拖延的了项目原本的进程，在今后定会不断学习，并系统性的去完善项目。

整个过程中，从需求分析到设计、编码、测试，我都力求规范化和文档化，努力让自己以前学的知识运用到本网站的开发中，尽量保证整个系统的开发进度和质量，顺利完成这次的毕业设计，为自己的大学生涯画上一个完美的句号。

最后，我相信，随着互联网技术的不断发展，CAC 电脑装机小屋的用户会越来越多，应用的功能也会越来越丰富，服务会越来越全面。

参考文献

[1] 武汉厚溥教育科技有限公司,《使用.NET 技术开发 Web 应用程序》.北京.清华大学出版社.2019:20-40
[2] 李爱玲,《Bootstrap 从入门到项目实战》.北京.清华大学出版社.2019:33-38
[3] 明日科技,《SQL Server 从入门到精通核心技术分册》.北京.清华大学出版社.2020:56-80
[4] 刘欢,《HTML5 基础知识：核心技术与前沿案例》.北京. 人民邮电出版社.2021:48-70
[5] 斯托扬,《JavaScript 面向对象编程指南（第 2 版）》.北京. 人民邮电出版社.2021:78-79

致谢

时间飞逝，大学的学习生活很快就要过去，在这几年的学习生活中，我收获了很多，而这些成绩的取得是和一直关心帮助我的人分不开的。

首先要感谢我的老师。本文是在老师的精心指导下完成的，从选题、设计直至完成的整个过程中，都得到了老师耐心细致的指导。老师严谨的治学态度、渊博的学识、独特的学术思维、一丝不苟的工作作风、热情待人的品质，使我满怀敬意。

感谢湖南铁路科技职业技术学院所有的领导和老师，这些年的学习基础对我非常重要，感谢软件技术专业中的所有老师和领导，在完成该项目中得到了你们支持和帮助。

感谢我亲爱的同学们，在学习中我们相互帮助，互相激励和关心。

感谢我的家人，这么多年来，正是你们的支持和鼓励，才使我顺利地完成学业;正是你们的关心和默默的奉献，给我创造了优越的条件，使我在学习的道路上乐观向上、勇往直前。