南京信息工程大学

《数据库系统》课程项目设计报告

题　　目　　实验室仪器设备管理系统

学生姓名　梅应宝、丁诚诚

学　　号　202383290121 202383290102

学　　院　计算机学院

专　　业　计算机科学与技术

指导教师　顾韵华

二〇二五　年　五　月　二十八　日

目 录

实验室仪器设备管理系统

梅应宝、丁诚诚

南京信息工程大学计算机学院，江苏 南京 210044

1 引言

1.1 课题概况

随着高校科研教学活动的不断发展，实验室设备的种类和数量日益增多，传统依赖人工登记和纸质管理的方式已无法满足现代管理的高效性与精准性要求。管理人员在面对设备维护、借还登记、责任划分、损耗统计等环节时，常常因信息混乱而造成设备丢失、数据滞后、责任不清等问题。为提升高校实验室设备管理的信息化水平，优化设备使用流程，本课题设计并实现了一套基于 C# 与 SQL Server 的《实验室仪器设备管理系统》。系统采用分层架构，前端界面简洁直观，后端数据库结构规范严谨，可实现设备信息、使用状态、借还记录、负责人信息的统一管理，全面提升实验室设备使用的可视性、规范性与高效性。

本系统的开发不仅具有现实应用价值，还有助于提升开发者对数据库系统结构设计、用户权限控制、WinForm 图形界面开发等方面的综合能力，具有良好的教学示范作用和实际推广意义。

1.2 课题内容

本系统采用 C# 编程语言开发客户端界面，结合 SQL Server 数据库进行后端数据管理，主要完成以下功能模块的设计与实现：

1. 用户管理：实现用户信息的注册、登录及权限管理；
2. 实验室信息管理：对实验室基本信息进行维护；
3. 仪器设备信息管理：支持设备信息的录入、修改、删除；
4. 保管员（负责人）信息管理：指定各设备的负责人；
5. 设备借还管理与状态记录：记录设备的借出、归还、维修、报废等状态；
6. 初步日志功能：记录设备操作历史，便于追踪使用记录。

本系统面向的用户主要包括系统管理员与普通用户两类。管理员具有所有模块的访问与操作权限，普通用户只能管理自己负责的设备，并执行部分功能操作，并且管理员可对普通用户的各个权限进行编辑。

2 需求分析

2.1 总体需求描述

本系统主要面向实验室管理人员（管理员）与设备负责人（普通用户）两类用户，不同角色具有不同的权限范围与功能模块。

管理员拥有系统的全部权限，可进行用户管理（包括权限管理以及账号密码管理）、实验室管理、设备登记与分配、设备状态更新、数据统计查询等操作；

普通用户则仅可登录系统，查看和管理自己负责的设备，执行设备借还登记、状态修改等操作。

系统主要实现对实验室仪器设备从采购入库到借用归还、维修报废等环节的全生命周期管理，旨在提升实验室仪器设备管理效率、规范使用流程，并便于后续的设备追溯和数据分析。

2.2 数据需求

本系统涉及的数据包括：用户信息、实验室信息、仪器设备信息、权限信息、设备日志信息等，描述如下：

用户信息包括：用户名（登录账号）、姓名、联系方式、密码（Hash 加密处理）、角色；

实验室信息包括：实验室编号、实验室名称、实验室位置；

设备信息包括：设备编号、设备名称、型号、购买日期、状态、所属实验室编号、管理人员用户名；

权限信息包括：权限编号、权限名称、用户名、授权时间；

设备操作日志信息包括：日志编号、设备编号、操作类型、操作人、操作时间、备注信息。

2.3 功能需求

系统面向的用户包括两类：系统管理员和普通用户（设备负责人）。管理员具有系统的全部管理权限，普通用户则只能操作自己负责的设备。具体功能需求如下：

系统管理：包括用户信息的添加、删除、修改与查询；设置用户角色（管理员 / 普通用户）；为普通用户分配权限，实现系统访问控制。

实验室管理：支持实验室基础信息的维护功能，包括新增实验室、修改名称与位置、删除实验室信息等操作。

设备管理：支持设备信息的登记与维护，包括添加设备、编辑设备名称与型号、指定负责人、修改设备状态、删除设备记录等；并能根据状态（如借出、维修）自动调整可用性。

设备使用状态登记：用户可对设备进行借出、归还、维修、报废等操作，系统将自动记录操作时间与操作人，以便后续追溯与统计。

查询与统计功能：系统支持按照实验室、负责人、设备状态、关键字等多条件组合进行设备信息查询；管理员可查看设备总数、状态分布、实验室占比等统计数据。

权限管理：管理员可配置权限列表，并通过用户权限表将操作权限赋予不同用户，实现细粒度的功能访问控制。

设备管理的主要规则包括：每台设备只能归属于一个实验室；每台设备只能指定一个负责人；每位用户可管理多台设备；设备状态仅限于“正常”、“借出”、“维修”、“报废”四类；所有状态操作均应记录在设备日志中以备审计。

3 数据库设计与实现

3.1 概念结构设计

3.1.1 系统中的实体

（1）用户实体（UserInfo）

用户实体如图 3-1 所示，包含的属性：用户名、用户姓名、联系方式、密码、角色；主码是用户名。

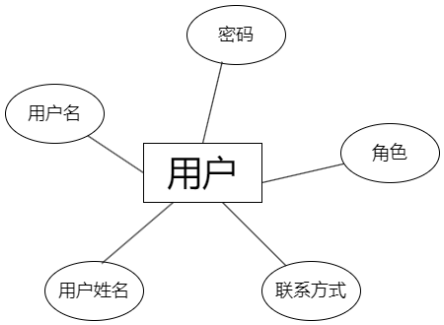


图 3-1 用户实体属性图

（2）实验室实体（Lab）

实验室实体如图 3-2 所示，包含的属性为：实验室编号、实验室名称、实验室位置；主码是实验室编号。

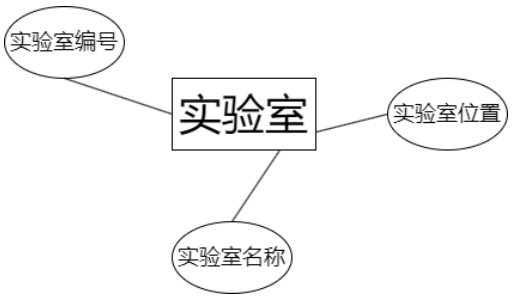


图 3-2 实验室实体属性图

（3）设备实体（Device）

设备实体如图 3-3 所示，包含的属性为：设备编号、设备名称、型号、购买日期、状态、所属实验室编号、管理人员用户名；主码是设备编号，其中实验室编号与负责人用户名为外键，分别关联实验室表与用户表。

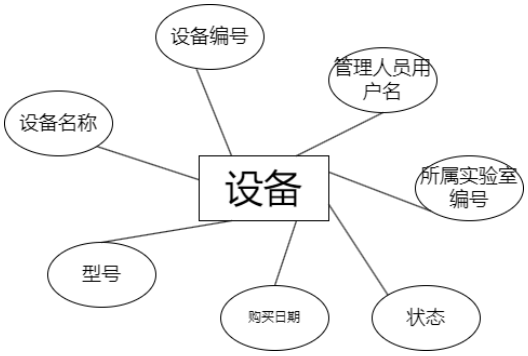


图 3-3 设备实体属性图

（4）权限实体（Permission 与 UserPermission）

权限信息实体如图 3-4 所示，其中：

Permission 表包含属性：权限编号、权限名称，主码是权限编号；

UserPermission 表包含属性：用户名、权限编号、授权时间；主码为（用户名，权限编号）联合主键，两个字段均为外键，分别关联 UserInfo 与 Permission。

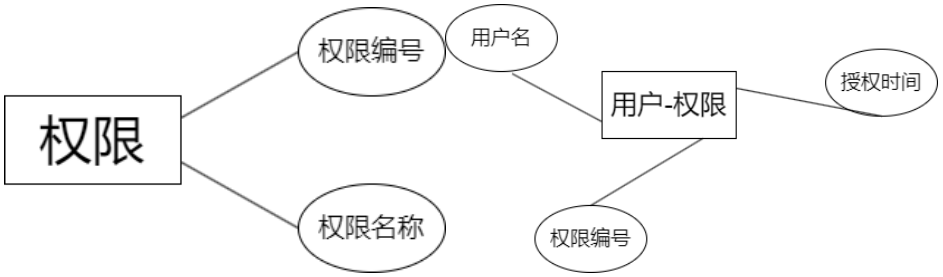


图 3-4 权限实体属性图

（5）设备日志实体（DeviceLog）

设备日志实体如图 3-5 所示，包含的属性为：日志编号、设备编号、操作类型、操作人用户名、操作时间、备注信息；主码是日志编号，设备编号为外键，关联设备表。

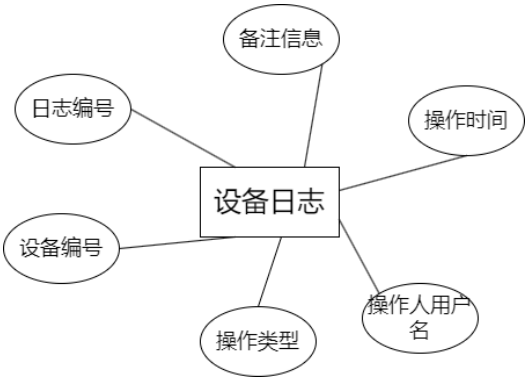
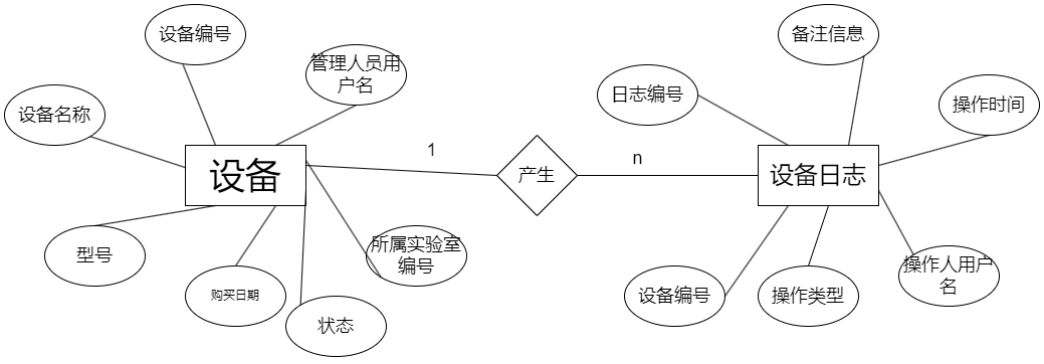
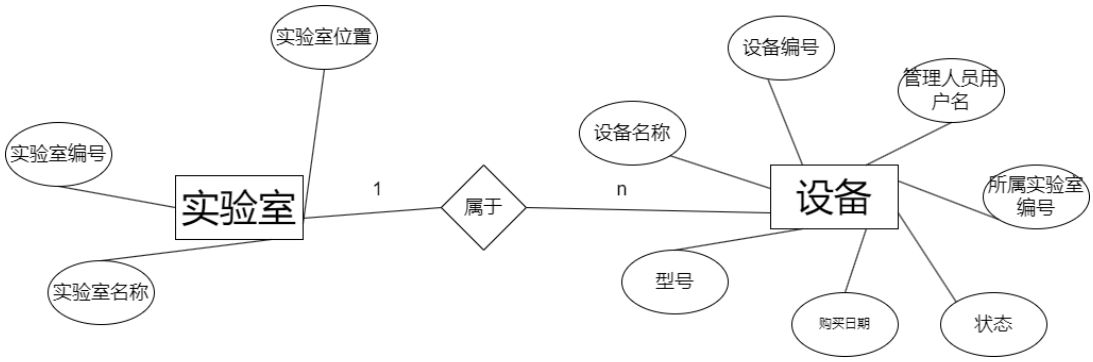


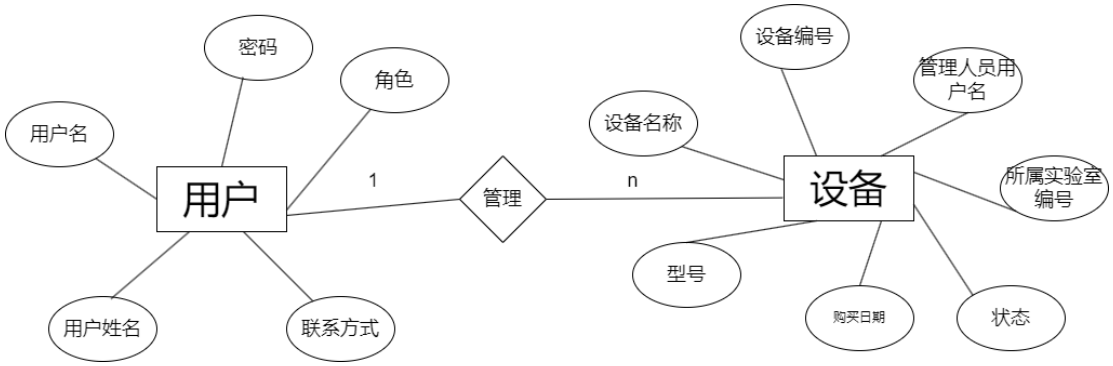
图 3-5 设备日志实体属性图

3.1.2 系统 ER 图

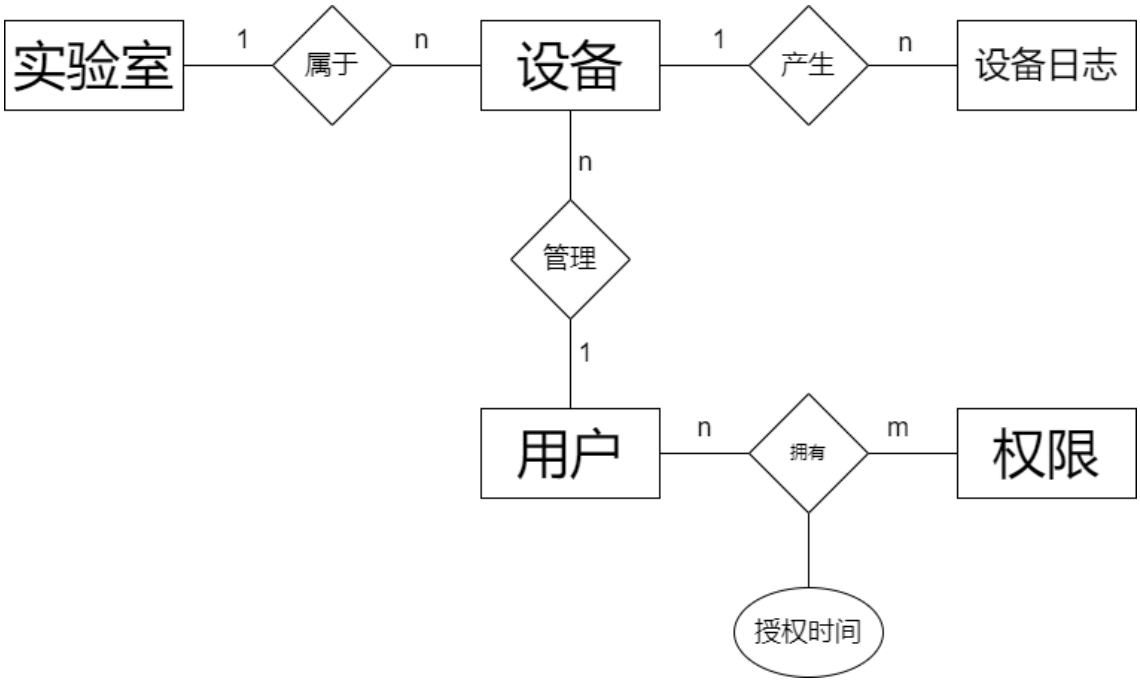
（1）分 ER 图







（2）总 ER 图



3.2 逻辑结构设计

（1）UserInfo 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
UserName	nvachar（50）	否	系统登陆用户名	主码
User_name	nvachar（50）	否	用户姓名	
Contact	nvachar（50）	可	联系方式（手机号/邮箱）	
PasswordHash	Varbinary(64)	否	用户密码哈希值	
Role	nvachar（50）	否	用户角色	只能取“管理员”或者“普通用户”

（2）Lab 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
LabID	int	否	实验室编号	主码
LabName	nvachar（100）	否	实验室名称	
Location	nvachar（100）	可	实验室位置	

（3）Device 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
DeviceID	int	否	设备编号	主码
DeviceName	nvachar（100）	否	设备名称	
Model	nvachar（50）	可	设备型号	
PurchaseDate	Date	可	购买日期	
Status	nvachar（20）	否	设备状态	check 约束(正常/借出/维修/报废)
LabID	Int	否	所属实验室编号	外码->Lab(LabID)
ManagerID	nvachar（50）	否	设备负责人用户名	外码->UserInfo(UserName)

（4）Permission 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
PermissionID	int	否	权限编号	主码
PermissionName	nvachar（50）	否	权限名称	唯一约束

（5）UserPermission 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
UserName	nvachar（50）	否	用户名	主码，外码->UserInfo
PermissionID	int	否	权限编号	主码，外码->Permission
GrantTime	datetime	可	授权时间	默认值为当前时间

（6）DeviceLog 表

列　　名	数 据 类 型	是否可取空值	含　　义	说　　明
LogID	int	否	日志编号	主码
DeviceID	int	否	设备名称	主码->Device
Action	nvachar（50）	否	操作类型	
Operator	nvachar（50）	否	操作者用户名	
ActionDate	datetime	否	操作时间	默认值为当前时间
Note	nvachar（255）	可	备注信息	

3.3 物理设计与实施

数据库的物理结构采用系统默认文件结构，采用系统自动建立的索引。利用界面方式或 CREATE 语句创建数据库和各个基本表，输入测试样例数据，这样便完成了数据库的物理设计和实施。

（1）UserInfo 表

	UserName	User_name	Contact	PasswordHash	Role
▶	dcc	丁诚诚	2122854828@qq.com	<二进制数据>	管理员
	li	李老师	12345678911	<二进制数据>	普通用户
	liu	刘老师	12345678911	<二进制数据>	普通用户
	meiyingbao	梅应宝	13914993851	<二进制数据>	管理员
	testuser	test	1243123	<二进制数据>	普通用户
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

（2）Lab 表

	LabID	LabName	Location
▶	1	化学实验室	教学楼 A-304
	2	物理实验室	教学楼 A-304
	9	test1	test1
*	*NULL*	*NULL*	*NULL*

（3）Device 表

	DeviceID	DeviceName	Model	PurchaseDate	Status	LabID	ManagerID
▶	2	光谱分析仪	SP-2001	2022-09-01	正常	1	li
	3	显微镜	SP-2021	2024-08-21	正常	2	liu
	8	光刻机	SF-10012	2025-05-26	正常	2	li
	11	12312	21312	2025-05-28	借出	9	liu
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

（4）Permission 表

	PermissionID	PermissionName
▶	1	查看设备
	4	管理用户
	3	删除设备
	2	添加设备
*	*NULL*	*NULL*

（5）UserPermission 表

	UserName	PermissionID	GrantTime
▶	LI	1	2025-05-26 14:06:16.337
	LI	2	2025-05-26 14:06:16.337
	LI	3	2025-05-26 14:06:16.337
	testuser	1	2025-05-26 22:11:00.787
	testuser	2	2025-05-26 22:11:00.787
*	*NULL*	*NULL*	*NULL*

（6）DeviceLog 表

LogID	DeviceID	Action	Operator	ActionDate	Note
1	9	删除设备	li	2025-05-26 14:14:16.210	删除设备：213421
2	10	创建设备	meiyingbao	2025-05-26 22:10:01.533	通过编辑页面自动记录
3	10	修改设备...	meiyingbao	2025-05-26 22:10:13.670	通过编辑页面自动记录
4	10	删除设备	meiyingbao	2025-05-26 22:10:19.067	删除设备：21412
5	11	创建设备	meiyingbao	2025-05-28 16:26:16.203	通过编辑页面自动记录
NULL	NULL	NULL	NULL	NULL	NULL

4 系统设计与实现

4.1 系统总体功能

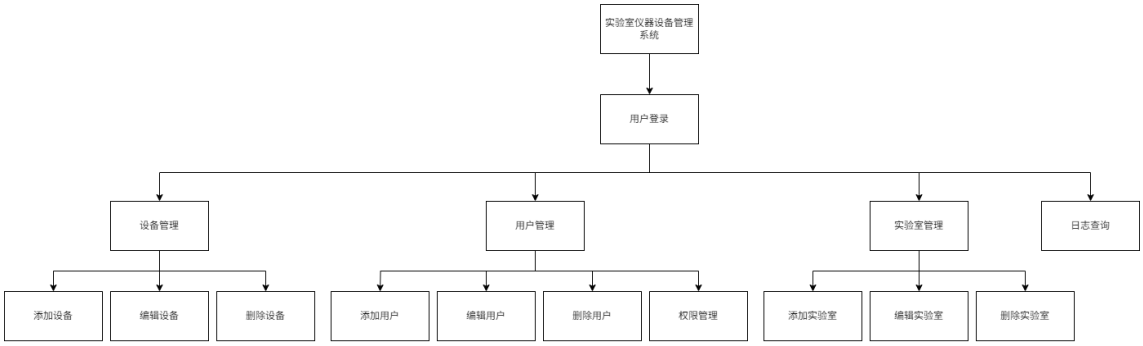


图 3-1 系统总体功能结构图

由于订单涉及多个实体，所以将它的录入与维护分在两个模块中设计。此外，设计用户登录功能，用户使用本系统需要先登录，通过系统验证后，才能进入系统。

下面给出每个功能模块的功能描述。

- 用户登录：对用户输入的用户名与密码进行验证，判断其角色为“管理员”或“普通用户”。
- 用户管理：用于管理员对用户信息的添加、修改、删除操作，包括姓名、联系方式、角色等。
- 实验室管理：管理员可对实验室的基本信息进行维护，包括实验室编号、名称与位置。
- 设备管理：实现对设备信息的增加、修改、删除操作。普通用户仅能维护自己负责的设备。
- 权限管理：管理员为用户分配功能权限，例如“添加设备”、“删除用户”等。
- 状态登记：对设备的借出、归还、维修与报废操作进行登记，并自动写入日志表中。
- 日志查询：展示所有设备的操作记录信息，支持按条件筛选。

4.2 系统开发技术

类别	技术 / 工具名称
编程语言	C#
开发环境	Visual Studio 2022
系统平台	Windows 10 / 11
数据库系统	SQL Server 2019
数据访问方式	ADO.NET
图形界面框架	Windows Forms（WinForm）
密码加密方式	HASHBYTES(SHA-256)
数据结构组织	多表设计 + 外键关联 + 日志

4.3 各模块详细设计与实现

4.3.1 用户登录模块设计与实现

定义了一个 CurrentUser 类，用来存储当前用户信息，为之后权限判断提供条件。

```
21 个引用
class CurrentUser
{
    8 个引用
    public static string UserName { get; set; }
    8 个引用
    public static string Role { get; set; }
    5 个引用
    public static List<string> Permissions { get; set; } = new List<string>();
}
```

图 4-3-1-1 CurrentUser 类

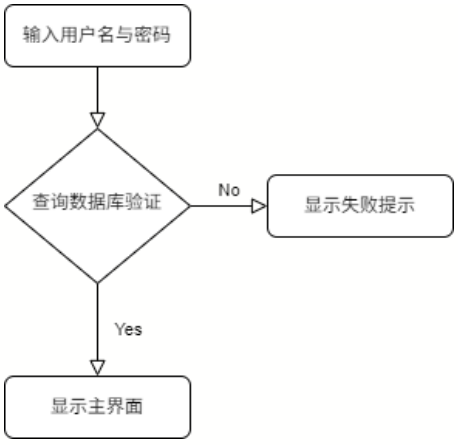


图 4-3-1-2 用户登录模块流程图

图 4-3-1-3 用户登录功能实现界面

用户登录功能的核心代码如下：

```
1 个引用
private void button1_Click(object sender, EventArgs e)
{
    string username = txtUsername.Text.Trim();
    string password = txtPassword.Text.Trim();

    if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
    {
        lblMessage.Text = "请输入用户名和密码";
        return;
    }

    string connStr = "Data Source=localhost;Initial Catalog=LabDeviceManagement;Integrated Security=True;";
    string sql = @"
        SELECT Role FROM UserInfo
        WHERE UserName = @username
        AND PasswordHash = HASHBYTES('SHA2_256', @password)";

    using (SqlConnection conn = new SqlConnection(connStr))
    using (SqlCommand cmd = new SqlCommand(sql, conn))
    {
        cmd.Parameters.AddWithValue("@username", username);
        cmd.Parameters.Add("@password", SqlDbType.VarChar, 64).Value = password;

        try
        {
            conn.Open();
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    string role = reader.GetString(0);

                    // 设置当前用户信息（示例 CurrentUser）
                    CurrentUser.UserName = username;
                    CurrentUser.Role = role;

                    if (role == "管理员")
                    {
                        CurrentUser.Permissions = new List<string> { "查看设备", "添加设备", "删除设备", "管理用户" };
                    }
                    else
                    {
                        CurrentUser.Permissions = LoadUserPermissionsFromDatabase(username);
                    }

                    DialogResult = DialogResult.OK;
                    this.Close();
                }
                else
                {
                    lblMessage.Text = "用户名或密码错误";
                }
            }
        }
        catch (Exception ex)
        {
            lblMessage.Text = "数据库连接失败：" + ex.Message;
        }
    }
}
```

图 4-3-1-4 用户登录功能核心代码

4.3.2　主界面设计与实现

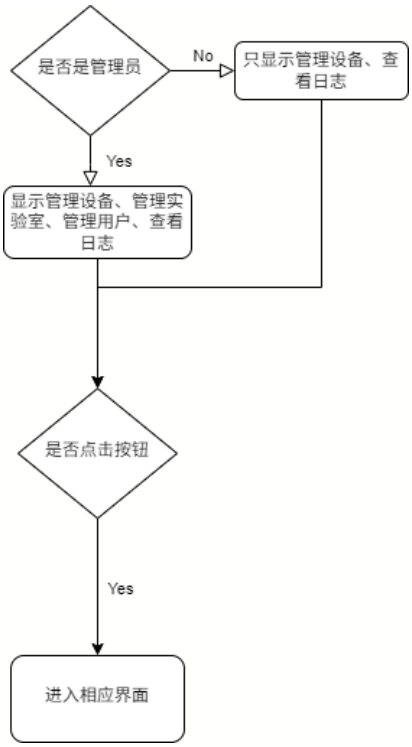


图 4-3-2-1 主界面模块流程图

图 4-3-2-2 管理员主界面实现界面

图 4-3-2-3 普通用户主界面实现界面

主界面的核心代码如下：

```
1 个引用
public MainForm()
{
    InitializeComponent();

    this.Text = $"欢迎 {CurrentUser.UserName} 使用实验室仪器管理系统 - 当前角色：{CurrentUser.Role}";

    // 权限控制
    btnUserManager.Visible = CurrentUser.Role == "管理员";
    btnLab.Visible = CurrentUser.Role == "管理员";
}

1 个引用
private void btnDeviceManager_Click(object sender, EventArgs e)
{
    DeviceManagementForm form = new DeviceManagementForm();
    form.ShowDialog();
}

1 个引用
private void btnUserManager_Click(object sender, EventArgs e)
{
    UserManagementForm form = new UserManagementForm();
    form.ShowDialog();
}

1 个引用
private void btnViewLog_Click(object sender, EventArgs e)
{
    DeviceLogForm form = new DeviceLogForm();
    form.ShowDialog();
}

1 个引用
private void btnLab_Click(object sender, EventArgs e)
{
    EditLabForm form = new EditLabForm();
    form.ShowDialog();
}
```

图 4-3-2-4 主界面核心代码

4.3.3 实验室管理设计与实现

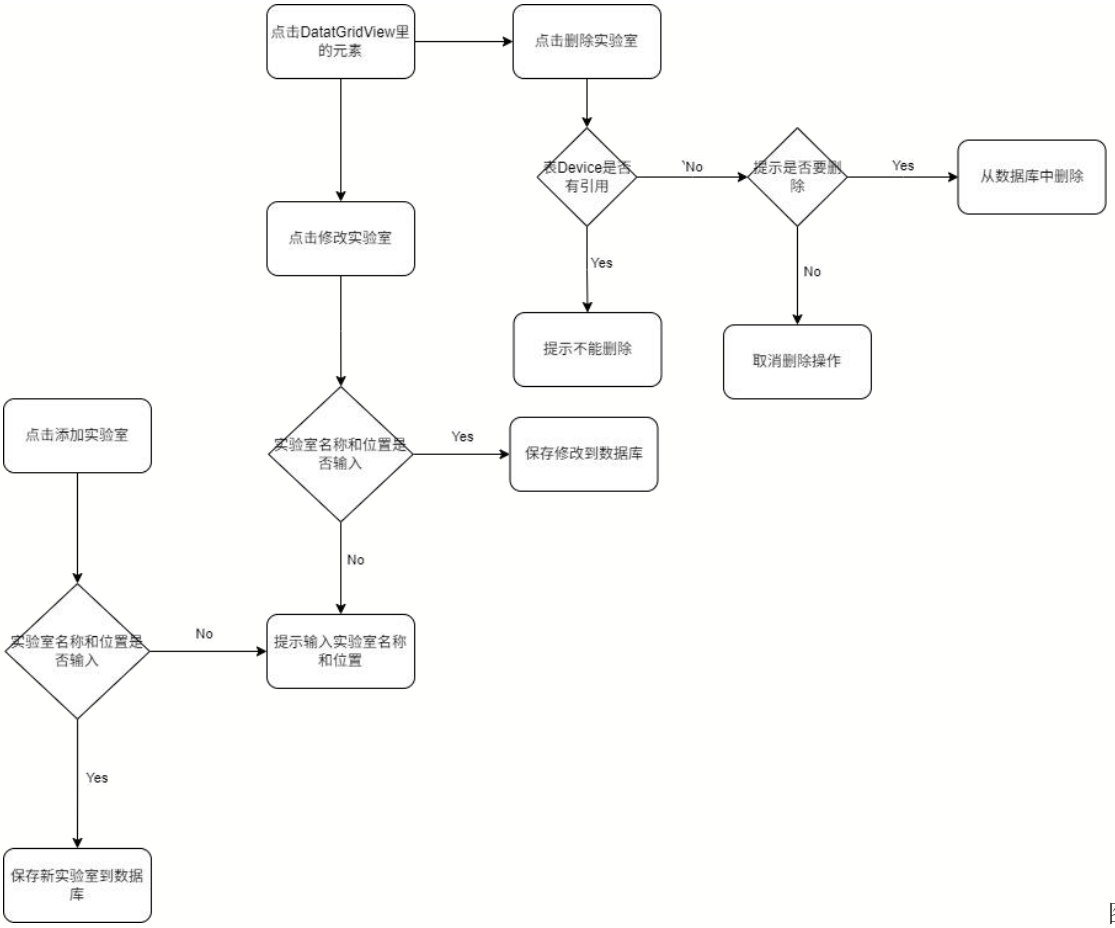


图 4-3-3-1 实验室管理流程图

图 4-3-3-2 实验室管理实现界面

实验室管理的核心代码如下：

```
private void btnAdd_Click(object sender, EventArgs e)
{
    string name = txtLabName.Text.Trim();
    string location = txtLabLocation.Text.Trim();

    if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(location))
    {
        MessageBox.Show("实验室名称和位置都不能为空！", "输入错误", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        string sql = "INSERT INTO Lab (LabName, Location) VALUES (@name, @location)";
        using (SqlCommand cmd = new SqlCommand(sql, conn))
        {
            cmd.Parameters.AddWithValue("@name", name);
            cmd.Parameters.AddWithValue("@location", location);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }

    LoadLabs();
    txtLabName.Clear();
    txtLabLocation.Clear();
}
```

```
1 个引用
private void btnEdit_Click(object sender, EventArgs e)
{
    if (dgvLab.CurrentRow == null) return;

    int labId = Convert.ToInt32(dgvLab.CurrentRow.Cells["LabID"].Value);
    string newName = txtLabName.Text.Trim();
    string newLoc = txtLabLocation.Text.Trim();

    if (string.IsNullOrEmpty(newName) || string.IsNullOrEmpty(newLoc))
    {
        MessageBox.Show("实验室名称和位置都不能为空！", "输入错误", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        string sql = "UPDATE Lab SET LabName=@name, Location=@loc WHERE LabID=@id";
        using (SqlCommand cmd = new SqlCommand(sql, conn))
        {
            cmd.Parameters.AddWithValue("@name", newName);
            cmd.Parameters.AddWithValue("@loc", newLoc);
            cmd.Parameters.AddWithValue("@id", labId);
            conn.Open();
            cmd.ExecuteNonQuery();
        }
    }

    LoadLabs();
}
```

```
1 个引用
private void btnDelete_Click(object sender, EventArgs e)
{
    if (dgvLab.CurrentRow == null) return;

    int labId = Convert.ToInt32(dgvLab.CurrentRow.Cells["LabID"].Value);
    string name = dgvLab.CurrentRow.Cells["LabName"].Value.ToString();

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        conn.Open();

        // 1  检查是否有设备引用该实验室
        string checkSql = "SELECT COUNT(*) FROM Device WHERE LabID = @id";
        using (SqlCommand checkCmd = new SqlCommand(checkSql, conn))
        {
            checkCmd.Parameters.AddWithValue("@id", labId);
            int count = (int)checkCmd.ExecuteScalar();

            if (count > 0)
            {
                MessageBox.Show($"该实验室【{name}】已被 {count} 个设备引用，无法删除。", "删除失败", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
        }

        // 2  无引用，确认删除
        DialogResult result = MessageBox.Show($"确认要删除实验室【{name}】吗？", "确认删除", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            string deleteSql = "DELETE FROM Lab WHERE LabID = @id";
            using (SqlCommand deleteCmd = new SqlCommand(deleteSql, conn))
            {
                deleteCmd.Parameters.AddWithValue("@id", labId);
                deleteCmd.ExecuteNonQuery();
            }

            LoadLabs();
        }
    }
}
```

```
1 个引用
private void dgvLab_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (dgvLab.CurrentRow != null)
    {
        txtLabName.Text = dgvLab.CurrentRow.Cells["LabName"].Value.ToString();
        txtLabLocation.Text = dgvLab.CurrentRow.Cells["Location"].Value?.ToString();
    }
}
```

图 4-3-3-3 实验室管理核心代码

4.3.4　设备管理设计与实现

数据库预览表格，如果用户是管理员将会显示所有设备信息，如果是普通用户则只会显示由他管理的设备的信息，并且添加设备的时候默认管理员为自己

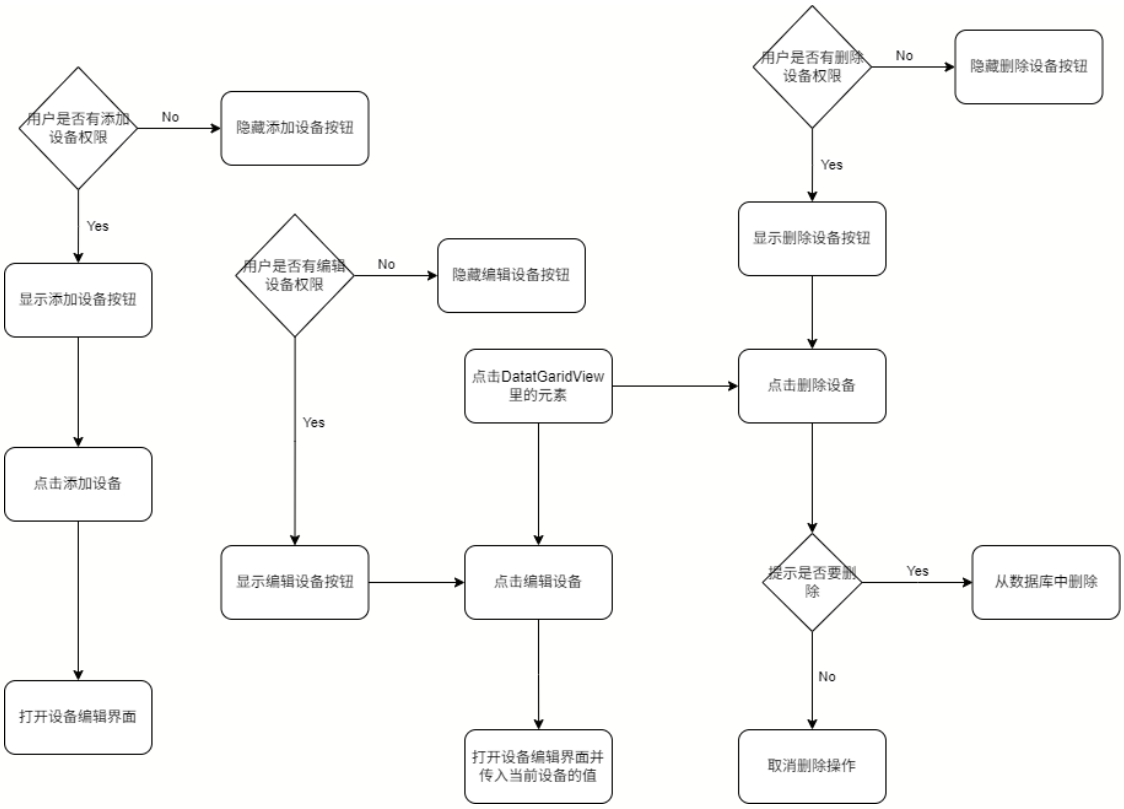


图 4-3-4-1 设备管理流程图

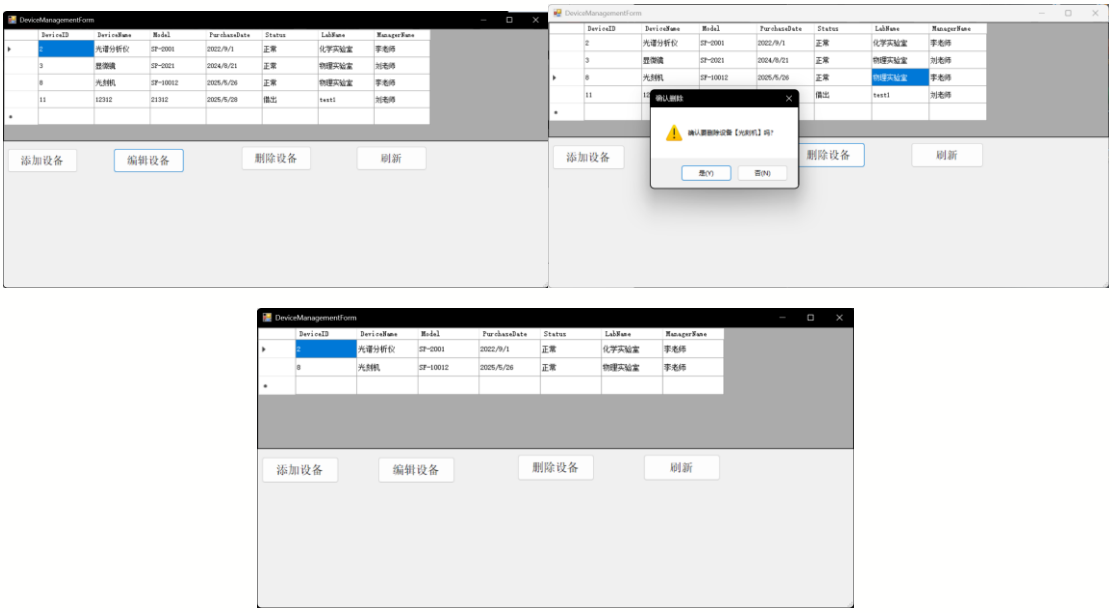


图 4-3-4-2 设备管理实现界面

设备管理的核心代码如下：

```
1 个引用
private void ApplyPermissions()
{
    btnAdd.Enabled = CurrentUser.Permissions.Contains("添加设备");
    btnEdit.Enabled = CurrentUser.Permissions.Contains("添加设备");
    btnDelete.Enabled = CurrentUser.Permissions.Contains("删除设备");
}
```

```
private void btnDelete_Click_1(object sender, EventArgs e)
{
    if (dgvDevices.CurrentRow != null)
    {
        int deviceId = Convert.ToInt32(dgvDevices.CurrentRow.Cells["DeviceID"].Value);
        string deviceName = dgvDevices.CurrentRow.Cells["DeviceName"].Value.ToString();

        DialogResult result = MessageBox.Show(
            $"确认要删除设备【{deviceName}】吗？",
            "确认删除",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);

        if (result == DialogResult.Yes)
        {
            using (SqlConnection conn = new SqlConnection(connStr))
            {
                conn.Open();

                // 1  删除设备
                string sql = "DELETE FROM Device WHERE DeviceID = @id";
                using (SqlCommand cmd = new SqlCommand(sql, conn))
                {
                    cmd.Parameters.AddWithValue("@id", deviceId);
                    cmd.ExecuteNonQuery();
                }

                // 2  写入日志
                string logSql = @"INSERT INTO DeviceLog (DeviceID, Action, Operator, ActionDate, Note)
                                  VALUES (@deviceId, @action, @operator, @date, @note)";
                using (SqlCommand logCmd = new SqlCommand(logSql, conn))
                {
                    logCmd.Parameters.AddWithValue("@deviceId", deviceId);
                    logCmd.Parameters.AddWithValue("@action", "删除设备");
                    logCmd.Parameters.AddWithValue("@operator", CurrentUser.UserName);
                    logCmd.Parameters.AddWithValue("@date", DateTime.Now);
                    logCmd.Parameters.AddWithValue("@note", $"删除设备：{deviceName}");
                    logCmd.ExecuteNonQuery();
                }

                LoadDeviceList();
            }
        }
    }
}
```

```
5 个引用
private void LoadDeviceList()
{
    string sql;

    if (CurrentUser.Role == "管理员")
    {
        sql = @"
    SELECT D.DeviceID, D.DeviceName, D.Model, D.PurchaseDate, D.Status,
           L.LabName, U.User_name AS ManagerName
    FROM Device D
    JOIN Lab L ON D.LabID = L.LabID
    JOIN UserInfo U ON D.ManagerID = U.UserName";
    }
    else
    {
        sql = @"
    SELECT D.DeviceID, D.DeviceName, D.Model, D.PurchaseDate, D.Status,
           L.LabName, U.User_name AS ManagerName
    FROM Device D
    JOIN Lab L ON D.LabID = L.LabID
    JOIN UserInfo U ON D.ManagerID = U.UserName
    WHERE D.ManagerID = @managerId";
    }

    using (SqlConnection conn = new SqlConnection(connStr))
    using (SqlCommand cmd = new SqlCommand(sql, conn))
    {
        if (CurrentUser.Role != "管理员")
        {
            cmd.Parameters.AddWithValue("@managerId", CurrentUser.UserName);
        }

        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        DataTable table = new DataTable();
        adapter.Fill(table);
        dgvDevices.DataSource = table;
    }
}
```

图 4-3-4-3 设备管理核心代码

4.3.5 设备编辑界面设计与实现

这个界面会根据设备管理界面点击的是添加设备还是编辑设备而不引用或引用相关设备信息，而且会根据当前用户而决定是否锁定设备管理员一栏。

实验室选择和管理员选择采用下拉框的形式，通过读取数据库数据来显示下拉框内容。

在设备编辑界面，在实验室选择和管理员选择的下拉框中还可以添加相应信息，由于普通用户的管理员选择一栏是锁定的，所以并不会发生权限冲突。

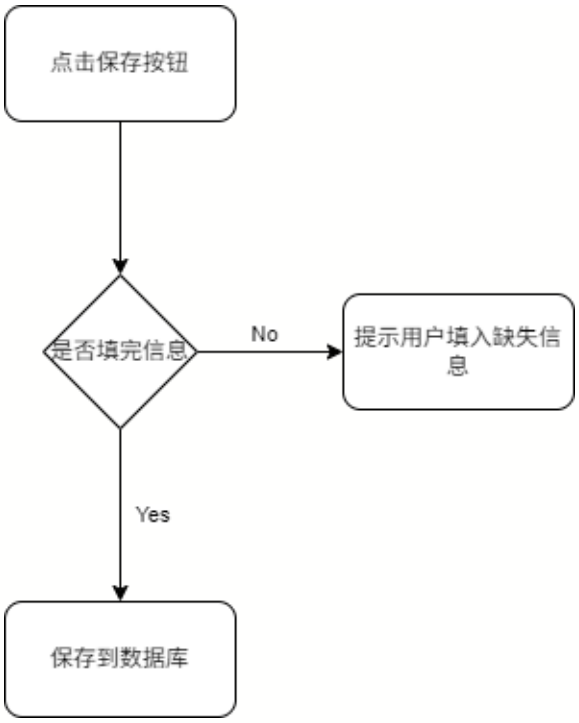


图 4-3-5-1 设备编辑界面流程图

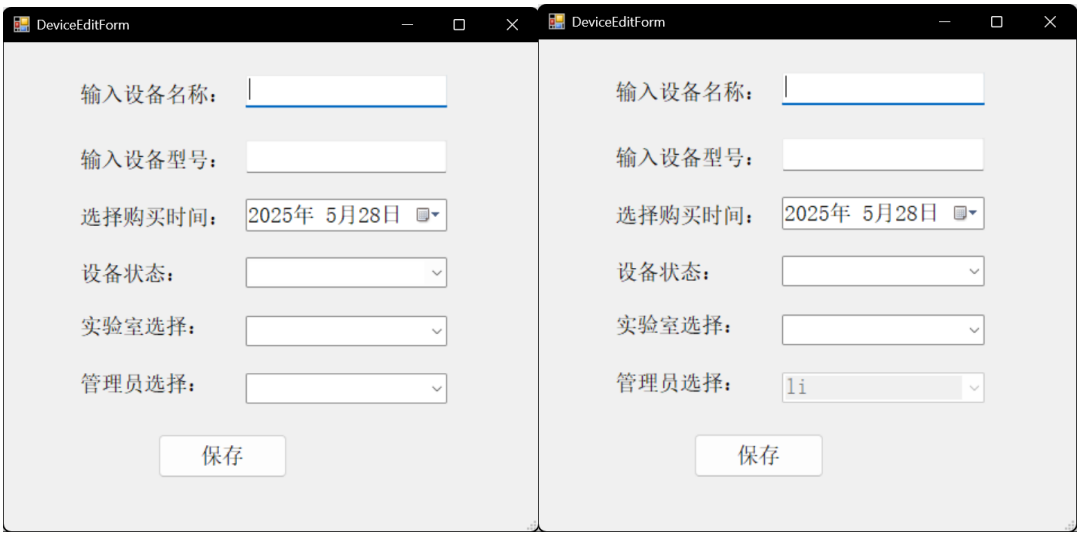


图 4-3-5-2 添加设备下的设备编辑实现界面

图 4-3-5-3 编辑设备下的设备编辑实现界面

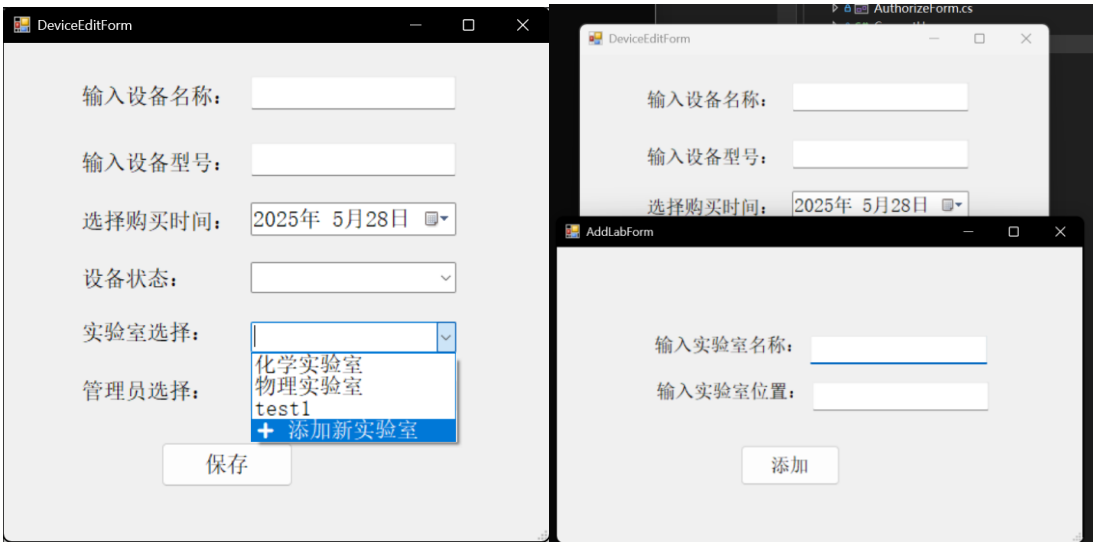


图 4-3-5-4 实验室选择添加实现界面

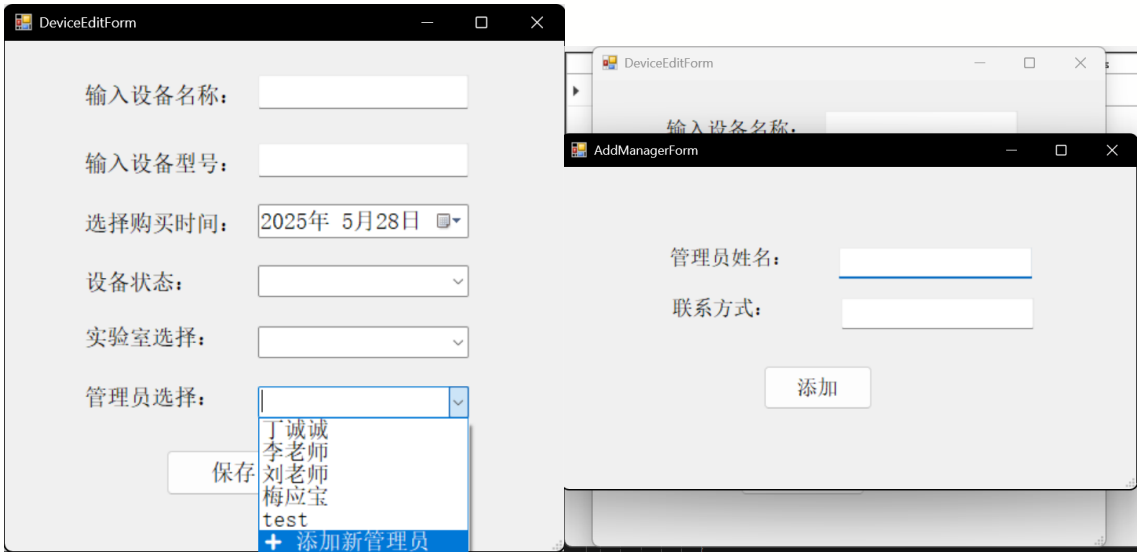


图 4-3-5-5 管理员选择添加实现界面

设备编辑的核心代码如下：

```
3 个引用
private void LoadLabAndManager()
{
    comboLab.Items.Clear();
    comboManager.Items.Clear();

    using (SqlConnection conn = new SqlConnection(connStr))
    {
        conn.Open();

        // 加载实验室
        SqlCommand labCmd = new SqlCommand("SELECT LabID, LabName FROM Lab", conn);
        SqlDataReader labReader = labCmd.ExecuteReader();
        while (labReader.Read())
        {
            comboLab.Items.Add(new ComboBoxItem(labReader["LabName"].ToString(), (int)labReader["LabID"]));
        }
        labReader.Close();

        comboLab.Items.Add(new ComboBoxItem("➕ 添加新实验室", -1));

        if (CurrentUser.Role == "管理员")
        {
            // 加载所有用户作为设备管理员
            SqlCommand mgrCmd = new SqlCommand("SELECT UserName, User_name FROM UserInfo", conn);
            SqlDataReader mgrReader = mgrCmd.ExecuteReader();
            while (mgrReader.Read())
            {
                comboManager.Items.Add(new ComboBoxItem(mgrReader["User_name"].ToString(), mgrReader["UserName"].ToString()));
            }
            mgrReader.Close();

            comboManager.Items.Add(new ComboBoxItem("➕ 添加新管理员", -1));
        }
        else
        {
            // 普通用户，只能选择自己
            comboManager.Items.Add(new ComboBoxItem(CurrentUser.UserName, CurrentUser.UserName));
            comboManager.SelectedIndex = 0;
            comboManager.Enabled = false; // 不允许修改
        }
    }
}
```

```
1 个引用
private void LoadDeviceInfo(int id)
{
    string sql = "SELECT * FROM Device WHERE DeviceID = @id";
    using (SqlConnection conn = new SqlConnection(connStr))
    using (SqlCommand cmd = new SqlCommand(sql, conn))
    {
        cmd.Parameters.AddWithValue("@id", id);
        conn.Open();
        SqlDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            txtName.Text = reader["DeviceName"].ToString();
            txtModel.Text = reader["Model"].ToString();
            datePurchase.Value = Convert.ToDateTime(reader["PurchaseDate"]);
            comboStatus.Text = reader["Status"].ToString();

            // 设置 comboLab 选中项
            foreach (var obj in comboLab.Items)
            {
                if (obj is ComboBoxItem item && (int)item.Value == (int)reader["LabID"])
                {
                    comboLab.SelectedItem = item;
                    break;
                }
            }

            // 设置 comboManager 选中项
            foreach (var obj in comboManager.Items)
            {
                if (obj is ComboBoxItem item && item.Value.ToString() == reader["ManagerID"].ToString())
                {
                    comboManager.SelectedItem = item;
                    break;
                }
            }
        }
    }
}
```

图 4-3-5-6 设备编辑核心代码

4.3.6 用户管理界面设计与实现

这个界面仅限拥有用户管理权限的用户可用，具备添加用户、修改用户、删除用户和管理权限功能。

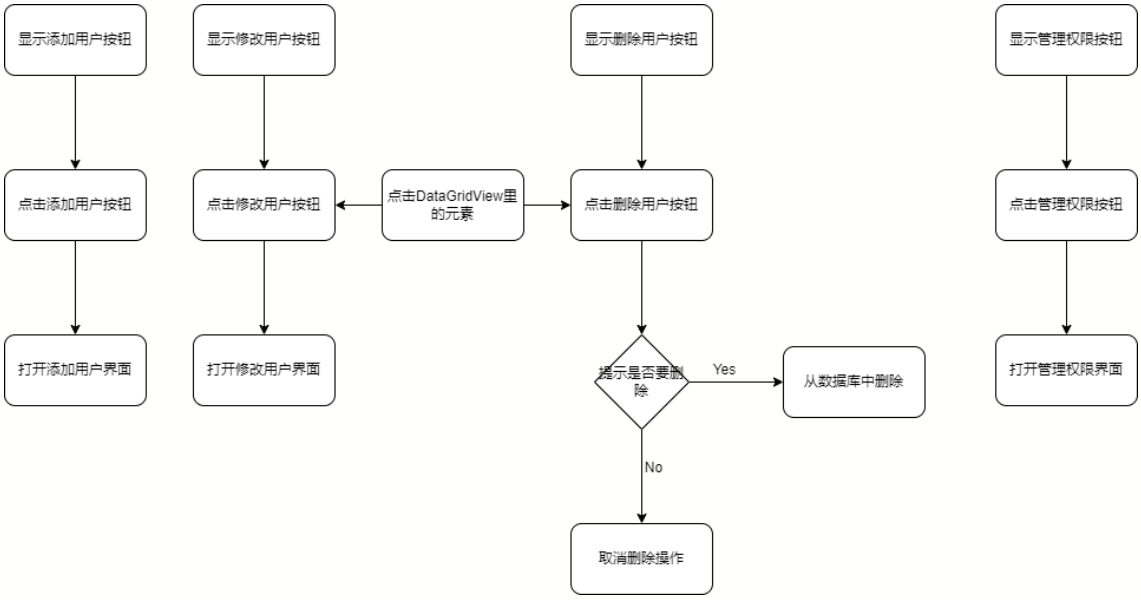


图 4-3-6-1 用户管理界面流程图

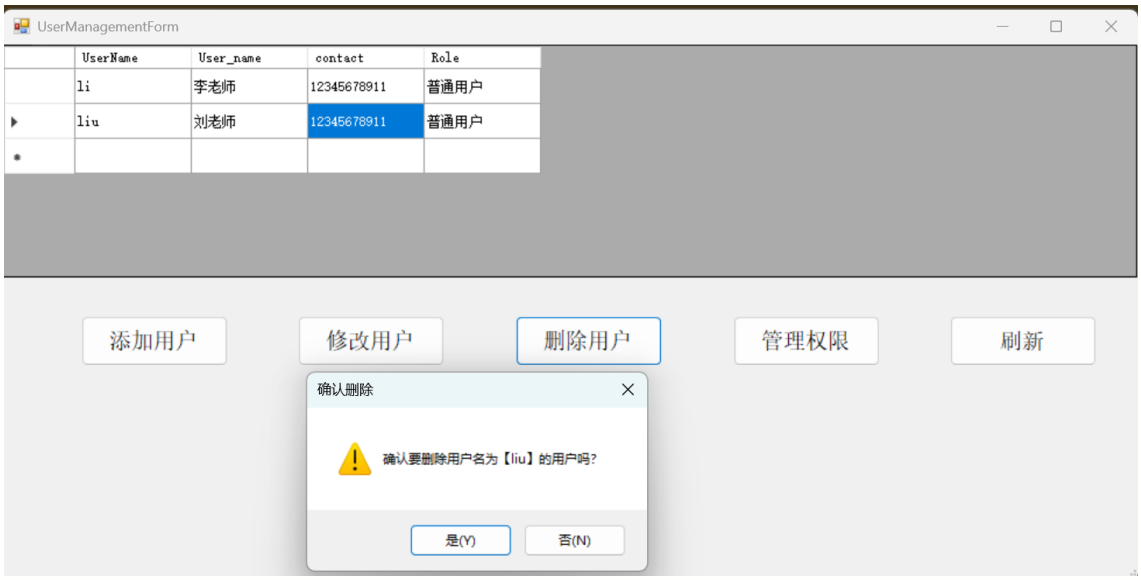


图 4-3-6-2 用户管理具体功能实现界面

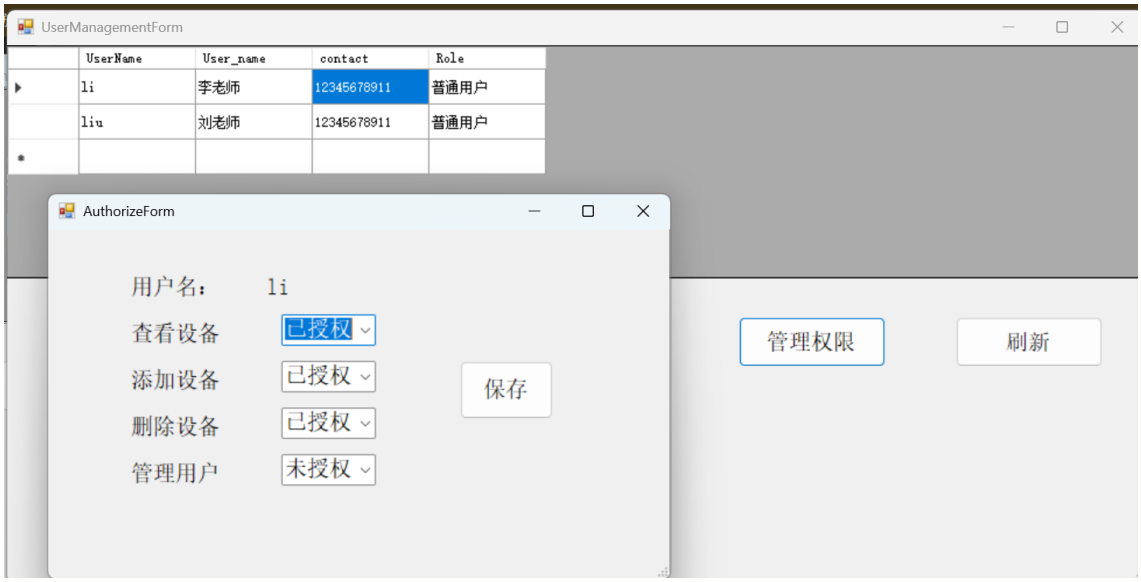


图 4-3-6-3 用户管理具体功能实现界面

用户管理的核心代码如下：

```
1 个引用
private void btnAdd_Click(object sender, EventArgs e)
{
    UserEditForm form = new UserEditForm();
    if (form.ShowDialog() == DialogResult.OK)
    {
        LoadUserList();
    }
}

1 个引用
private void btnEdit_Click(object sender, EventArgs e)
{
    if (dgvUsers.CurrentRow != null)
    {
        string strname=dgvUsers.CurrentRow.Cells["UserName"].Value.ToString();
        UserEditForm form = new UserEditForm(strname);
        if (form.ShowDialog() == DialogResult.OK)
        {
            LoadUserList();
        }
    }
}
```

```
1 个引用
private void btnDelete_Click(object sender, EventArgs e)
{
    if(dgvUsers.CurrentRow != null)
    {
        string strname = dgvUsers.CurrentRow.Cells["UserName"].Value.ToString();
        DialogResult result = MessageBox.Show(
            $"确认要删除用户名为【{strname}】的用户吗？",
            "确认删除",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Warning);
        if (result == DialogResult.Yes)
        {
            using (SqlConnection conn = new SqlConnection(connStr))
            {
                conn.Open();
                // 删除用户
                string sql = @"DELETE FROM UserInfo
                               WHERE UserName = @name";
                using (SqlCommand cmd = new SqlCommand(sql, conn))
                {
                    cmd.Parameters.AddWithValue("@name", strname);
                    cmd.ExecuteNonQuery();
                }
            }
            LoadUserList();
        }
    }
}
1 个引用
private void btnAuthorize_Click(object sender, EventArgs e)
{
    if (dgvUsers.CurrentRow != null)
    {
        string strname = dgvUsers.CurrentRow.Cells["UserName"].Value.ToString();
        AuthorizeForm form = new AuthorizeForm(strname);
        form.ShowDialog();
    }
}
```

图 4-3-6-4 用户管理核心代码

4.3.7　查看日志界面设计与实现

这个界面为全体用户提供日志的查询服务，可查看具体日志信息。

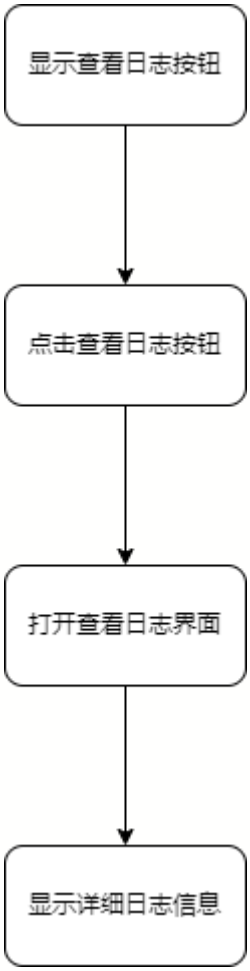


图 4-3-7-1 查看日志界面流程图

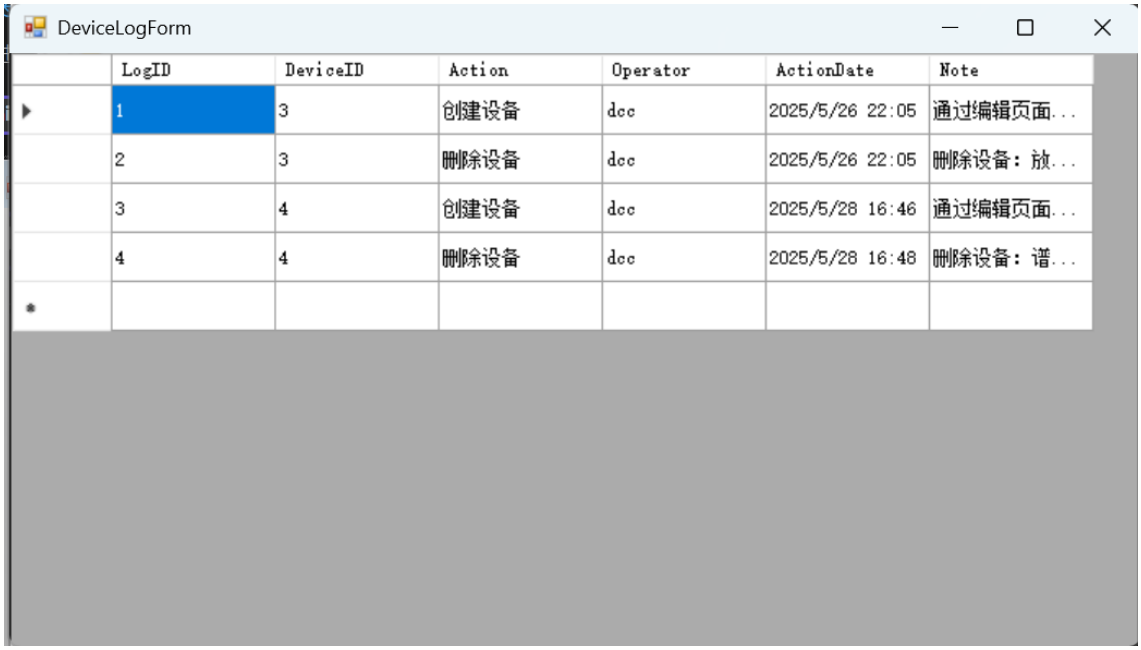


图 4-3-7-2 查看日志实现界面

查看日志的核心代码如下：

```
namespace LabManagement
{
    public partial class DeviceLogForm: Form
    {
        private string connStr = "Data Source=localhost;Initial Catalog=LabDeviceManagement;Integrated Security=True;";

        public DeviceLogForm()
        {
            InitializeComponent();
        }

        1 个引用
        private void DeviceLogForm_Load(object sender, EventArgs e)
        {
            LoadDeviceLog();
        }

        1 个引用
        private void LoadDeviceLog()
        {
            string sql = @"SELECT LogID, DeviceID, Action, Operator, ActionDate, Note
                           FROM DeviceLog";

            using (SqlConnection conn = new SqlConnection(connStr))
            using (SqlCommand cmd = new SqlCommand(sql, conn))
            {
                SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                DataTable table = new DataTable();
                adapter.Fill(table);
                dgvDeviceLog.DataSource = table;
            }
        }
    }
}
```

图 4-3-7-6 查看日志核心代码

5 总结

课程设计围绕《实验室仪器设备管理系统》展开，基于 C#与 SQL Server 技术实现高校实验室设备全生命周期管理。系统通过分层架构设计用户管理、设备管理、实验室管理、日志查询等核心模块，数据库采用 ER 模型构建实体关系。在开发过程中，我们遇到了以下问题：

其一，外码关联数据删除的逻辑校验问题。例如删除实验室记录时，需先判断是否有设备引用该实验室编号，避免因外码约束导致的数据不一致。系统通过 SQL 事务机制，在删除操作前查询设备表引用计数，若存在关联记录则弹出提示，阻止删除操作以保障数据完整性。

其二，数据录入时的属性完整性校验问题。例如用户添加或修改设备信息时，系统对必

填字段进行校验，若检测到字段为空或格式错误，立即弹出提醒信息，强制用户补全信息后再提交至数据库，避免无效数据写入。

系统特色体现在通过 UserPermission 表动态分配权限和动态数据加载机制（实验室/管理员下拉框支持实时刷新与临时新增）。但仍存在跨平台适配不足（仅限 Windows）、统计分析功能单一、大数据量查询性能待优化等局限。未来计划优化外码操作策略，增强前端校验的实时反馈，并通过索引优化、Web 化改造提升系统扩展性与响应效率。本次设计深度实践了数据库事务处理、界面交互逻辑与业务规则校验的整合，为实验室设备信息化管理提供了可落地的解决方案，同时积累了复杂业务场景下的数据一致性保障经验。