目录

中国人民大学[大型仪器预约管理系统实现报告 1](#_Toc55326044)

[一、项目基本信息 1](#_Toc55326045)

[**1．项目人员** 1](#_Toc55326046)

[**2．项目执行情况** 1](#_Toc55326047)

[二、系统任务书 2](#_Toc55326048)

[**1．系统概述** 2](#_Toc55326049)

[**2．系统角色** 2](#_Toc55326050)

[**3. 用户使用流程图** 2](#_Toc55326051)

[**4. 数据字典** 3](#_Toc55326052)

[**5 安全性需求** 3](#_Toc55326053)

[**6. 运行环境需求** 3](#_Toc55326054)

[三、概念结构设计 3](#_Toc55326055)

[**1．E-R图** 3](#_Toc55326056)

[**2．遇到的问题及解决方案** 3](#_Toc55326057)

[四、系统设计 4](#_Toc55326058)

[**1．总体设计** 4](#_Toc55326059)

[**2．数据结构** 5](#_Toc55326060)

[**3. 技术选型** 5](#_Toc55326061)

[**4. 界面设计（可选）** 5](#_Toc55326062)

[五、系统测试 5](#_Toc55326063)

[**1．系统概述** 5](#_Toc55326064)

[**2．安装及部署说明** 7](#_Toc55326065)

中国人民大学大型仪器预约管理系统实现报告

# 一、项目基本信息

**1．项目人员**

**2．项目执行情况**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **任务** | **起始日期** | **结束日期** | **任务负责人**  **及工作量比例** | **任务参与人**  **及工作量比例** |
|  |  |  |  |  |
|  |  |  |  |  |

# 二、系统任务书

**1．系统概述**

目前我国高等院校发展迅速，拥有的大型仪器设备日益增多，提高这些大型仪器设备的使用率，使得这些仪器设备更好的为教学科研发挥更大的作用，成为大型仪器管理的重要目标。

这是一个专为本校大型仪器预约管理而设计的系统，我们致力于实现大型仪器设备信息管理、预约信息管理、用户信息及角色管理、记录实验日志信息等功能模块，使得各大型仪器能够被实时预约、合理使用。

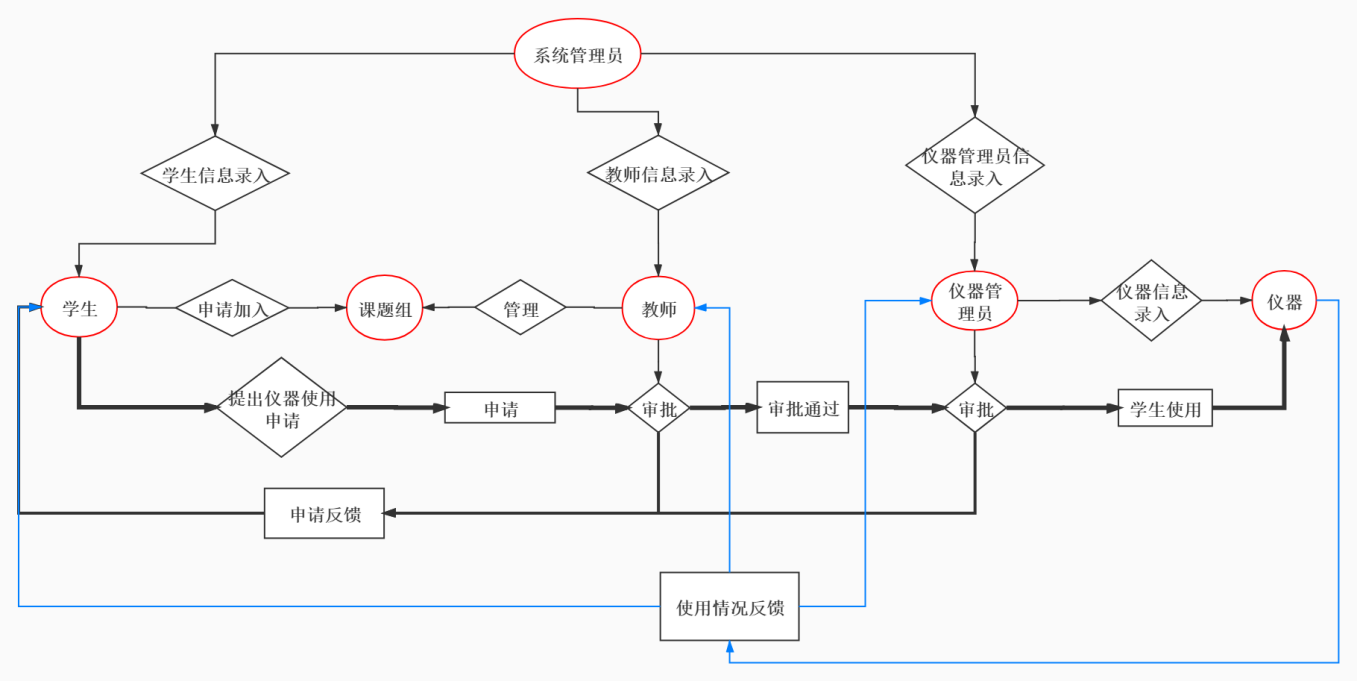
**2．系统角色**

由于管理需求，用户在使用时，根据需求以及权限功能将用户分为4类，分别是：系统管理员，仪器管理员，课题组导师，学生。各类用户具有不同的使用权限。

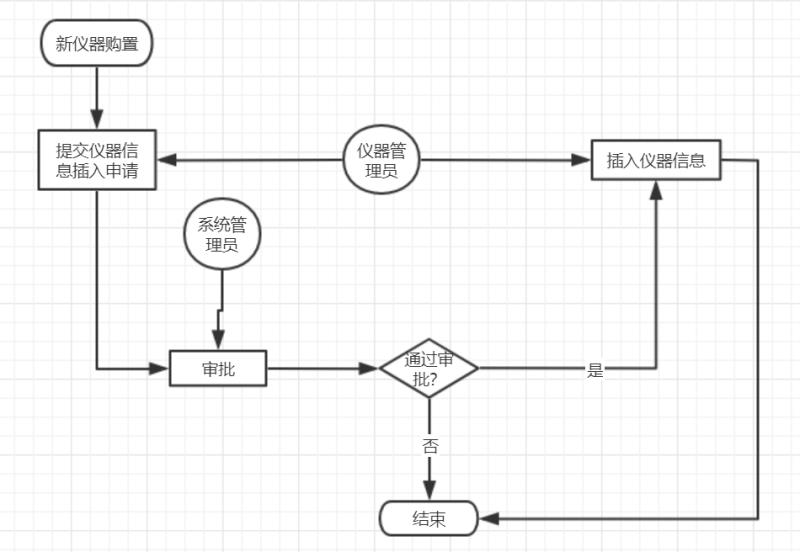
|  |  |  |  |
| --- | --- | --- | --- |
| 角色名称 | 职责描述 | 可使用功能 | 权限等级 |
| 学生 | 直接使用仪器的角色，隶属于一个或多个课题组 | 对系统管理的大型仪器设备的基本信息进行浏览和检索；  网上预约仪器使用；  本人仪器使用日志的浏览以及检索 | 4 |
| 课题组导师 | 作为某个课题组的负责人，负责审批学生用户是否具有操作仪器的资格 | 审批本课题组学生仪器预约申请的资格；  管理本课题组学生信息；  本课题组成员仪器使用日志的浏览以及检索 | 3 |
| 仪器管理员 | 作为一个或多个仪器的管理人，负责仪器设备的管理和维护，负责学生预约的第二次审批 | 对仪器设备的基本信息进行管理和维护；  负责学生预约的第二次审批；  管理维护实验日志；  修改仪器状态（是否故障）；  管理具有仪器使用资格的学生（添加或删除） | 2 |
| 系统管理员 | 拥有最大权限，以保证系统的测试运行成功 | 拥有所有权限 | 1 |

*表1 系统角色分类*

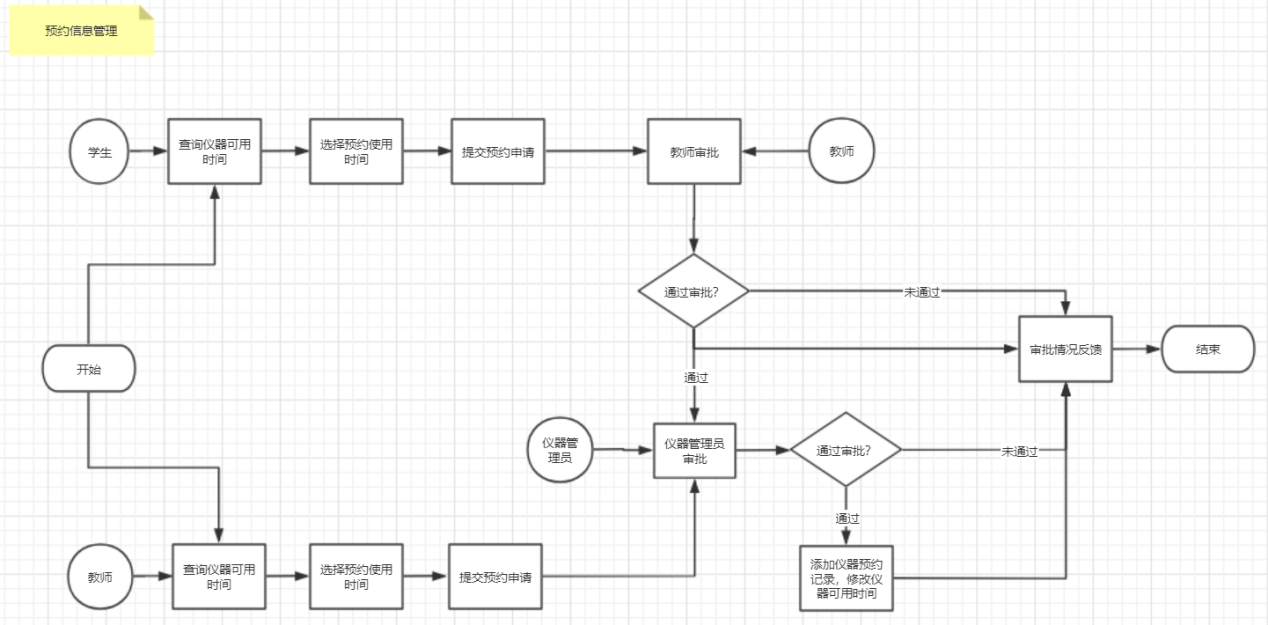
1. **用户使用流程图**



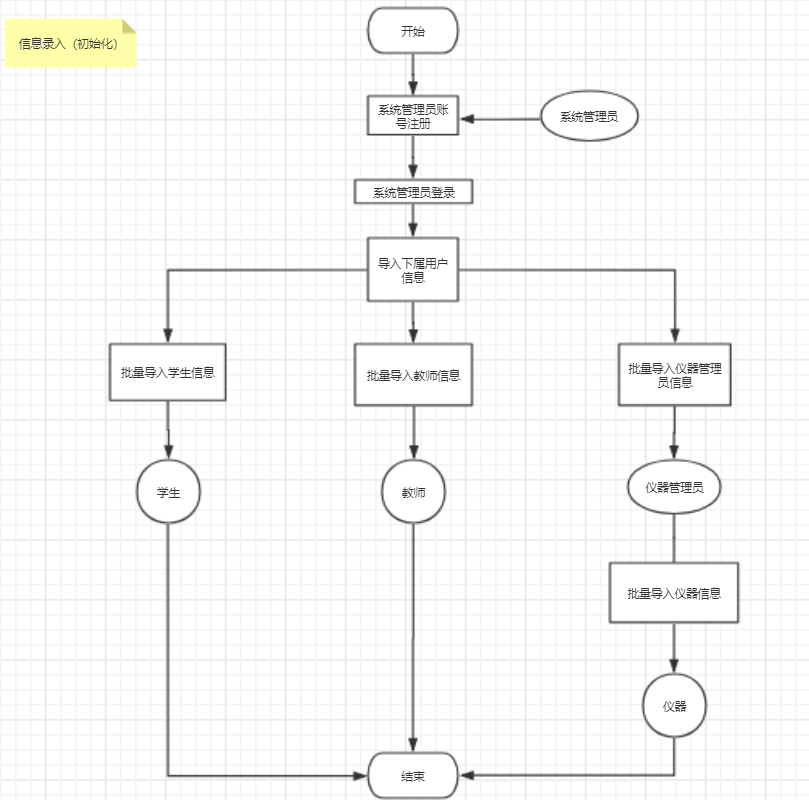
* 1. 大型仪器信息管理

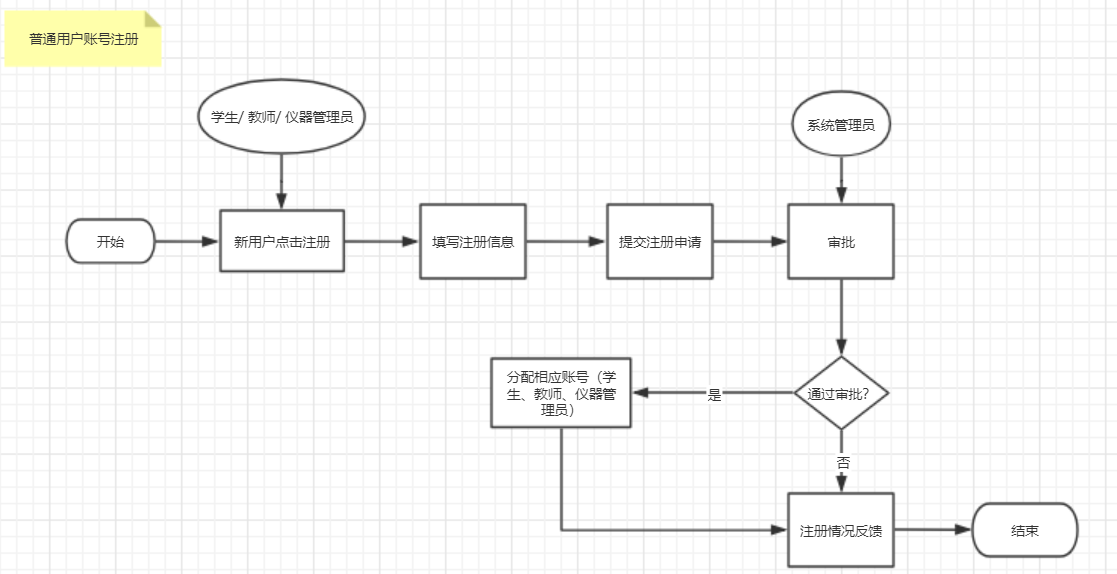


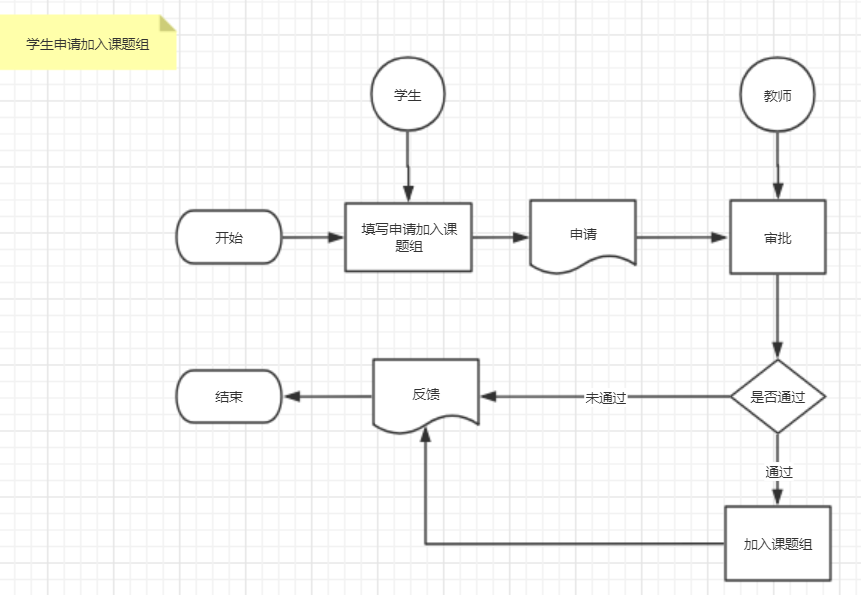
* 1. 预约管理



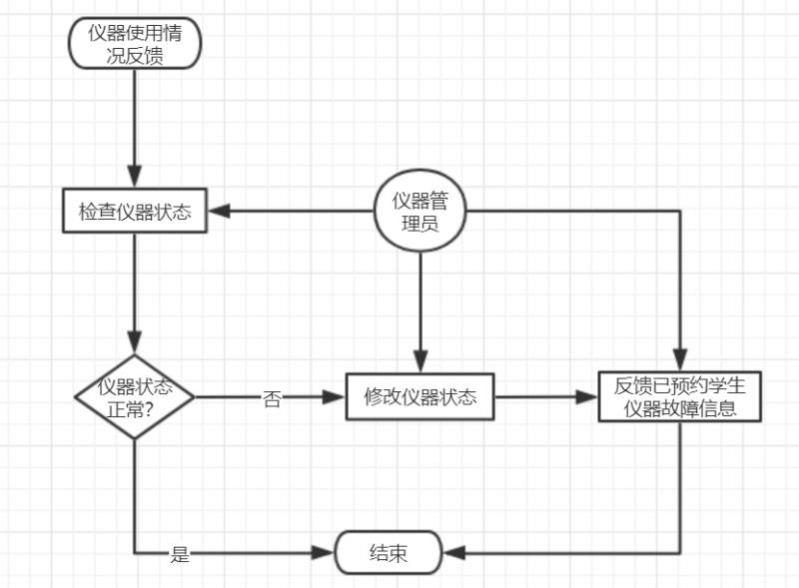
* 1. 用户信息及角色管理

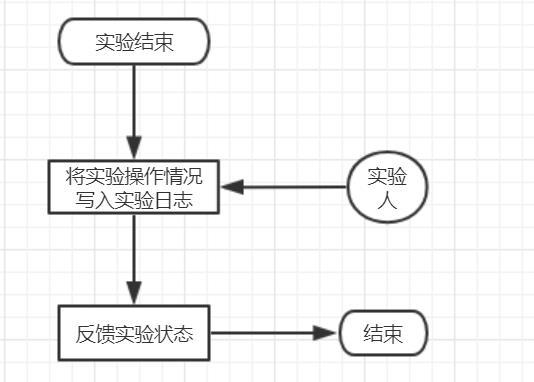






* 1. 实验日志管理





1. **数据字典**

|  |  |  |
| --- | --- | --- |
| 用户(User:) |  |  |
| 账号(UserID) | char(10) | 账号是用户的唯一标识 |
| 密码(UserPassword)  权限(UserPower) | varchar(50)  integer | 密码字符长度在30以内，非空  对应学生,导师,仪器管理员,系统管理员 |
| 姓名(UserName)  性别(Gender)  年级(Session)  职称(Title type)  课题组(UserPID)  邮箱(UserEmail) | vachar(20)  CHAR(2)  CHAR(4)  VARCHAR(10)  char(10)  Varchar(50） | 用户真实姓名，非空  用户性别  针对学生用户，学生年级  针对教师用户，教师职称  用户所在课题组  邮箱，非空 |

|  |  |
| --- | --- |
| 仪器使用日志(EquipLog)： |  |
| 仪器使用日志编号(LogID)  仪器编号(EquipID)  开始使用时间(StartTime)  结束使用时间(EndTime) | char(20)  char(20)  datetime  datetime |
| 操作者账号(UserID)  实验项目(Item)  用户所属课题组编号(Course id)  预约编号(Appoint id)  仪器状况(e\_status)  日志状态(log\_status) | char(10)  varchar(50)  INTEGER  INTEGER  ENUM(‘正常’,’故障’)  ENUM(‘未完成’,’已提交’) |
| 备注(Details) | vachar(200) |

|  |  |
| --- | --- |
| 课题(Project): |  |
| 课题号(PID) | char(10) |
| 课题组导师编号(TID) | char(10) |
| 课题研究细节(Pdetails) | vachar(200) |

|  |  |
| --- | --- |
| 仪器预约申请(Precontract): |  |
| 申请人账号(UserID)  仪器编号(EquipID) | char(10)  char(20) |
| 开始使用时间(StartUseTime)  结束使用时间(EndUseTime)  实验项目(Item) | datetime  datetime  varchar(50) |
| 备注(Details) | vachar(200) |

|  |  |
| --- | --- |
| 大型仪器(Equip): |  |
| 仪器编号(EquipID) | char(20) |
| 仪器名称(EquipName) | varchar(50) |
| 仪器规格(Spec)  功能描述(Desc)  所在地址(Address)  仪器使用资格(qualification)  仪器是否需要操作资格判断(check\_qualification)  仪器购买日期(buy\_date)  仪器状态(status) | vachar(50)  varchar(200)  Varchar(50)  Varchar(50)  Enum(‘是’，’否’)  DATE  Enum(‘正常’，’故障’) |

**5 安全性需求**

通过对仪器预约业务的理解，结合当前预约平台的概况，系统设计需要有严格的权限管理问题，各功能模块需要有相应的权限才能进入（如学生权限不能修改仪器编号等），在同一功能中，不同权限，不同用户可以检索到的内容不同(如每个学生只能查询自己的仪器使用日志，导师可以查询本课题组成员的仪器使用日志)。

**6. 运行环境需求**

6.1操作系统要求：Win10

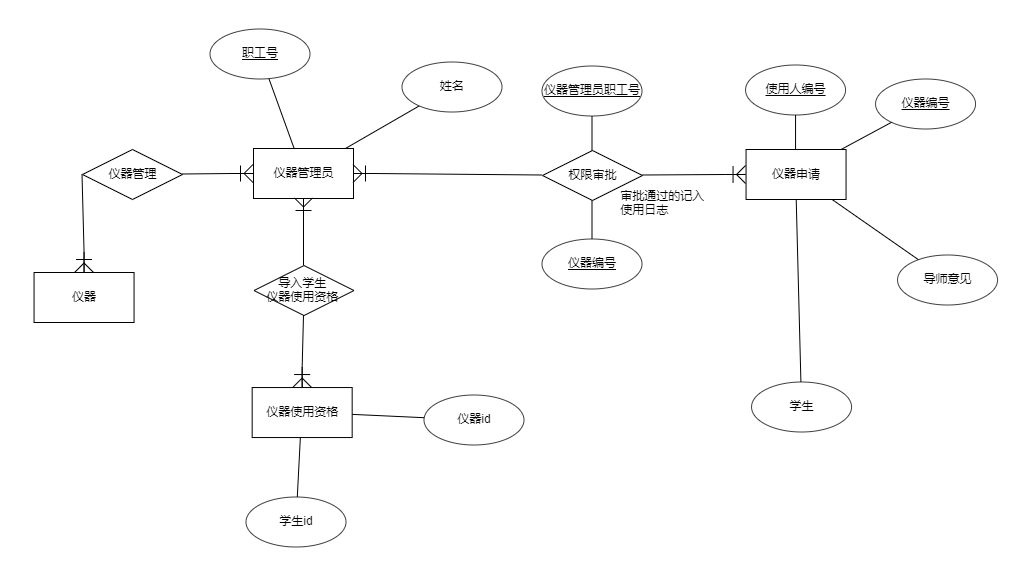
6.2浏览器要求:IE浏览器或google浏览器

6.3软件要求：暂无

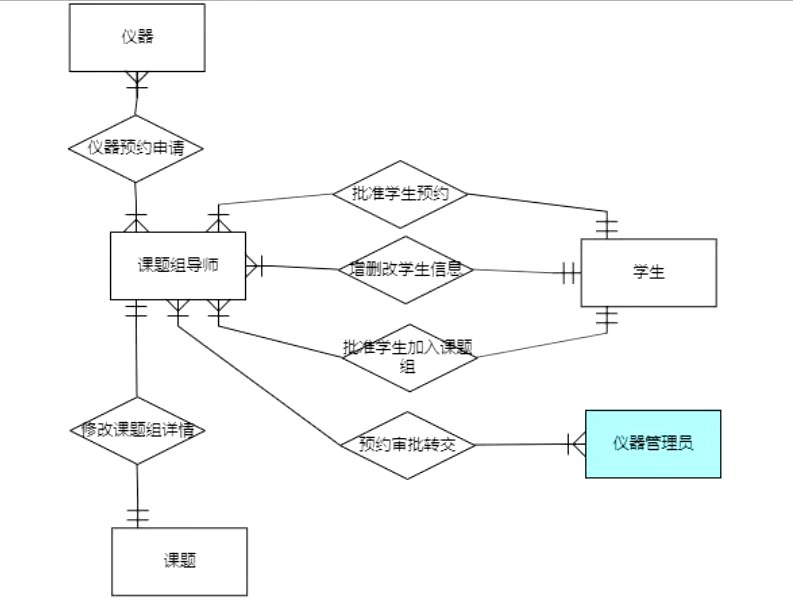
# 三、概念结构设计

**1．E-R图**

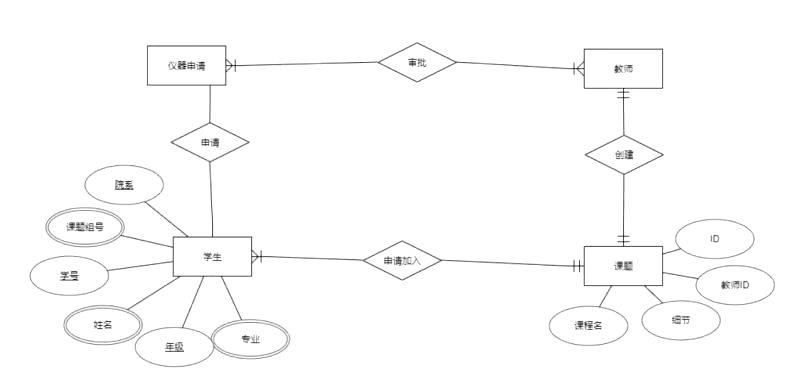
1.1 仪器管理员

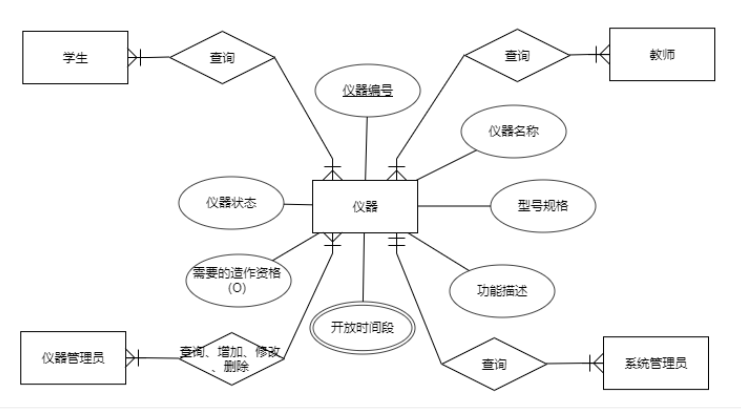


1.2 课题组导师

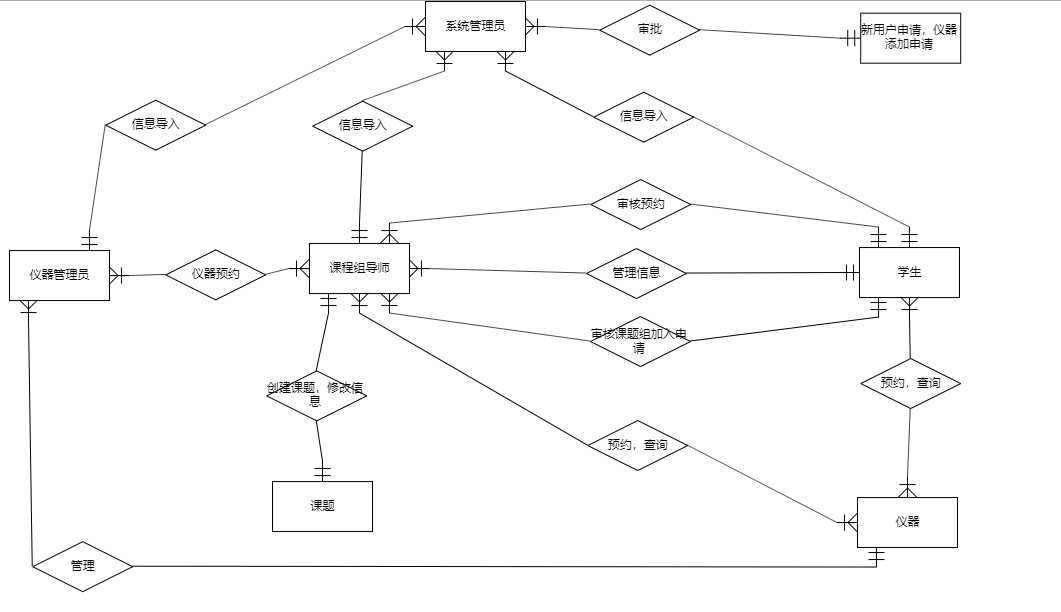


1.3 学生



1.4 仪器

1.5 整体 E-R 图

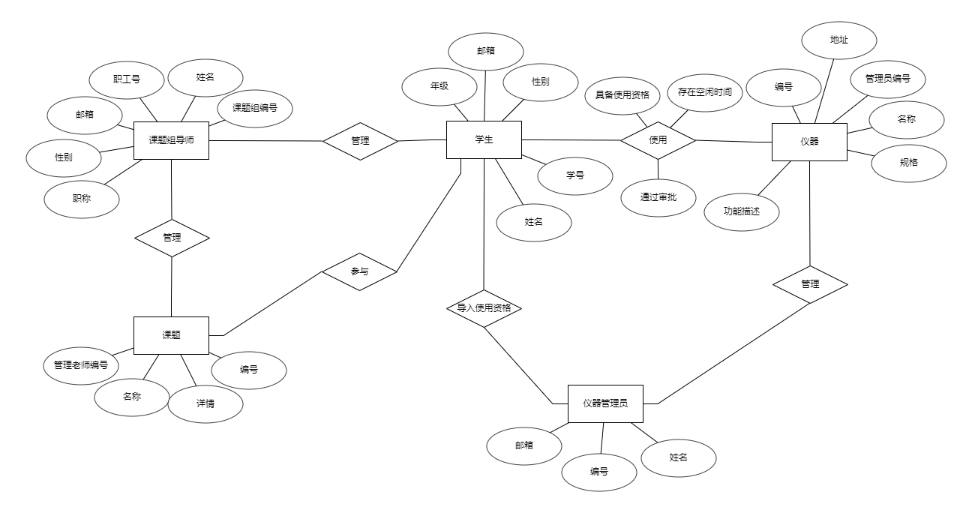


**2．遇到的问题及解决方案**

2.1 逻辑表达比较杂乱，逻辑上相近的主体应该更接近，且关系表达不够清楚，没有突出核心关系

2.2 布局比较混乱，不够简洁明了

**修改后的 E-R 图 :**

****

# 四、系统设计

**1．总体设计**



图1 系统结构图

|  |  |  |
| --- | --- | --- |
| 功能模块 | 功能名称、标识符 | 描述 |
| **账号登录/注册** | 账户密码输入，  记住账户信息 | 接受并匹配用户输入的账号密码，匹配成功跳转到角色对应页面；匹配失败提示错误信息；记住成功登录的用户信息，用户可通过”我的资料”查看自己个人信息 |
| 新账户注册 | 新用户可点击注册创建新账户 |
| **学生** | 课题组查询 | 学生进入界面即可看到自己所属课题组，可点击进入某一个课题组查看详情信息 |
| 仪器信息查询 | 可以查看当前所有仪器信息 |
| 仪器时间预约 | 可以选择仪器的预约使用时间(周/月/日) |
| 查看预约记录 | 查看自己预约使用过的所有记录 |
| 日志编写与查询 | 学生可编写/查看自己已操作实验的实验日志 |
| **课题组导师** | 修改学生信息 | 导师可添加学生进入自己课题组；可从课题组内删除学生 |
| 仪器预约审批 | 导师在学生提交了仪器预约申请之后，进行申请的第一道资格审批 |
| 仪器信息查询 | 可以查看当前所有仪器信息 |
| 日志查询 | 导师可查询课题组下所有学生的仪器使用日志 |
| **仪器管理员** | 仪器预约审批 | 在学生的预约申请通过导师审批后，进行第二道资格审批 |
| 仪器状态修改 | 依据提交的实验日志及反馈，修改仪器的状态为故障/正常 |
| 仪器信息更新 | 更新原有仪器信息 |
| 新仪器添加 | 添加新购置仪器信息 |
| 仪器信息查询 | 查看当前所有仪器信息 |
| 日志查询 | 查看自己所管理的仪器的所有实验日志 |
| **仪器预约** | 查看可用仪器 | 查看当前空闲仪器信息 |
| 选择仪器 | 点击按钮，选择需要预约的仪器 |
| 选择预约时间 | 下拉选择要预约的时间 |
| 导师审批 | 申请提交导师资格审批 |
| 仪器管理员审批 | 申请提交仪器管理员审批 |
| **个人信息操作** | 查看个人资料 | 各角色均可查看个人资料 |

图2 模块功能说明

1. **数据结构**

**2.1 表结构**

2.1.1 课题组

create table Group(

    id INTEGER PRIMARY KEY, -- 课题组编号

    name VARCHAR(100) NOT NULL, -- 课题组名称

    details VARCHAR(100)  -- 课题组描述

);

范式分析: 存在函数依赖: id-->name;id-->details ; 关系以 id 为唯一主键，其他每一个非主属性都完全函数依赖于该候选码，所以属于 2NF，又因为关系中不存在传递依赖，属于 3NF，同时主键只有一个属性，是唯一的决定因素，所以属于BCNF。

完整性：通过定义主键 id,每一个课题组对应唯一 id，保证了实体完整性；又通过定义 NOT NULL 约束，使关系符合日常语义。

改进数据冗余：删除了属性 member\_count,课题组成员数可以通过查询学生-课题组选择表计算得出，所以存在数据冗余，容易引发不一致问题，改进时删除了该属性。

2.1.2 课程组导师

create table Teacher(

    userid VARCHAR(64) PRIMARY KEY, -- 教师id

    name VARCHAR(64) NOT NULL, -- 姓名

    group\_id INTEGER NOT NULL, -- 课题组编号

    title\_type VARCHAR(10) NOT NULL, -- 职称

    gender CHAR(2) NOT NULL, -- 性别

    email VARCHAR(30) NOT NULL, -- 邮箱

**FOREIGN KEY**(group\_id) **REFERENCES** Group(id) ON UPDATE CASCADE

)

范式分析：存在函数依赖：userid-->name,userid-->group\_id,userid-->title\_type,userid-->gender,userid-->email;关系以 userid 为主码，其他所有非主属性均完全函数依赖于 userid 且不存在传递依赖，所以属于 3NF,又 userid 作为唯一决定因素，关系属于 BCNF。

完整性：通过主键 userid 定义关系的实体完整性，通过外键约束，使得课题组导师和课题组之间一一对应，实现参照完整性，同时，通过非空等约束使得关系符合现实语义的要求。

2.1.3 学生

create table Student(

    userid VARCHAR(64) PRIMARY KEY, -- 学生账户

    name VARCHAR(64) NOT NULL, -- 学生姓名

    session CHAR(4) NOT NULL, -- 年级

    gender CHAR(2) NOT NULL, --性别

    email varchar(30) NOT NULL -- 邮箱

);

范式分析：存在函数依赖：userid-->name,userid-->session,userid-->gender,userid-->email;关系以userid 为主码，其他所有非主属性均完全函数依赖于 userid 且不存在传递依赖，所以属于 3NF,又 userid 作为唯一决定因素，关系属于 BCNF。

完整性：通过主键 userid 定义关系的实体完整性，同时，通过非空等约束使得关系符合现实语义的要求。

2.1.4 实验仪器

create table Equip(

    equip\_id VARCHAR(64) PRIMARY KEY, --仪器编号

    equip\_name VARCHAR(50) NOT NULL, -- 仪器名称

    type VARCHAR(50) NOT NULL, -- 仪器规格

    address VARCHAR(50) NOT NULL, -- 仪器所在位置

    description VARCHAR(200), -- 仪器使用描述

    qualification VARCHAR(50),-- 仪器操作资格

    buy\_date DATE NOT NULL, -- 购买日期

    status ENUM('正常','故障') NOT NULL default '正常' -- 仪器状态

);

范式分析：存在函数依赖：equip\_id-->equip\_name,equip\_id-->type,equip\_id-->adress,equip\_id-->buy\_date,equip\_id-->status;equip\_name-->description,equip\_name-->qualification : 在设计关系时，考虑到后续操作简便，若同时存在多台同样的仪器，分设多条记录，拥有不同的 equip\_id,但 equip\_name 相同，因为是同样的仪器，所以使用描述和操作规格是相同的，即 description 和 qualification 依赖于 equip\_name。关系以equip\_id 为主码，每一个非主属性都完全依赖于码，所以属于 2NF，但关系中存在传递依赖 ： equip\_id-->equip\_name,equip\_name-->description &&equip\_id-->equip\_name,equip\_name-->qualification,所以关系不属于3NF,更不属于 BCNF。

完整性：通过主键 equip\_id 定义关系的实体完整性，同时，通过非空等约束使得关系符合现实语义的要求。

2.1.5 仪器管理员

create table EquipManager(

    manager\_id VARCHAR(64) PRIMARY KEY, -- 管理员编号

    name  VARCHAR(20) NOT NULL, -- 姓名

    email VARCHAR(30) NOT NULL -- 邮箱

);

范式分析：存在函数依赖：manager\_id-->name,manager-->email;关系以manager\_id 为主码，其他所有非主属性均完全函数依赖于 manager\_id 且不存在传递依赖，所以属于 3NF,又 manager\_id作为唯一决定因素，关系属于 BCNF。

完整性：通过主键 manager\_id 定义关系的实体完整性，同时，通过非空等约束使得关系符合现实语义的要求。

2.1.6 学生-课题组 (描述学生所加入的课题组)

create table SC(

    student\_id VARCHAR(64), -- 学生 id

    group\_id INTEGER NOT NULL, -- 学生所加入的课题组 id

    PRIMARY KEY(student\_id,group\_id),

**FOREIGN KEY**(group\_id) **REFERENCES** Group(id) ON UPDATE CASCADE **ON DELE TE CASCADE**,

**FOREIGN KEY**(student\_id) **REFERENCES** Student(userid) ON UPDATE CASCAD E **ON DELETE CASCADE**

);

范式分析：关系的两个属性 student\_id,course\_id均为主键，不存在非主属性，所以关系满足 2NF,3NF,BCNF 定义，关系属于 BCNF。

完整性：通过主键(student\_id,course\_id)定义关系的实体完整性，通过外键约束，使得课题组id和学生id分别与系统内的课题组表和学生表对应，实现参照完整性，保证数据的一致性。同时，通过非空等约束使得关系符合现实语义的要求。

2.1.7 管理员-仪器 (描述仪器管理信息：哪些管理员负责哪些仪器)

create table Equip\_Manage(

    equip\_id VARCHAR(64),

    manager\_id VARCHAR(64),

    PRIMARY KEY(equip\_id,manager\_id),

**FOREIGN KEY**(equip\_id) **REFERENCES** Equip(equip\_id) ON UPDATE CASCADE **ON DELETE CASCADE**,

**FOREIGN KEY**(manager\_id) **REFERENCES** EquipManager(manager\_id) ON UPDATE CASCADE **ON DELETE CASCADE**

);

范式分析：关系的两个属性 equip\_id,manager\_id均为主键，不存在非主属性，所以关系满足 2NF,3NF,BCNF 定义，关系属于 BCNF。

完整性：通过主键(equip\_id,manager\_id)定义关系的实体完整性，通过外键约束，使得仪器id和管理员id分别与系统内的仪器表和仪器管理员表对应，实现参照完整性，保证数据的一致性。同时，通过非空等约束使得关系符合现实语义的要求。

2.1.8 仪器预约 (记录单条预约时间信息: 用于设置预约)

create table Appointment(

    appointment\_id INTEGER,

    use\_time INTEGER NOT NULL,

    use\_date DATE NOT NULL,

    PRIMARY KEY(appointment\_id)

);

范式分析：存在函数依赖：appointment\_id-->use\_time,appointment\_id-->use\_date;关系以appointment\_id 为主码，其他所有非主属性均完全函数依赖于 appointment\_id且不存在传递依赖，所以属于 3NF,又 appointment\_id作为唯一决定因素，关系属于 BCNF。

完整性：通过主键 appointment\_id 定义关系的实体完整性，同时，通过非空等约束使得关系符合现实语义的要求。

2.1.9 仪器预约记录（记录用户预约信息: 具体用户预约）

create table Appoint\_index(

    index\_id INTEGER PRIMARY KEY, -- 记录

    start\_id INTEGER, -- 开始时间对应的单条预约的 appointment\_id

    end\_id INTEGER, -- 结束时间对应的单条预约的 appointment\_id

    userid VARCHAR(64), -- 用户 id

    equip\_id VARCHAR(64), -- 仪器 id

    group\_id INTEGER, -- 预约用户所属课题组

    item VARCHAR(200), -- 实验项目

    status ENUM('-4','-3','-2','-1','0','1','2'), -- 预约状态

**FOREIGN KEY**(start\_id) **REFERENCES** Appointment(appointment\_id) ON UPDATE CASCADE,

**FOREIGN KEY**(end\_id) **REFERENCES** Appointment(appointment\_id) ON UPDATE CASCADE,

**FOREIGN KEY**(userid) **REFERENCES** Student(userid) ON UPDATE CASCADE,

**FOREIGN KEY**(equip\_id) **REFERENCES** Equip(equip\_id) ON UPDATE CASCADE,

**FOREIGN KEY**(group\_id) **REFERENCES** Group(id) ON UPDATE CASCADE

);

关系说明：Appointment 表记录的是单条的预约时间信息，如:(1,8,2020/11/29)说明的是 1 号预约记录的仪器使用时间为 2020/11/29 8:00-9:00，仪器的预约使用时间以一个小时为最小单位。Appoint\_index 用以记录具体的用户预约详细信息，若已存在两条 Appointment 记录 l1,l2,则以l1为start\_id,以 l2为end\_id 的用户预约所预约的时间为 l1所记录的时间 -- l2所记录的时间。对 status:依据预约的不同阶段及可能出现的状况为预约设置不同的状态： -4:取消预约，-3:仪器故障或仪器信息被删除,-2:仪器管理员驳回,-1:老师驳回，0:学生提交，1:通过教师审批，2:通过仪器管理员审批。

范式分析：存在函数依赖 index\_id-->start\_id,index\_id-->end\_id,index\_id-->user\_id,index\_id-->equip\_id,index\_id-->item,user\_id-->group\_id,index\_id-->status，关系以index\_id 为主码，每一个非主属性都完全依赖于码，所以属于 2NF，但关系中存在传递依赖 ： index\_id-->user\_id,user\_id-->group\_id ,所以关系不属于3NF,更不属于 BCNF。

完整性：通过主键index\_id定义关系的实体完整性，通过外键约束，使得用户id、仪器id、课题组id及开始/结束预约id分别与系统内已定义的表相对应，实现参照完整性，保证数据的一致性。同时，通过非空等约束使得关系符合现实语义的要求。

2.1.10 实验日志 (实验结束后记录实验操作状态)

create table Equip\_Log(

    log\_id INTEGER PRIMARY KEY, -- 日志编号

    equip\_id VARCHAR(64) NOT NULL, - -仪器编号

    group\_id INTEGER NOT NULL,-- 用户所属课题组编号

    appoint\_id INTEGER NOT NULL, -- 对应用户预约的id

    start\_time TIMESTAMP NOT NULL, --实验开始时间

    end\_time TIMESTAMP NOT NULL, --实验结束时间

    user\_id VARCHAR(64) NOT NULL, -- 实验操作用户id

    item VARCHAR(50) NOT NULL, -- 实验项目

    e\_status ENUM('正常','故障'), -- 仪器状态

    details VARCHAR(200),  -- 实验描述

    log\_status ENUM('未完成','已提交'), -- 日志状态

**FOREIGN KEY**(appoint\_id) **REFERENCES** Appoint\_index(index\_id) ON UPDATE CASCADE,

**FOREIGN KEY**(equip\_id) **REFERENCES** Equip(equip\_id) ON UPDATE CASCADE

,

**FOREIGN KEY**(userid) **REFERENCES** Student(userid)ON UPDATE CASCADE

,

**FOREIGN KEY**(group\_id) **REFERENCES** Group(id)ON UPDATE CASCADE

);

关系说明：用户在进行预约时，若预约成功会在日志表中插入一条日志状态为‘未完成’的日志记录，当用户完成实验，填写实验日志并点击提交之后，该‘未完成’记录的状态被修改为‘已提交’。

范式分析：存在函数依赖 log\_id-->start\_time,log\_id-->end\_time,log\_id-->user\_id,log\_id-->equip\_id,index\_id-->item,user\_id-->group\_id,equip\_id-->e\_status，log\_id-->details,log\_id-->log\_status关系以log\_id 为主码，每一个非主属性都完全依赖于码，所以属于 2NF，但关系中存在传递依赖 ： log\_id-->user\_id,user\_id-->group\_id && log\_id-->equip\_id,equip\_id-->e\_status,所以关系不属于3NF,更不属于 BCNF。

完整性：通过主键log\_id定义关系的实体完整性，通过外键约束，使得用户id、仪器id、课题组id及预约id分别与系统内已定义的表相对应，实现参照完整性，保证数据的一致性。同时，通过非空等约束使得关系符合现实语义的要求。

**2.2 触发器**

为了在一些事件发生时，保证数据的完整性和一致性，定义了如下触发器，确立相关符合语义的控制约束规则。

2.2.1 故障状态检查触发器

DELIMITER $$

CREATE TRIGGER FaultCheck

AFTER UPDATE ON Equip\_Log

FOR EACH ROW

BEGIN

IF NEW.log\_status='已提交'and NEW.e\_status='故障' THEN

UPDATE Equip

SET status='故障'

WHERE equip\_id = NEW.equip\_id;

END IF;

END$$

用户在实验之后，需提交实验日志，实验日志中会报告仪器现在的使用状态，若用户在正式提交实验日志时上报了仪器为故障，则该仪器在实验中被损坏，此时，需修改实验仪器表对应仪器的状态为故障。考虑到用户修改实验日志之后可能只是保存而并没有提交日志，所以该触发器的触发条件为：日志状态为已提交且仪器状态为故障，触发的操作为更新仪器表，修改对应仪器状态。

2.2.2 预约状态检查触发器

DELIMITER $$

CREATE TRIGGER StatusCheck

AFTER UPDATE ON Equip

FOR EACH ROW

BEGIN

IF NEW.status='故障' THEN

UPDATE Appoint\_index

SET status='-3'

WHERE equip\_id = NEW.equip\_id;

END IF;

END$$

若用户已预约了某仪器，此时已存在预约记录，但在预约进行之前仪器发生故障被上报，仪器表中的仪器状态会被修改为故障，此时仪器不可用，应修改该仪器已有的所有预约，设置预约状态为 -3，表示仪器已故障。

2.2.3 仪器删除触发器

DELIMITER $$

CREATE TRIGGER DeleteEquip

BEFORE DELETE ON Equip

FOR EACH ROW

BEGIN

UPDATE Appoint\_index

SET equip\_id = '-1'

WHERE equip\_id = OLD.equip\_id;

END$$

DELIMITER $$

CREATE TRIGGER DeleteEquip1

BEFORE DELETE ON Equip

FOR EACH ROW

BEGIN

UPDATE Appoint\_index

SET status = '-3'

WHERE equip\_id = OLD.equip\_id;

END$$

DELIMITER $$

CREATE TRIGGER DeleteEquip2

BEFORE DELETE ON Equip

FOR EACH ROW

BEGIN

UPDATE Equip\_Log

SET equip\_id = '-1'

WHERE equip\_id = OLD.equip\_id;

END$$

定义一个特殊仪器，其编号为 -1，该仪器不可使用，当某一仪器信息被删除时，更改已有的该仪器相关的信息(Equip\_Log,Appoint\_index),用编号为 -1 的不可用仪器替换被删除的仪器信息，使得预约及实验日志记录能够保留同时又保证数据的正确性和一致性。

2.2.4 预约撤销触发器

DELIMITER $$

CREATE TRIGGER WithDraw

AFTER UPDATE ON Appoint\_index

FOR EACH ROW

BEGIN

IF NEW.status = '-4' THEN

DELETE

FROM Equip\_Log

WHERE appoint\_id = NEW.index\_id;

END IF;

END$$

由于预约的实现机制为：用户提交预约之后，会同时在实验日志中增加一条状态为‘未完成’的日志记录，若用户正常进行实验并提交实验日志，只需更改此日志记录的状态为‘已提交’即可。但若用户中途撤销预约，此时需删除该预存的日志记录，所以定义该触发器，在用户撤回预约，即Appoint\_index表中对应预约状态被修改为 -4 时，删除实验日志表中的相应记录，保证数据的一致准确。

2.2.5 资格检查触发器

DELIMITER $$

CREATE TRIGGER Check\_Qua

AFTER UPDATE ON Equip

FOR EACH ROW

BEGIN

IF OLD.check\_qualification= '是' and NEW.check\_qualification= '否' THEN

DELETE

FROM Qualification

WHERE equip\_id = NEW.equip\_id;

END IF;

END$$

学生在获得了某个仪器的使用资格之后，仪器管理员会导入学生对仪器的使用资格，学生申请使用仪器之前，系统会先查找该生是否具备相应仪器的使用资格，但有一些仪器并不需要专业的使用资格，这些仪器在申请时不需要事先查找使用资格。定义该触发器，当仪器是否需要专业使用资格的条件被修改时，修改资格表中的已有记录，保证数据的一致性。

**2.3 数据操纵**

2.3.1 设置主键编号自增

ALTER TABLE `Equip\_Log` CHANGE `log\_id` `log\_id` INT NOT NULL AUTO\_INCREMENT;

ALTER TABLE `appointment` CHANGE `appointment\_id` `appointment\_id` INT NOT NULL AUTO\_INCREMENT;

ALTER TABLE `Appoint\_index` CHANGE `index\_id` `index\_id` INT NOT NULL AUTO\_INCREMENT;

设置实验日志表的 log\_id, 预约表的 appointment\_id 以及预约记录表的 index\_id 为自增属性,以保证数据记录在插入或删除时的完整性和一致性。

2.3.2 数据级联更新

在上述各关系中存在外键约束的属性上，设置数据级联更新/删除，保证数据更新过程中的数据完整性和一致性

2.3.3 定期清理数据记录

对于实验日志记录，实验预约记录等记录信息，并无长期存在的必要且若一直留存，会占用大量空间且增大查询的复杂度，所以定义删除事件，定期清理记录信息，只保留近三个月内的记录。

DELIMITER //

create procedure del\_data()

BEGIN

DELETE FROM `Appoint\_index` WHERE created\_on < DATE\_SUB(CURDATE(),INTERVAL 90 DAY);

END//

DELIMITER ;

DELIMITER //

create procedure del\_data2()

BEGIN

DELETE FROM `Appointment` WHERE created\_on < DATE\_SUB(CURDATE(),INTERVAL 90 DAY);

END//

DELIMITER ;

DELIMITER //

create procedure del\_data1()

BEGIN

DELETE FROM `Equip\_Log`WHERE created\_on < DATE\_SUB(CURDATE(),INTERVAL 90 DAY);

END//

DELIMITER ;

SET GLOBAL event\_scheduler = ON;

create event del\_event

on schedule

EVERY 1 day

STARTS '2020-12-4 00:00:00'

do call del\_data();

create event del\_event1

on schedule

EVERY 1 day

STARTS '2020-12-4 00:00:00'

do call del\_data1();

create event del\_event2

on schedule

EVERY 1 day

STARTS '2020-12-4 00:00:00'

do call del\_data2();

1. **技术选型**

3.1 数据库配置

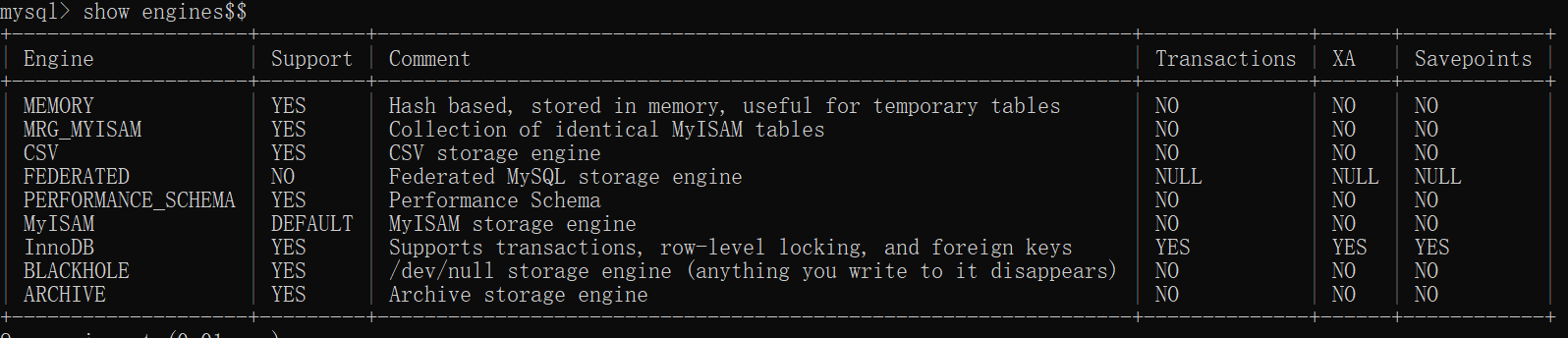
3.1.1 数据库服务器

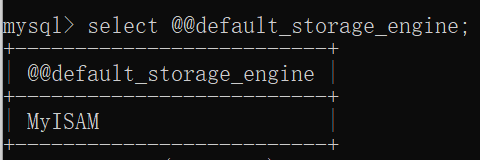
* 服务器：MySQL(127.0.0.1 via TCP/IP);
* 服务器类型：MYSQL;
* 服务器版本：8.0.21-MySQL Community Sever -GPL
* 协议版本 10
* 服务器字符集：UTF-8 Unicode

3.1.2 数据库物理存储结构

* 存储引擎

查看数据库存储引擎：



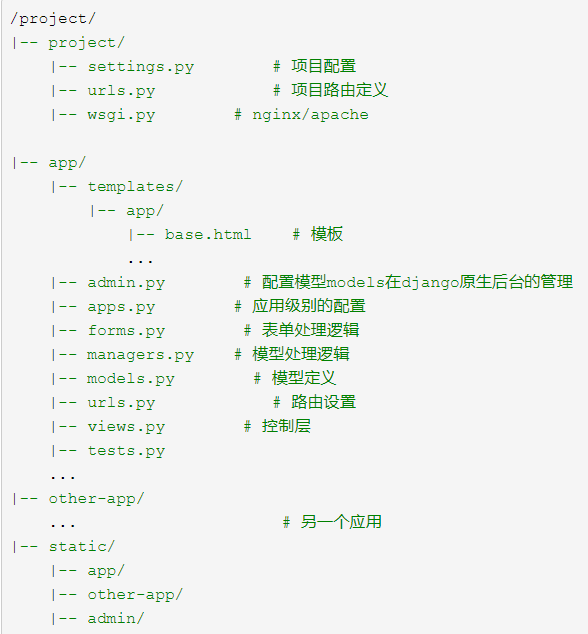


可以看到，系统默认的存储引擎为 MyISAM，我们在创建表时通过限定 engine = InnoDB,修改了存储引擎为 InnoDB。

3.2 前端实现

前端界面采用 Django + Bootstrap框架实现。使用此技术选型的原因在于Django和Bootstrap都可找到较为详细的教程，且能够很好地实现前端页面功能设计，满足项目需求。

3.2.1 Django 项目结构



3.2.2 静态文件

django的静态文件位置没有限制，为了方便管理,统一在根目录下面，按应用名建立目录。这样在web服务器 apahce/nginx 可以方便配置路径别名（Alias）转交web服务器直接进行处理。同时在 settings.py 中设置STATIC\_URL=`/project/static`，以在模板文件中通过{%static `app/xxx.css`}调用

3.2.3 Django 原生后台管理 admin.py

django 自带admin 后台，通过 admin.py 很容易对简单的模型进行增删改查管理操作。在业务逻辑不是很复杂的情况下，简单配置一下就可以实现对数据的管理。

3.2.4 Django 路由控制 urls.py

3.2.5 Django 模型定义 models.py && manage.py

django 把数据库结构也包含到了项目中，models的定义主要包括数据表结构的各个字段(Field)，表与表之间的关系(ForeignKey)，对表的查询操作(Manager)。Field 和 Manger 都支持自定义django 会根据models中的定义自动建立数据表。

managers.py 属于 models 的一部分，它主要提供2个功能：1、修改默认的数据集 objects,继承models.Manager,重写 get\_queryset 方法，对默认的 queryset 进行预处理;2、扩展默认的查询方法(可定义自己想要的各种复杂查询)

3.2.6 Django 逻辑控制 views.py & forms.py

该部分相当于MVC中的 Controller 控制层。定义为各种函数，从传入的 request 中获取请求参数，生成 response。django 提供了很多封装函数如reer,get\_object\_or\_404来方便处理。此外，还可以使用基本视图类处理请求。 forms 属于 views 的一部分，主要负责 views 从 request 中获取参数并预处理这部分逻辑。

3.2.7 Django 视图模板 templates

模板选择在views中通过render制定,或继承的 TemplateView 中定义template\_name。template 文件支持继承，子模板通过重写 <% block %> 块来渲染不同页面。

3.2.8 Django 处理 HTML 请求流程

1) 加载project settings  
 在通过django-admin.py创建project的时候，Django会自动生成默认的settings文件和manage.py等文件，在创建WSGIServer之前会执行下面的引用：from django.conf import settings。上面引用在执行时，会读取os.environ中的DJANGO\_SETTINGS\_MODULE配置，加载项目配置文件，生成settings对象。所以，在manage.py文件中你可以看到，在获取WSGIServer之前，会先将project的settings路径加到os路径中。

2) 创建WSGIServer  
 不管是使用runserver还是uWSGI运行Django项目，在启动时都会调用django.core.servers.basehttp中的run()方法，创建一个django.core.servers.basehttp.WSGIServer类的实例，之后调用其serve\_forever()方法启动HTTP服务。

在创建WSGIServer实例的时候会指定HTTP请求的Handler。当用户的HTTP请求到达服务器时，WSGIServer会创建WSGIRequestHandler实例，使用其handler方法来处理HTTP请求。WSGIServer通过set\_app方法设置一个可调用(callable)的对象作为application，handler方法最终会调用设置的application处理request，并返回response。其中，WSGIServer继承自wsgiref.simple\_server.WSGIServer,而WSGIRequestHandler继承自wsgiref.simple\_server.WSGIRequestHandler，wsgiref是Python标准库给出的WSGI的参考实现。

3)处理Request  
 第二步中说到的application,在Django中一般是django.core.handlers.wsgi.WSGIHandler对象，WSGIHandler继承自django.core.handlers.base.BaseHandler，这个是Django处理request的核心逻辑，它会创建一个WSGIRequest实例，而WSGIRequest是从http.HttpRequest继承而来。

4) 返回Response  
 上面提到的BaseHandler中有个get\_response方法，该方法会先加载Django项目的ROOT\_URLCONF，然后根据url规则找到对应的view方法(类)，view逻辑会根据request实例生成并返回具体的response。

3.3 功能实现访问流程



1. **物理设计**

在修改数据库存储引擎为 InnoDB 后，使用 InnoDB 的默认行为和结构

* 1. 关系模式存取方法

借助 B+ 树索引进行存取。

B+树属于多路平衡搜索树，理论复杂度和平衡二叉树相同操作都是log(n)。因为多路的特性，I/O操作上更有优势，并且也让树的深度降低。所以，设计B+树时，是**尽可能让树的深度降低**的。而**使用B+树而不是B-数**，是因为B+树的关键字全在叶子节点。这样一来，硬盘一个簇可以存更多节点。可减少IO次数。

InnoDB 存储引擎就是用 B+Tree 实现其索引结构。在 B-Tree 中，每个节点中有 key，也有 data，而每一个页的存储空间是有限的，如果 data 数据较大时将会导致每个节点（即一个页）能存储的 key 的数量很小。当存储的数据量很大时同样会导致 B-Tree 的深度较大，增大查询时的磁盘 I/O 次数，进而影响查询效率。在 B+Tree 中，所有数据记录节点都是按照键值大小顺序存放在同一层的叶子节点上，而非叶子节点上只存储 key 值信息，这样可以大大加大每个节点存储的 key 值数量，降低 B+Tree 的高度。

InnoDB 是以 ID 为索引的数据存储。采用 InnoDB 引擎的数据存储文件有两个，一个定义文件，一个是数据文件。InnoDB 通过 B+Tree 结构对 ID 建索引，然后在叶子节点中存储记录。若建索引的字段不是主键 ID，则对该字段建索引，然后在叶子节点中存储的是该记录的主键，然后通过主键索引找到对应的记录。

4.2 数据库存储结构

InnoDB数据存储结构：

1. **表的存储**：

MySQL 使用 InnoDB 存储表时，将表的定义和数据索引等信息分开存储，其中前者存储在 .frm 文件中，后者存储在 .ibd 文件中。

.frm:所有的 MySQL 表都会在硬盘上创建一个 .frm 文件用来描述表的格式/定义;.frm 文件的格式在不同的平台上均相同

.ibd:InnoDB 中用于存储数据的文件总共有两个部分，一是系统表空间文件，包括 ibdata1、 ibdata2 等文件，其中存储了 InnoDB 系统信息和用户数据库表数据和索引，是所有表公用的。  
当打开 innodb\_file\_per\_table 选项时，.ibd 文件就是每一个表独有的表空间，文件存储了当前表的数据和相关的索引数据。

**2、表空间**：

innodb存储引擎在存储设计上模仿了Oracle的存储结构，其数据是按照表空间进行管理的。新建一个数据库时，innodb存储引擎会初始化一个名为ibdata1 的表空间文件，默认情况下，这个文件会存储所有表的数据，以及我们所熟知但看不到的系统表sys\_tables、 sys\_columns、sys\_indexes 、sys\_fields等。此外，还会存储用来保证数据完整性的回滚段数据。

innodb存储引擎的设计很灵活，可以通过参数innodb\_file\_per\_table来设置，使得每一个表都对应一个自己的独立表空间文件，而不是存储到公共的ibdata1文件中。独立的表空间文件只存储对应表的B+树数据、索引和插入缓冲等信息，其余信息还是存储在默认表空间中。(索引文件由段,簇，页面组成)

**3、段**

  段是表空间文件中的主要组织结构，用来管理物理文件，是构成索引、表、回滚段的基本元素。  
  表空间是由各个段组成的，常见的段有数据段、索引段、回滚段等。InnoDB存储引擎表是索引组织的，因此数据即索引，索引即数据。那么数据段即为B+树的页节点，索引段即为B+树的非索引节点。

**4、簇**

innodb引入了簇的概念，在代码中被称为extent；簇是由64个连续的页组成的，每个页大小为16KB，即每个簇的大小为1MB。簇是构成段的基本元素，一个段由若干个簇构成。一个簇是物理上连续分配的一个段空间，每一个段至少会有一个簇，在创建一个段时会创建一个默认的簇。如果存储数据时，一个簇已经不足以放下更多的数据，此时需要从这个段中分配一个新的簇来存放新的数据。一个段所管理的空间大小是无限的，可以一直扩展下去，但是扩展的最小单位就是簇。

**5、页**

页是InnoDB磁盘管理的最小单位。常见的页类型有：数据页(B-tree Node),Undo页(Undo Log Page),系统页(System Page),事务数据页(Transaction system Page),插入缓冲位图页(Insert Buffer Bitmap),插入缓冲空闲列表页(Insert Buffer Free List),未压缩的二进制大对象页(Uncompressed BLOB Page),压缩的二进制大对象页(Compressed BLOB Page)。在逻辑上(页面号都是从小到大连续的)及物理上都是连续的。在向表中插入数据时，如果一个页面已经被写完，系统会从当前簇中分配一个新的空闲页面处理使用，如果当前簇中的64个页面都被分配完，系统会从当前页面所在段中分配一个新的簇，然后再从这个簇中分配一个新的页面来使用。

1. **文件维护**

InnoDB通过Inode Entry来管理每个Segment占用的数据页，每个segment可以看做一个文件页维护单元。Inode Entry所在的inode page有可能存放满，因此又通过头Page维护了Inode Page链表。  
  在.ibd的第一个Page中还维护了表空间内Extent的FREE、FREE\_FRAG、FULL\_FRAG三个Extent链表；而每个Inode Entry也维护了对应的FREE、NOT\_FULL、FULL三个Extent链表。这些链表之间存在着转换关系，以高效的利用数据文件空间。注意区别：表空间中的链表管理的是整个表空间中所有的簇，包括满簇、半满簇及空闲簇，而段的iNode信息中管理的是属于自己段中的满簇、半满簇及空闲簇。当创建一个新的索引时，实际上构建一个新的btree(btr\_create)，先为非叶子节点Segment分配一个inode entry，再创建root page，并将该segment的位置记录到root page中，然后再分配leaf segment的Inode entry，并记录到root page中。当删除某个索引后，该索引占用的空间需要能被重新利用起来。

# 五、系统测试

**1．系统概述**

这是一个专为本校大型仪器预约管理而设计的系统，我们实现了大型仪器设备信息管理、预约信息管理、用户信息及角色管理、记录实验日志信息等功能模块，使得各大型仪器能够被实时预约、合理使用。

本系统的功能展示：

各模块的详细功能展示以及测试如下：

1. **模块：**登陆注册等主页面

**测试名称：**验证系统主页，用户登陆，用户注册界面的功能

**测试目的：**验证基于账号、密码的登陆结果，验证基于账号、密码以及其他详细信息的注册结果。

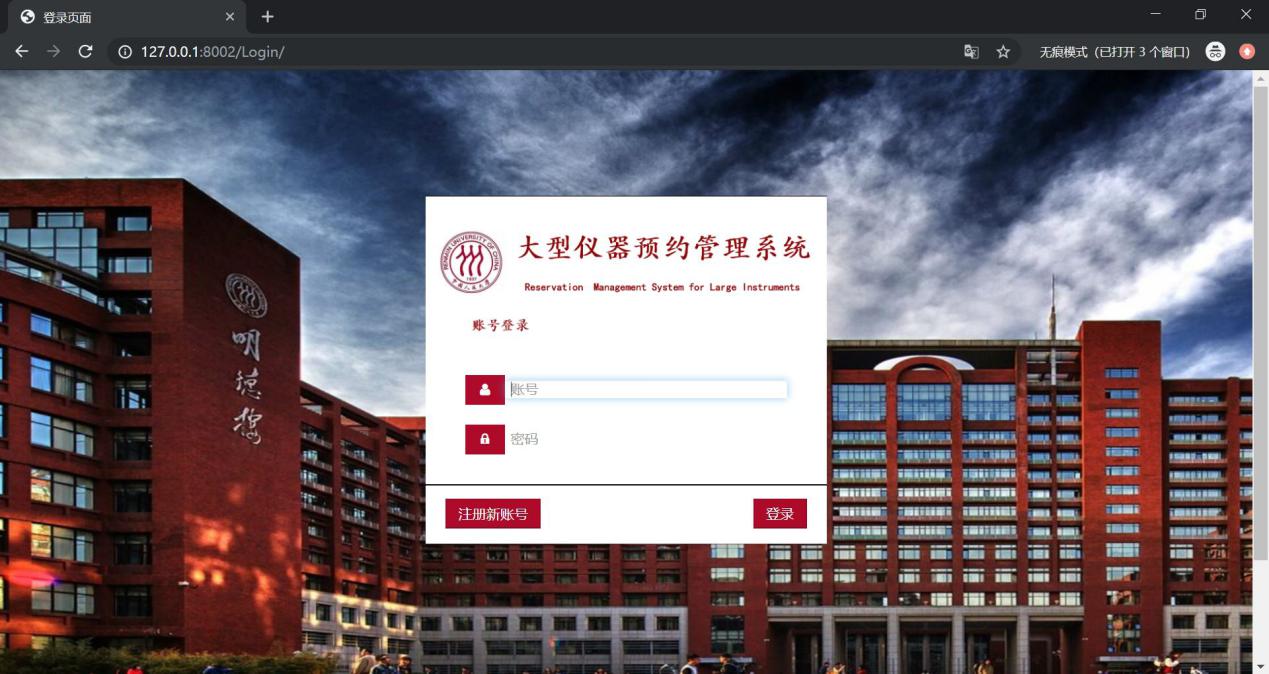
1. Homepage：期待输出系统homepage，可以由该页面跳转到登陆和注册界面。测试成功。

测试截图：



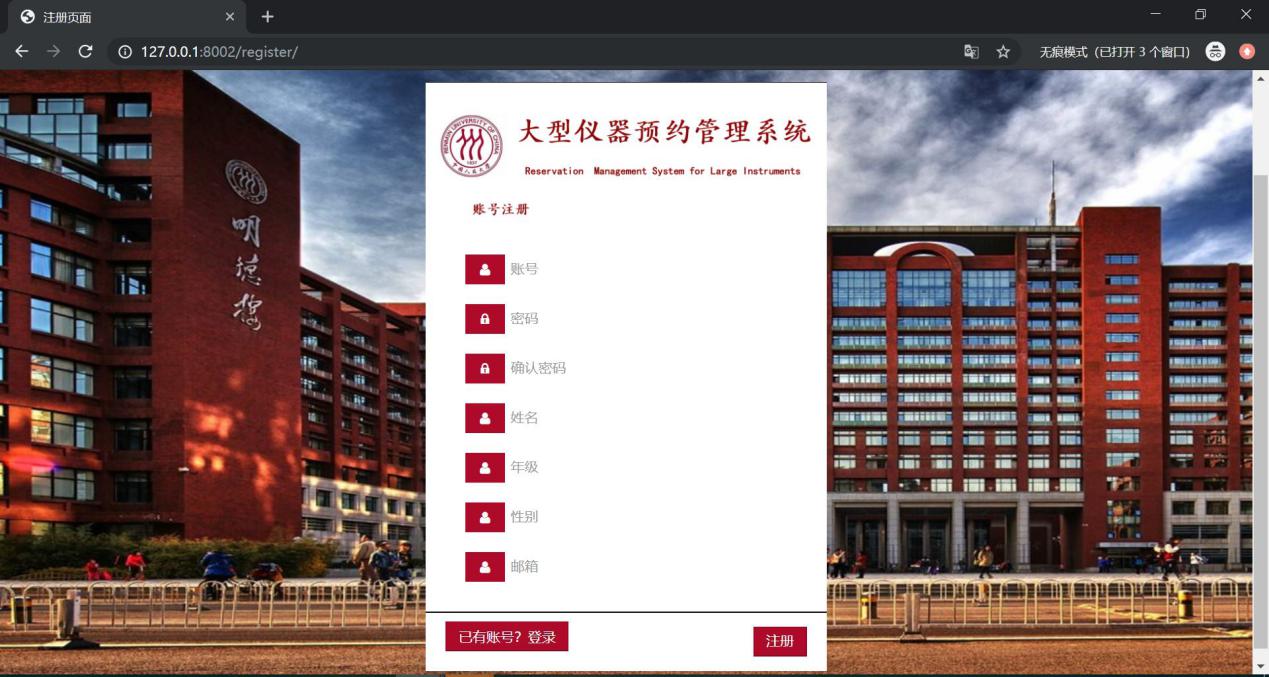
1. 登陆：期待输出登陆页面，可以输入账号以及密码，身份验证成功后根据用户信息分别跳转到对应的主页，否则，网页提醒“账号或密码输入错误”。测试成功。

测试截图：



1. 注册：期待输出注册页面，

测试截图：

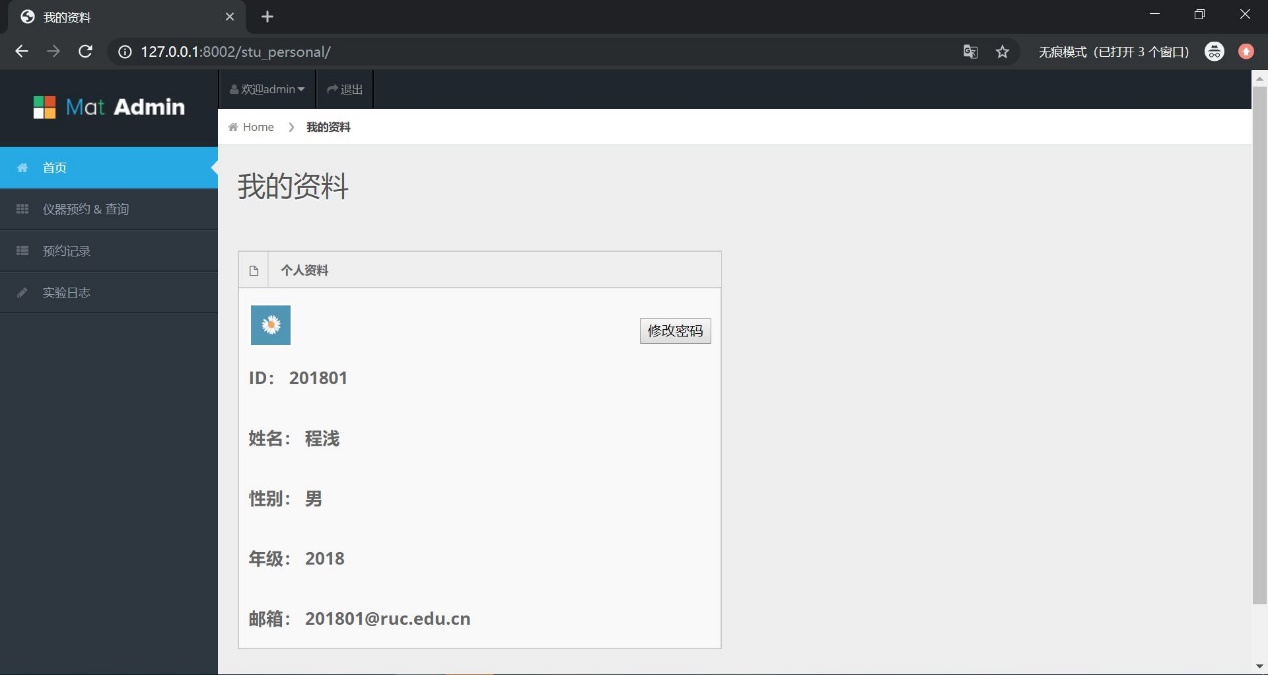


1. **模块：**学生

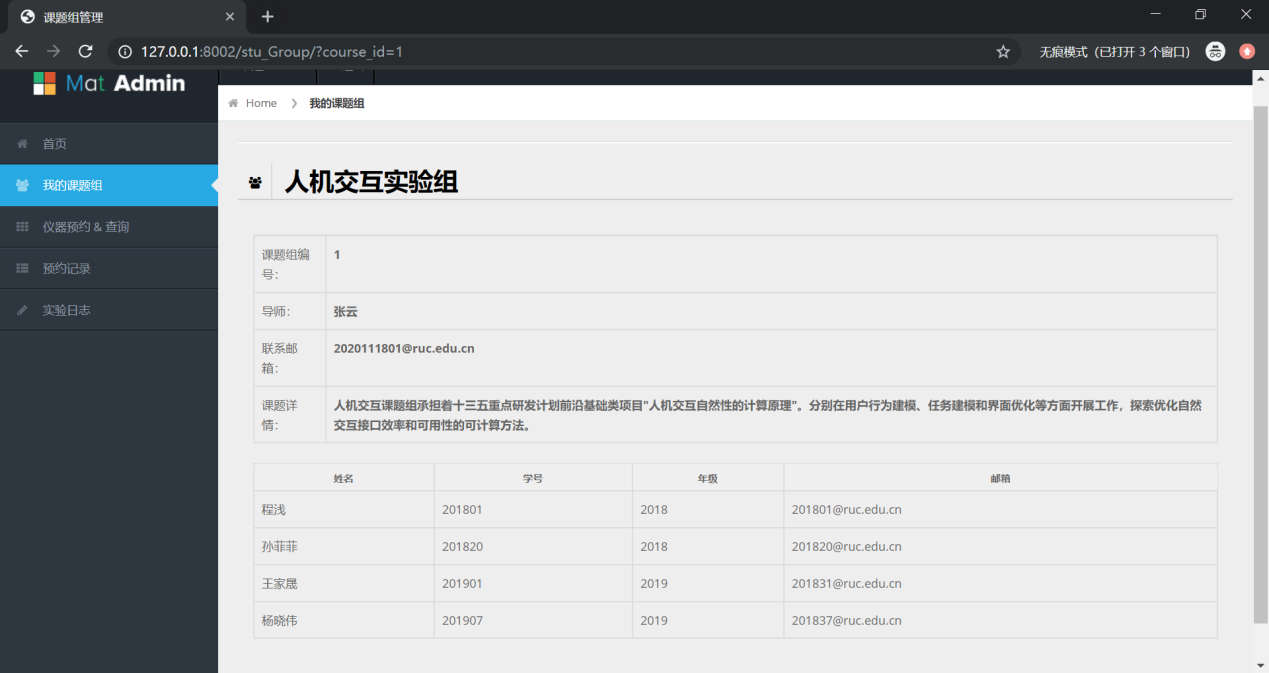
**测试名称：**验证用户身份为学生，进行资料查看，课题组查看，仪器查询，仪器日志查询，仪器详情查询，仪器预约，预约记录查看的功能

**测试目的：**验证账号为“201801”的学生查看个人资料以及课题组详情，并对编号为“10001”的眼动仪进行预约。

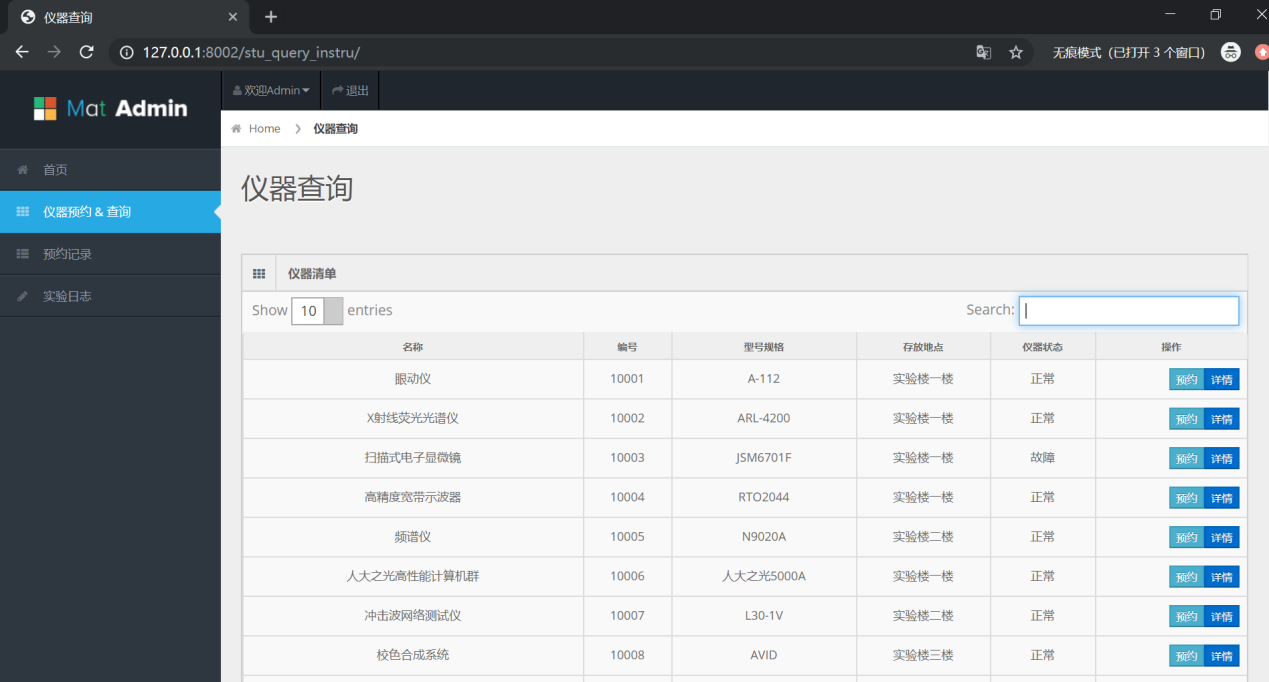
1. 个人资料查看：期待输出个人资料。测试成功

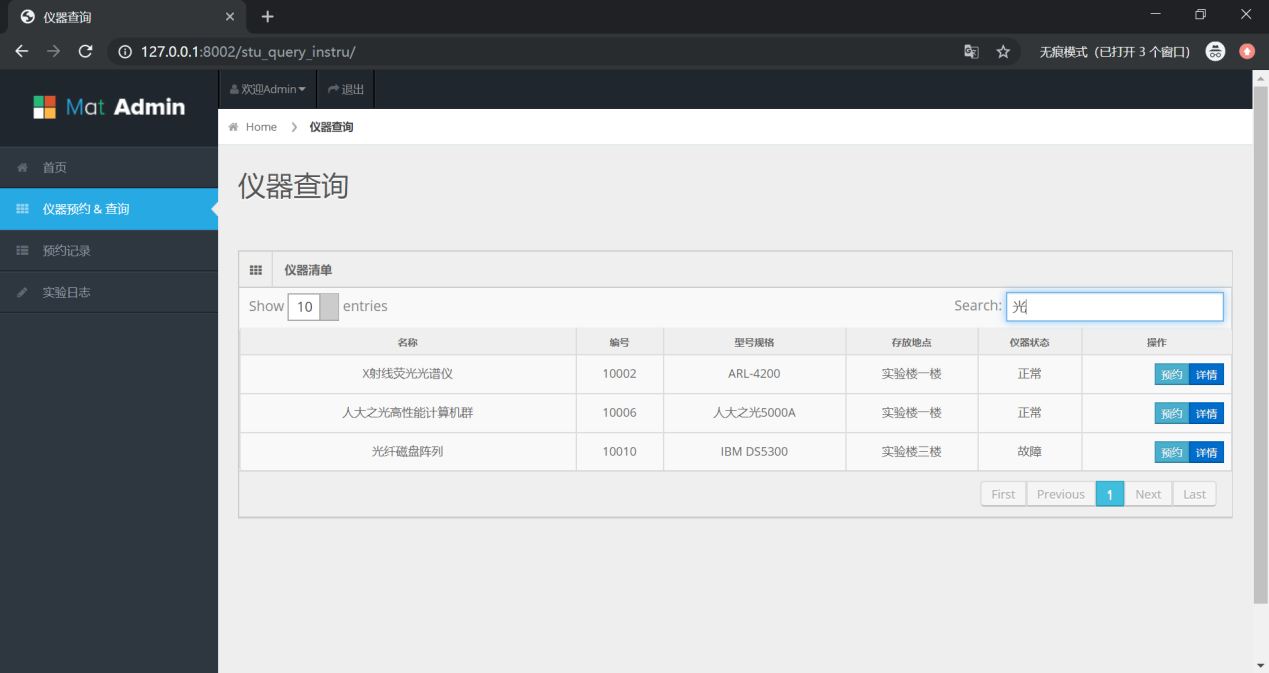


1. 课题组查看：期待输出加入的各个课题组详情，并显示课题组成员，测试成功。

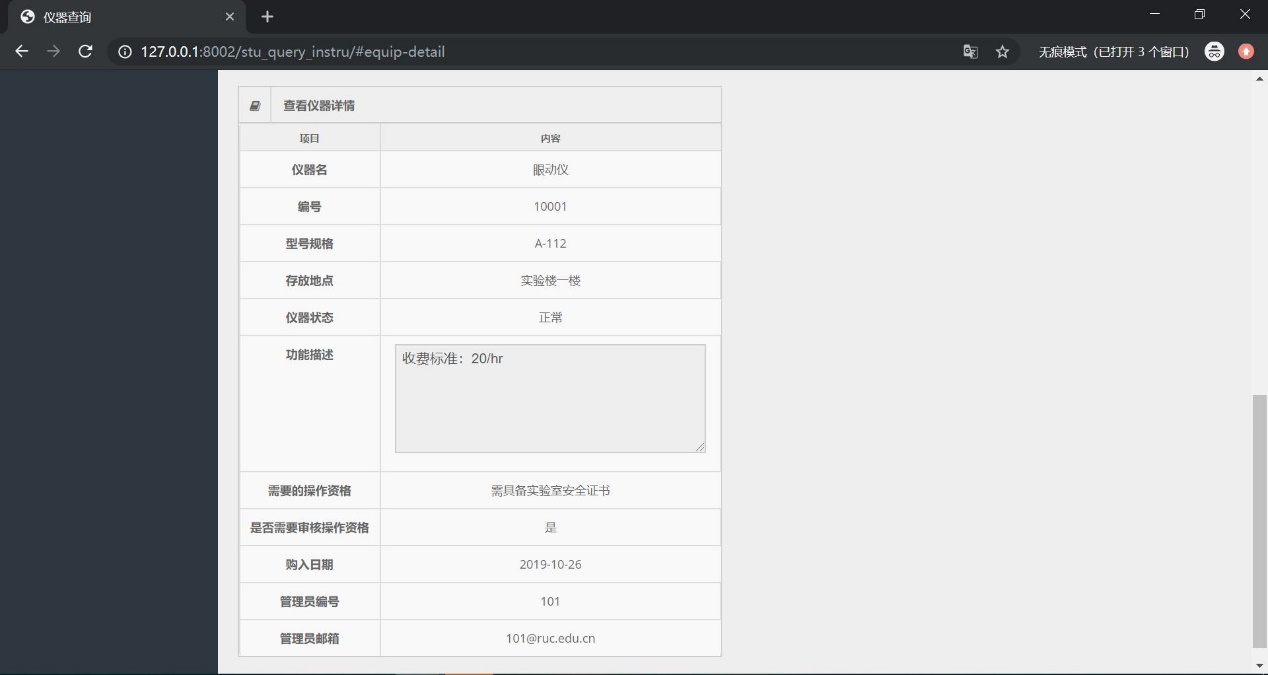


1. 仪器查询：进入页面显示所有仪器，期待输入仪器关键词，检索到相应仪器。测试成功。

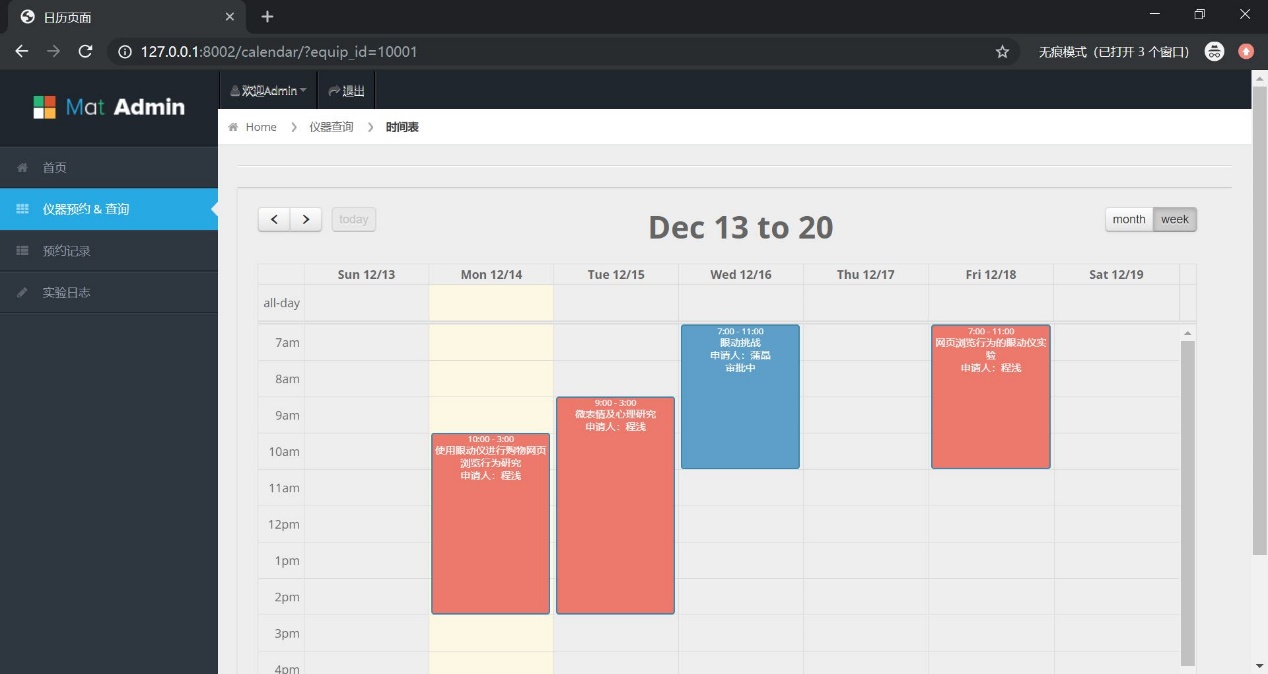


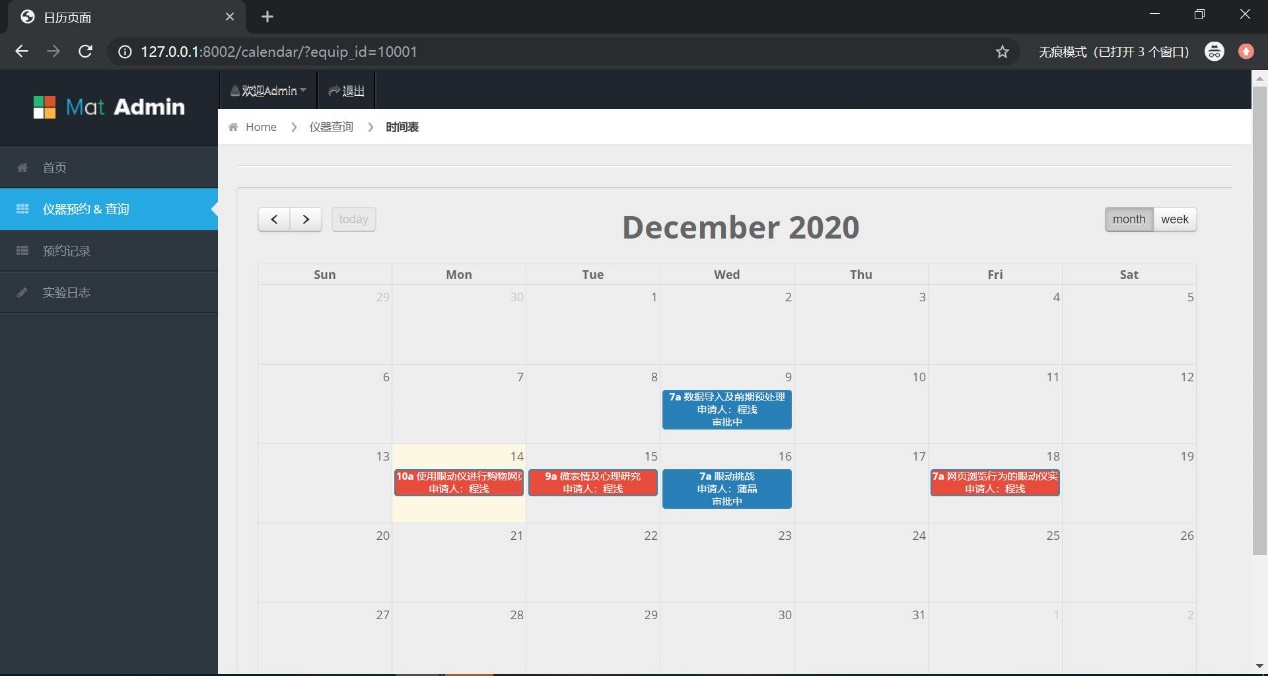


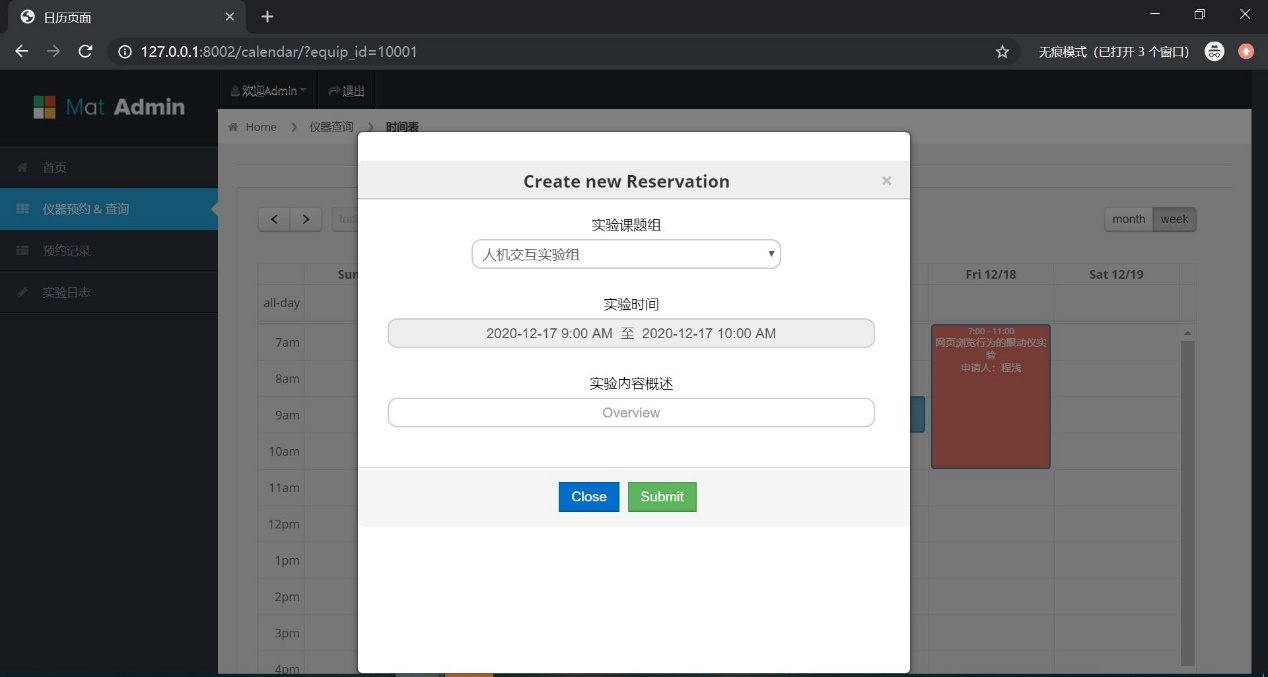
1. 仪器详情查看：期待点击仪器详情按钮后，输出仪器详情表单。测试成功。



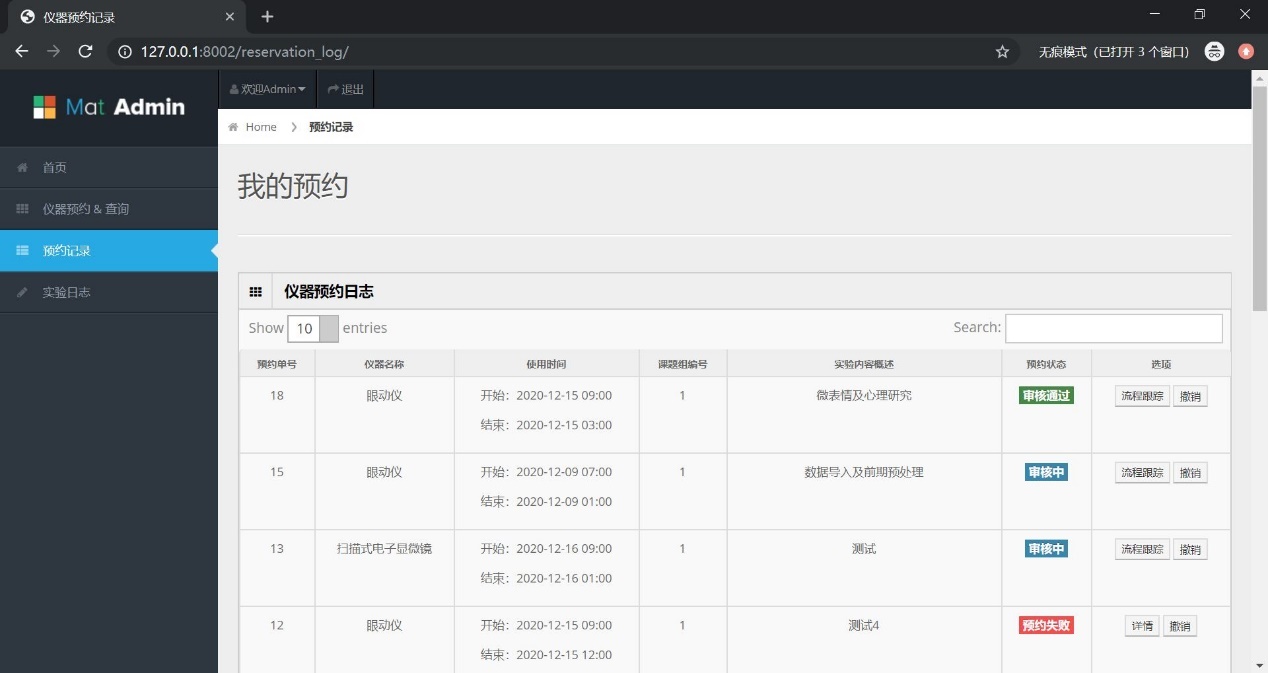
1. 仪器预约：期待点击仪器预约按钮后，输出仪器预约的周表和月表，并可以选择没有被占用的时间进行仪器预约，若有该仪器的使用资格，填表预约；否则，提示“没有该仪器的使用资格，请联系仪器管理员进行资格导入”。测试成功。

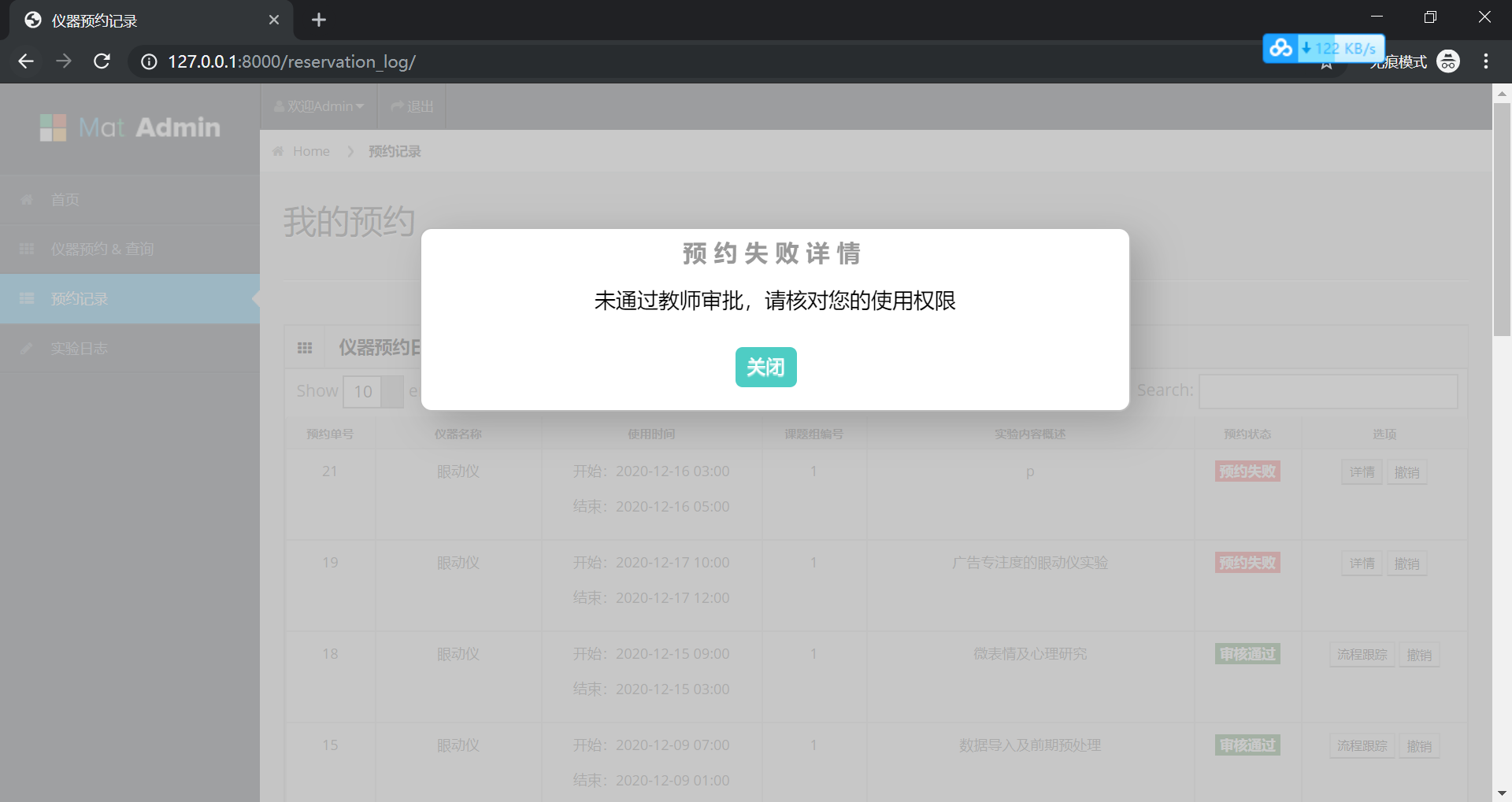


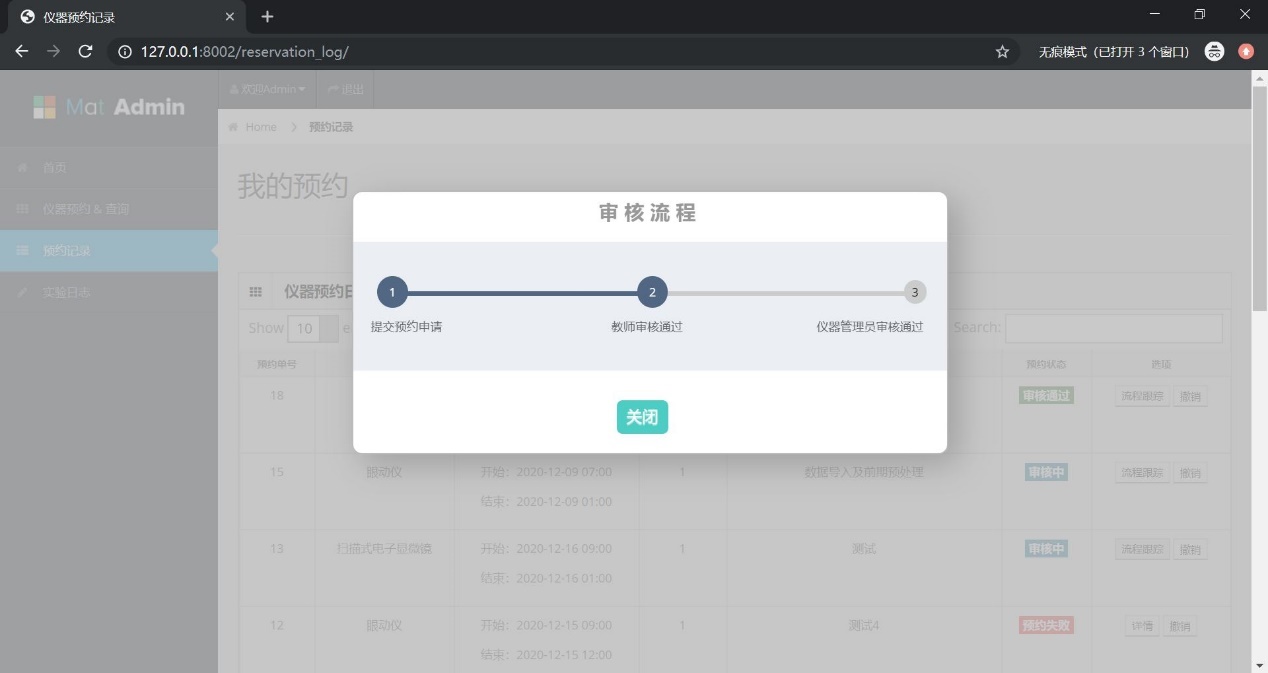




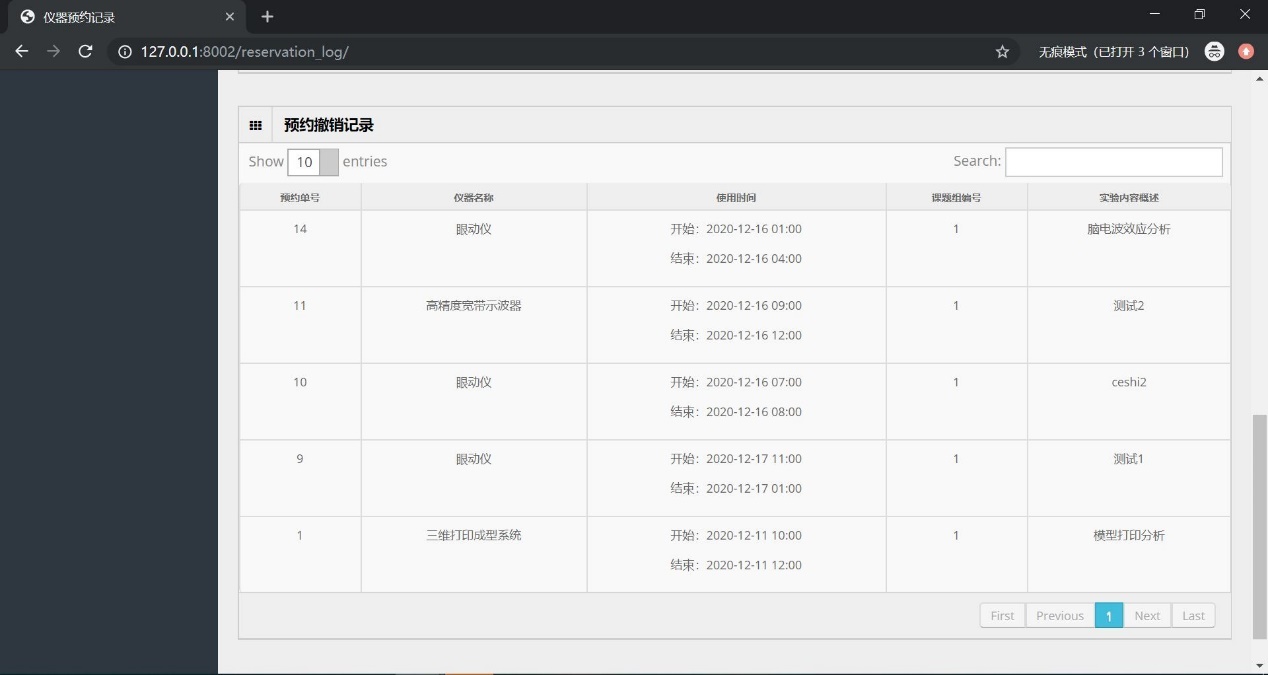
1. 预约记录查看：期待输出本人仪器预约记录，点击“流程跟踪”按钮显示该条预约的状态；点击“撤回”按钮撤回该条消息。测试成功。



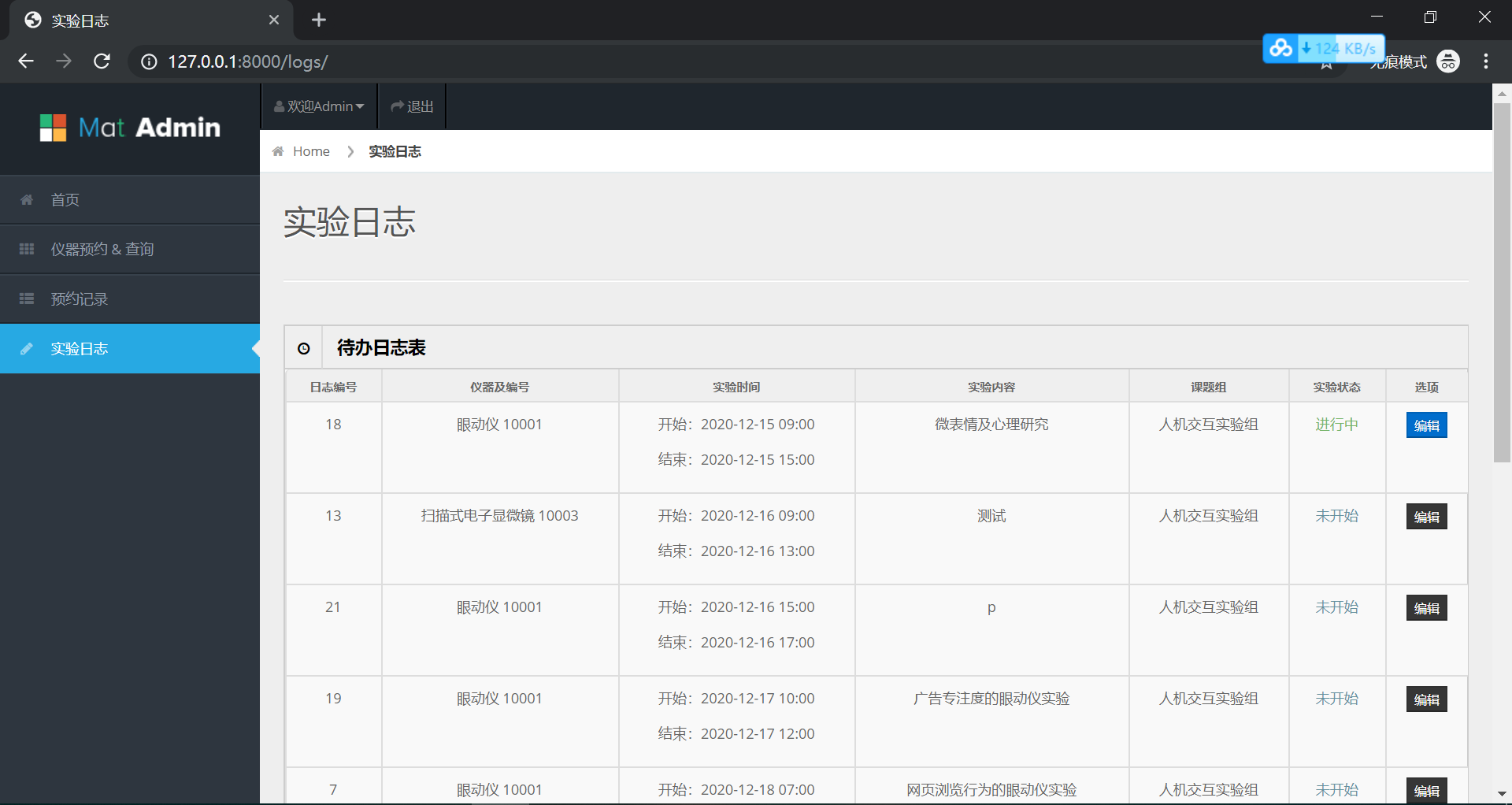


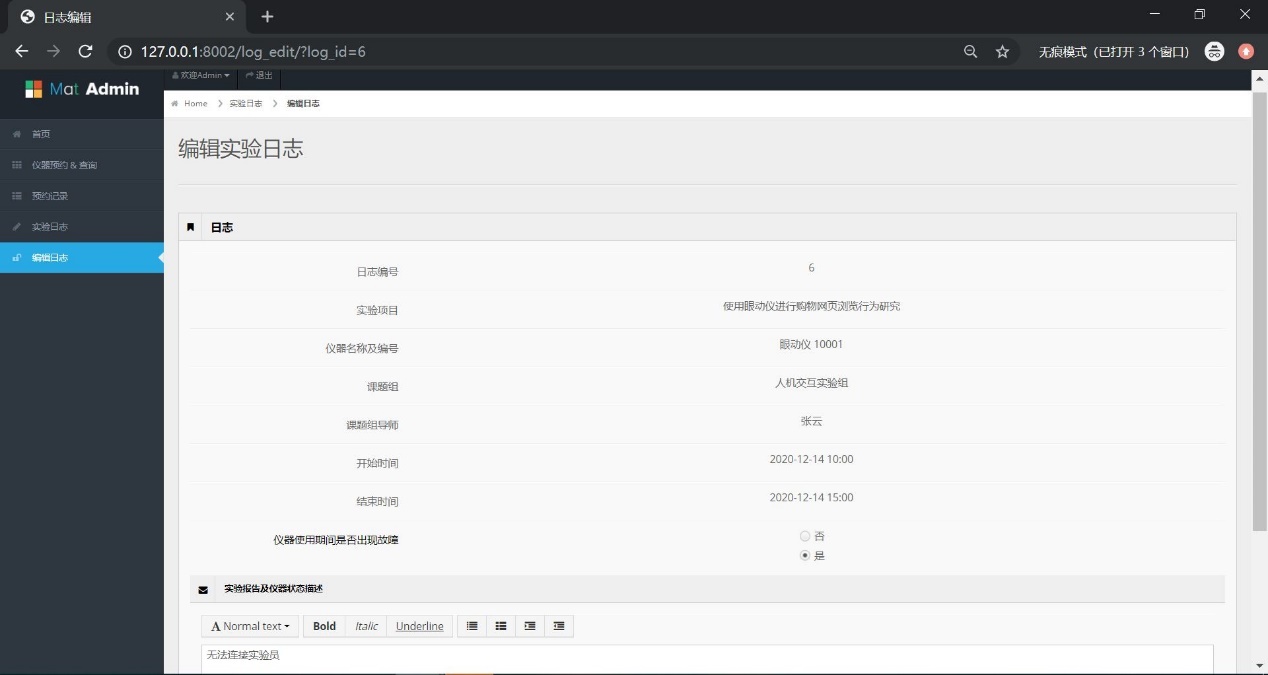


1. 查看撤回的预约：期待输出已撤回的预约及其详情。测试成功

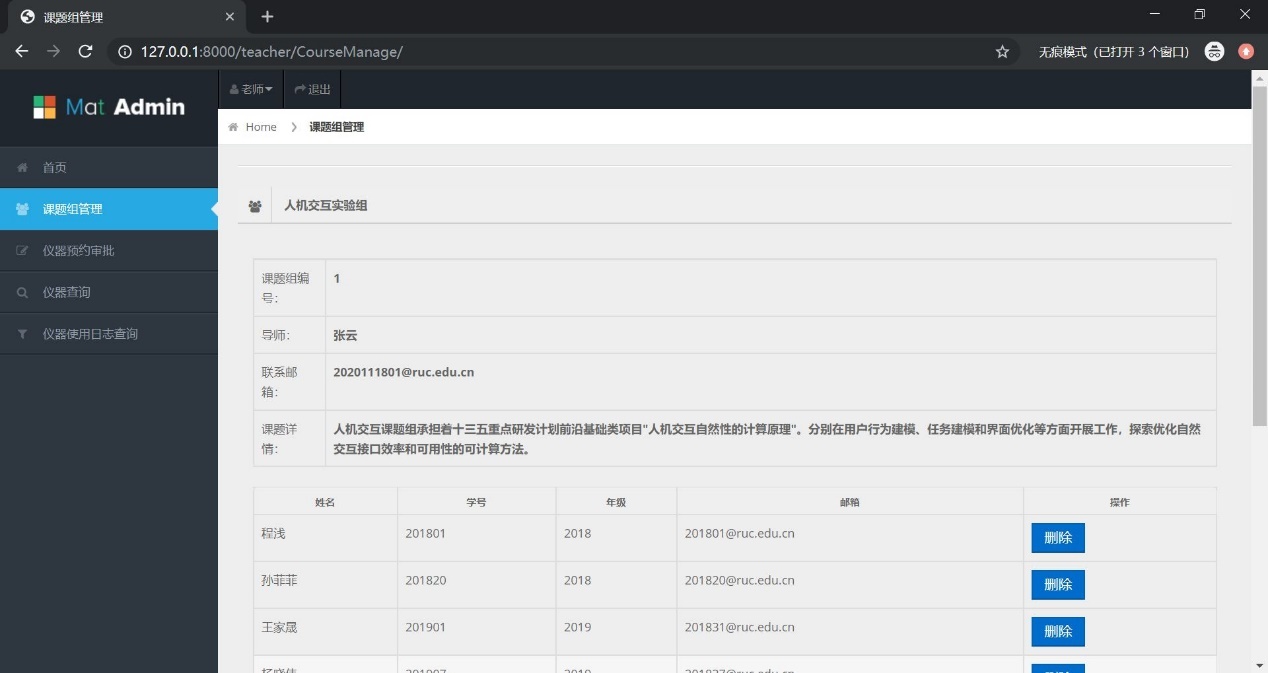


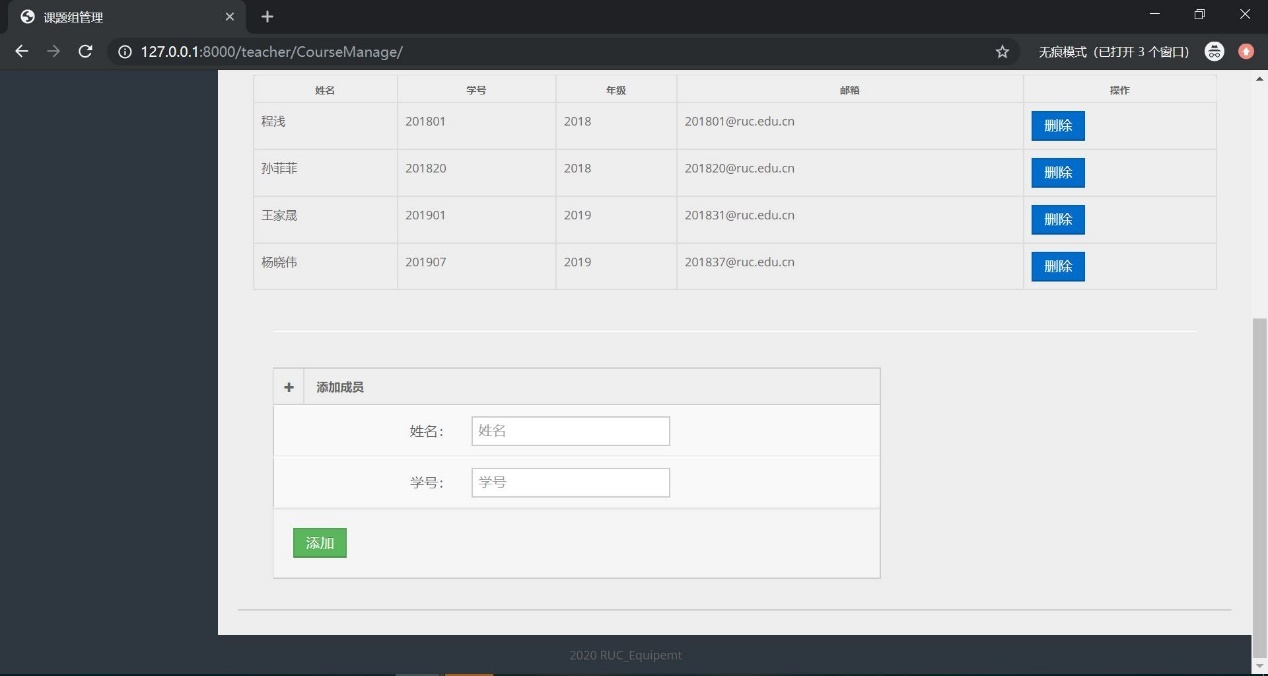
1. 管理实验日志：期待可以1）查看已提交实验日志，2）编辑并提交已进行的实验的实验日志，包括实验后仪器的状态，以及实验细节。测试成功。





1. **模块：**教师
   1. 管理课题组：期待1）输出课题组详情，2）输出课题组成员信息 3）对课题组成员进行删改。测试成功。

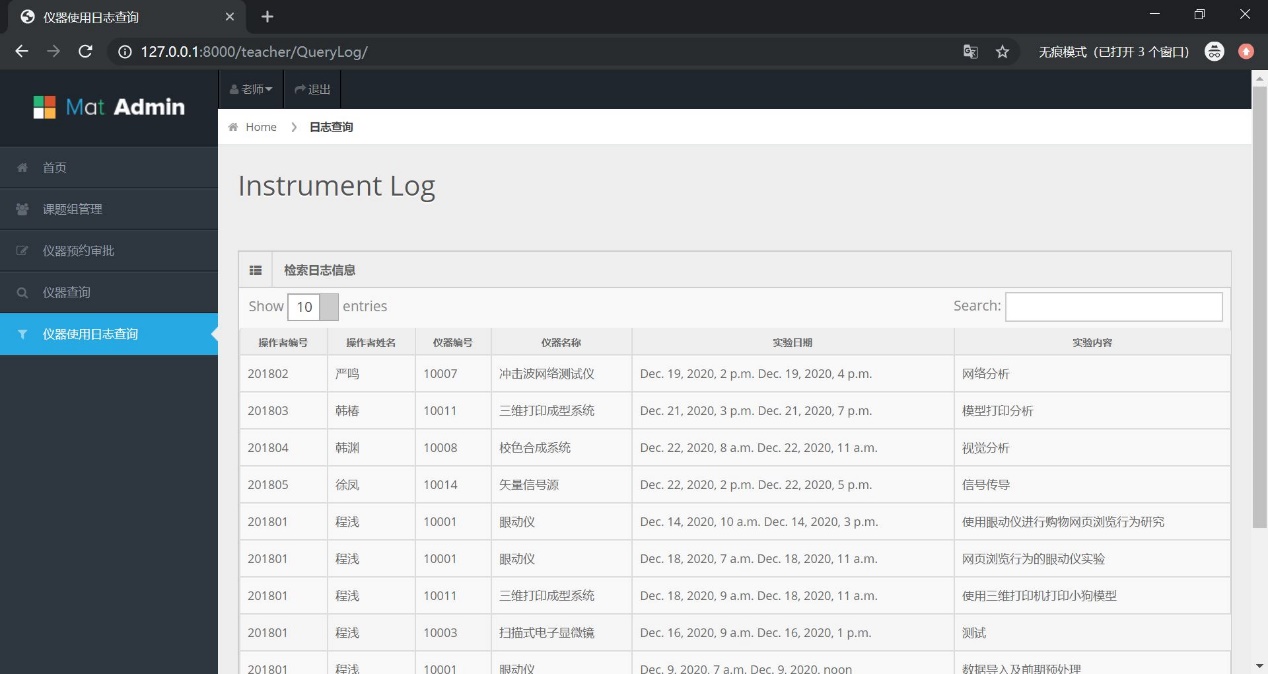




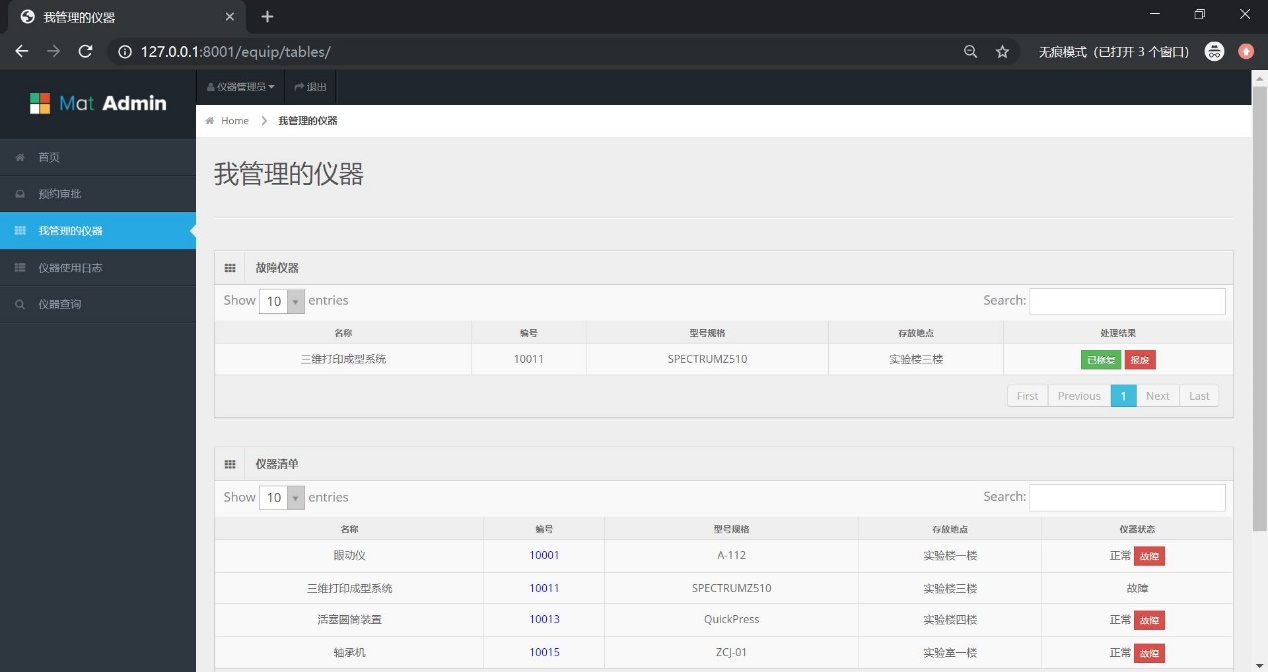
* 1. 预约审批：期待1）显示本课题组下成员的预约，点击“通过”按钮通过该预约，点击“驳回”按钮进行驳回；2）显示已完成的审批，并标明该预约是通过或是驳回。测试成功



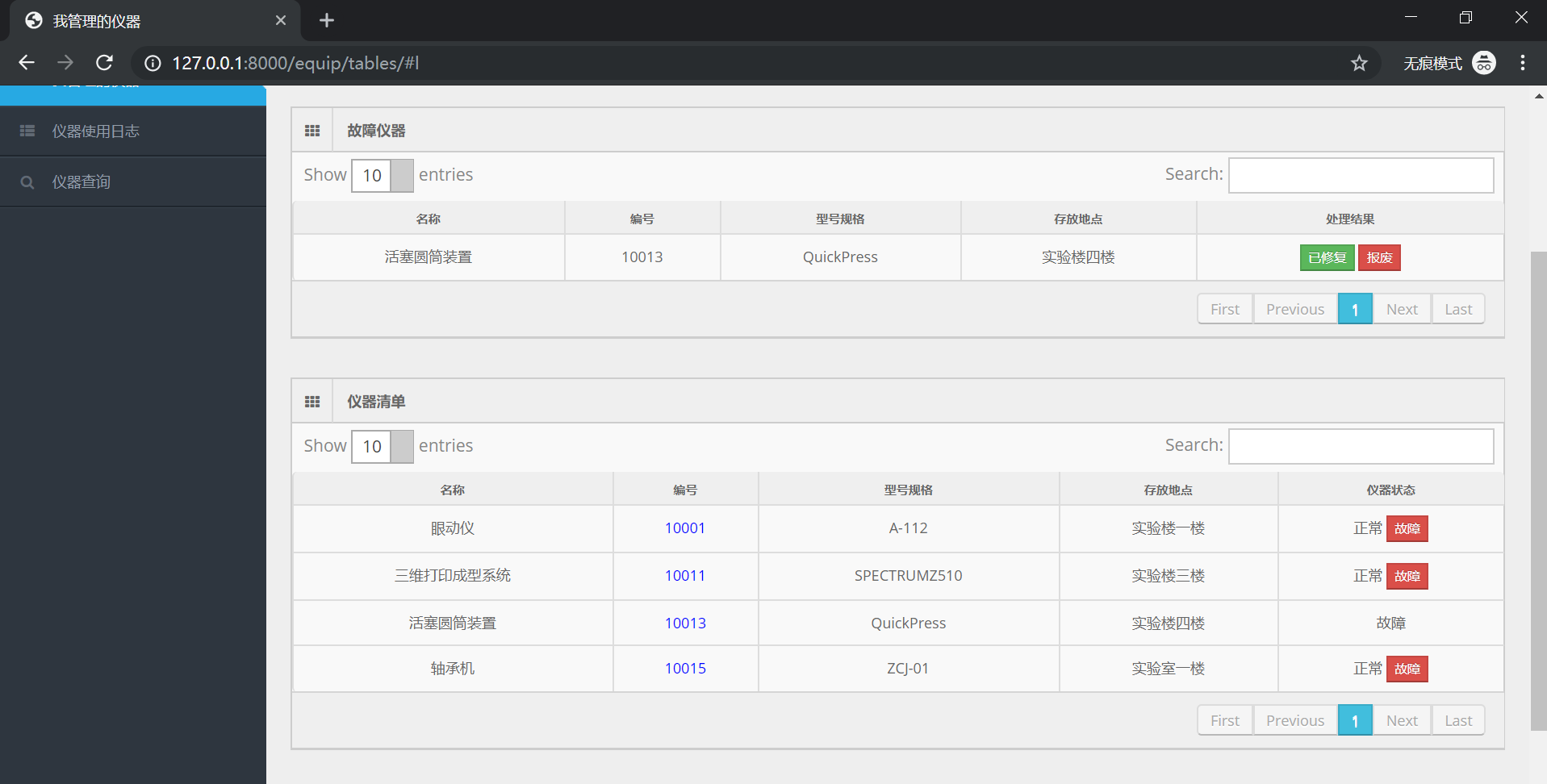
* 1. 仪器查询：期待可以查询所有仪器。测试成功，与学生模块仪器查询相同。
  2. 日志查询;：期待可以查询本课题组下所有成员的实验日志。测试成功。



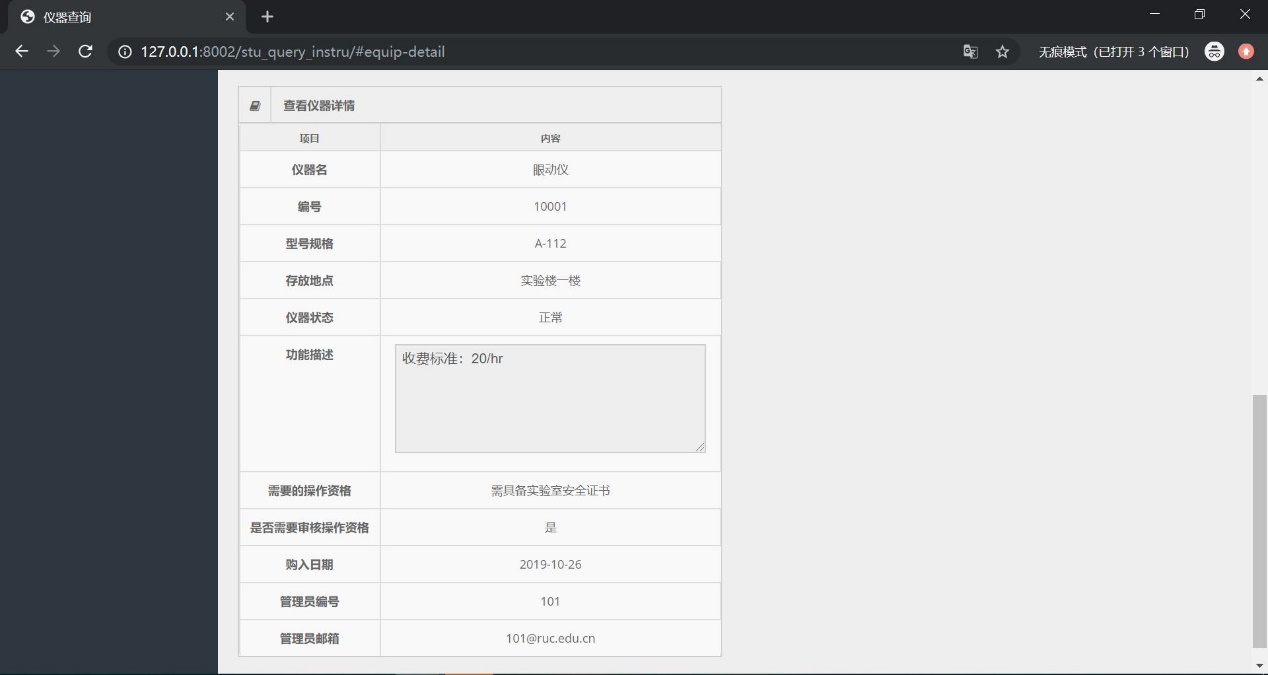
1. **模块：**仪器管理员
   1. 维护仪器：期待可以显示故障仪器，点击“已修复”使仪器状态正常， 可以进行预约和使用；点击“报废”将仪器永久删除。测试成功。



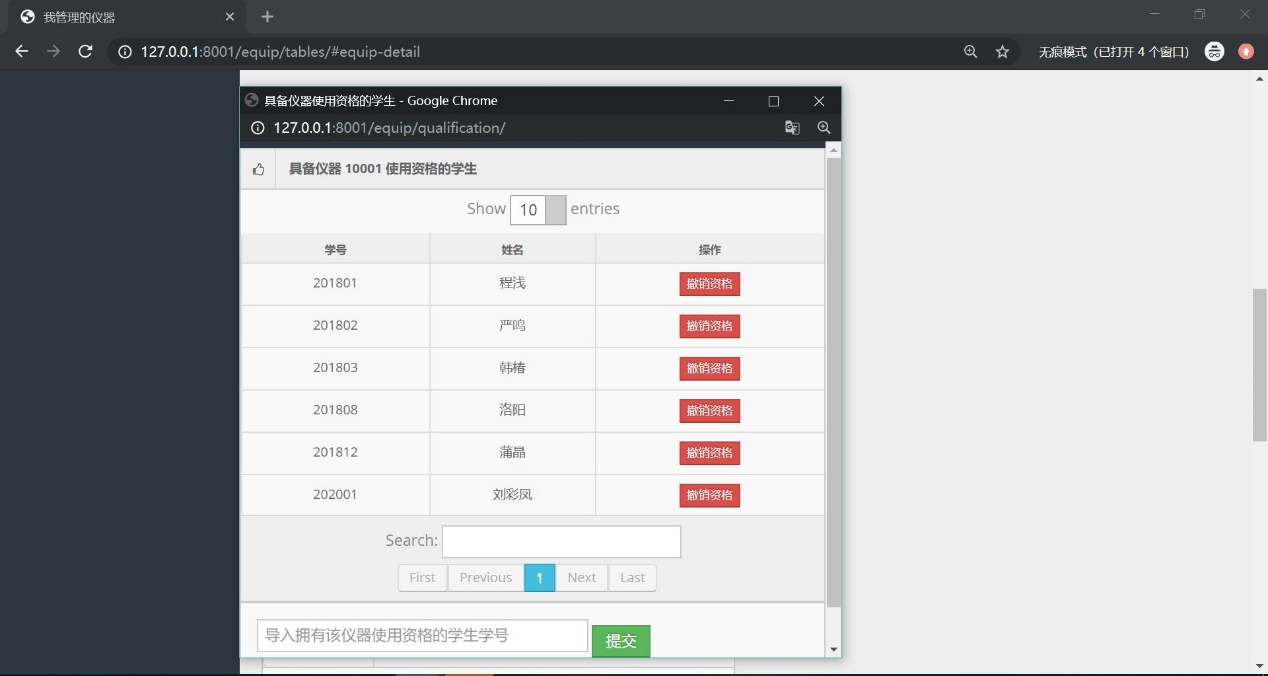
* 1. 显示仪器清单：期待1）显示仪器清单；2）点击正常仪器的“故障”按钮，可以将该仪器的状态变为故障；



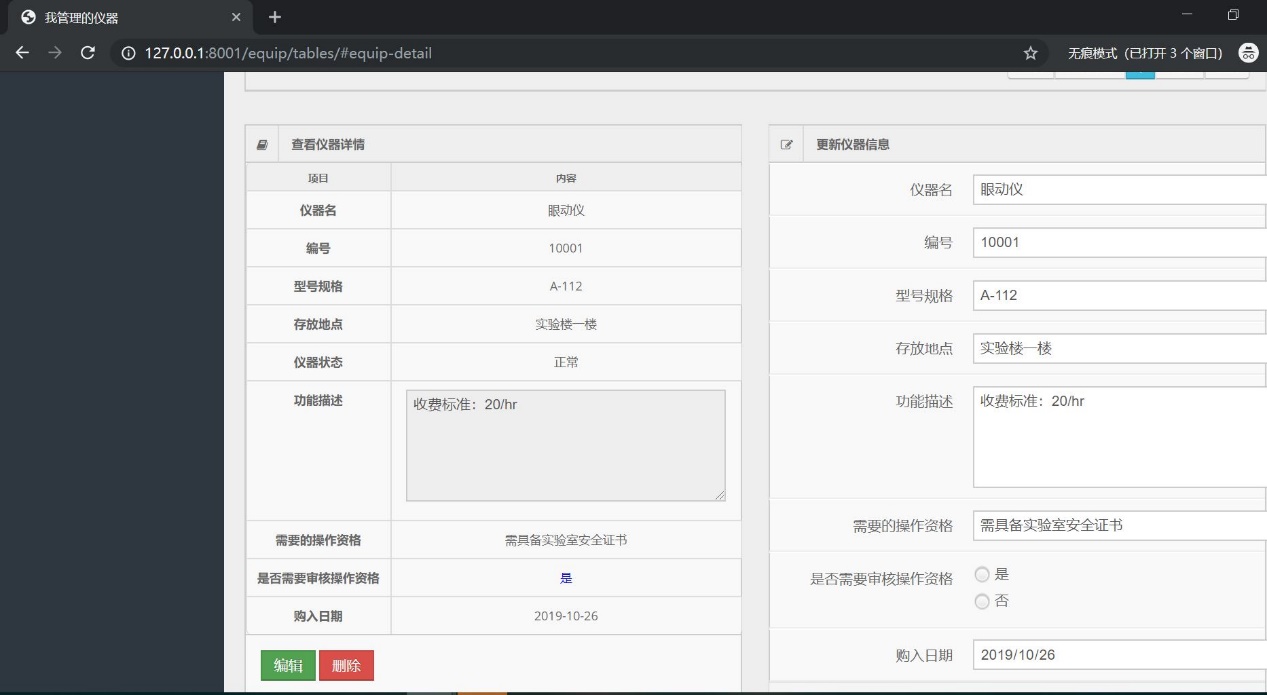
3）点击仪器编号展示仪器详情；



4）若该仪器需要操作资格，点击蓝字可以进行使用资格权限的导入与删除



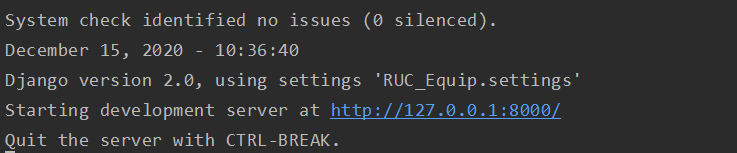
5）可以对自己管理的仪器进行增删改。实验成功



* 1. 预约审批：期待显示待审批的预约清单，点击“通过”按钮通过该条预约，点击“驳回”按钮进行预约驳回。
  2. 查看审批记录：期待显示已审批的预约清单，并显示预约与驳回的状态。



1. **安装及部署说明**
2. 对压缩包database\_project.zip进行解压。
3. 在/RUC\_Equip路径下配置Django环境，Django版本大于2.0。
4. 在mysql中导入new.sql,将创建一个名为equipment的数据库。
5. 执行python manage.py migrate 以及 python manage.py runserver



1. 进入<http://127.0.0.1:8000/homepage>到达本系统主页。