大作业总结报告

1. 实现环境

我们的医院管理系统采用前后端分离的架构。后端使用Python/FastAPI 提供 API 与SQL数据库操作，前端用Vue构建界面。

具体而言，我们使用 `uvicorn` 热重载服务、`aiomysql` 连接 MySQL、`python-dotenv` 管理`passlib[bcrypt]` 与 `python-jose` 做认证，外加 `openpyxl` 处理表格。整体来看是典型的异步 REST API 结构。

2. 系统功能结构图

2.1 总体功能架构

```
医院管理系统（HMS）
├── 用户认证与权限管理
├── 患者服务门户
├── 医生工作站
├── 药品管理系统
├── 护士工作站
├── 管理员功能模块
└── 公共模块
```

2.2 详细功能模块分解

2.2.1 用户认证与权限管理模块

```
用户认证与权限管理
├── 登录系统
│   ├── 患者登录
│   ├── 医生登录
│   ├── 护士登录
│   ├── 药师登录
│   └── 管理员登录（院长）
├── 用户注册
│   ├── 患者注册
│   └── 管理员分配医生/护士/药师账号
└── 修改信息
    ├── 修改个人信息
    └── 修改密码
```

2.2.2 患者服务门户模块

```
患者服务门户
├── 注册账号
├── 修改信息
├── 挂号申请/取消
├── 缴费
├── 查询功能
```

```
|   ├── 挂号信息查询
|   ├── 缴费记录查询
|   ├── 检查结果查询
|   ├── 主治医师查询
|   ├── 病历查询
|   ├── 处方查询
|   └── 住院记录查询
├── 缴费提醒
└── 就诊提醒
```

2.2.3 医生工作站模块

```
医生工作站
├── 接受挂号申请
├── 查询患者信息
├── 转院处理
├── 填写病历处方
├── 安排检查
├── 查询药品信息
├── 住院安排
└── 查询功能
    ├── 患者历史病历查询
    ├── 患者信息查询
    └── 药品信息查询
```

2.2.4 药品管理系统模块

```
药品管理系统
├── 发药功能
├── 更新库存
└── 查询功能
    ├── 药品使用情况查询
    └── 药品库存信息查询
```

2.2.5 护士工作站模块

```
护士工作站
├── 患者护理
├── 查询功能
    ├── 患者处方查询
    ├── 住院信息查询
    ├── 病房信息查询
    └── 护士排班查询
└── 今日待办
    └── 报警系统
```

2.2.6 护士长管理模块

```
 护士长管理
├── 办理出院证明
├── 管理护士排班
└── 查询功能
    ├── 排班信息查询
    ├── 病人处方查询
    ├── 住院信息查询
    └── 病房信息查询
```

2.2.7 管理员功能模块

```
 管理员功能
├── 注册/注销账号
|    ├── 医生账号管理
|    ├── 护士账号管理
|    └── 药师账号管理
└── 查询功能
    ├── 账号信息查询
    ├── 科室查询
    ├── 病房查询
    └── 营收查询
```

2.2.8 公共模块

```
公共模块
├── 用户与认证
|    └── 账号、角色、权限校验
├── 审计日志
|    ├── 请求链路
|    └── 操作记录入库
├── 数据模型
├── 触发器/自动结算
└── 配置与环境
    ├── 数据库连接
    └── 初始化默认数据
```

3. 基本表的定义，主外码等完整性约束定义，索引的定义

3.1 基本表及其主外码定义

useraccount

- Attributes: `phone`, `username`, `password_hash`, `role`, `register_time`, `status`
- PK: `phone`
- FK: —

department

- Attributes: `dept_id`, `dept_name`, `telephone`

- PK: `dept_id`
- FK: —

doctor

- Attributes: `doctor_id`, `name`, `gender`, `title`, `phone`, `dept_id`
- PK: `doctor_id`
- FK: `phone` → useraccount.phone； `dept_id` → department.dept_id

nurse

- Attributes: `nurse_id`, `name`, `gender`, `phone`, `is_head_nurse`
- PK: `nurse_id`
- FK: `phone` → useraccount.phone

patient

- Attributes: `patient_id`, `name`, `gender`, `birth_date`, `id_number`, `address`, `phone`
- PK: `patient_id`
- FK: `phone` → useraccount.phone

ward

- Attributes: `ward_id`, `dept_id`, `type`, `bed_count`
- PK: `ward_id`
- FK: `dept_id` → department.dept_id

nurseschedule

- Attributes: `schedule_id`, `nurse_id`, `ward_id`, `start_time`, `end_time`
- PK: `schedule_id`
- FK: `nurse_id` → nurse.nurse_id； `ward_id` → ward.ward_id

registration

- Attributes: `reg_id`, `patient_id`, `doctor_id`, `reg_type`, `reg_date`, `visit_date`, `status`, `fee`
- PK: `reg_id`
- FK: `patient_id` → patient.patient_id； `doctor_id` → doctor.doctor_id

medicalrecord

- Attributes: `record_id`, `reg_id`, `create_time`, `complaint`, `diagnosis`, `suggestion`
- PK: `record_id`
- FK: `reg_id` → registration.reg_id

examination

- Attributes: `exam_id`, `record_id`, `type`, `result`, `date`, `fee`
- PK: `exam_id`

- FK: `record_id` → medicalrecord.record_id

medicine

- Attributes: `medicine_id`, `name`, `price`, `stock`, `unit`, `expire_date`
- PK: `medicine_id`
- FK: —

prescription

- Attributes: `pres_id`, `record_id`, `create_time`, `total_amount`
- PK: `pres_id`
- FK: `record_id` → medicalrecord.record_id

prescriptiondetail

- Attributes: `detail_id`, `pres_id`, `medicine_id`, `quantity`, `usage`
- PK: `detail_id`
- FK: `pres_id` → prescription.pres_id； `medicine_id` → medicine.medicine_id

hospitalization

- Attributes: `hosp_id`, `record_id`, `ward_id`, `hosp_doctor_id`, `status`, `in_date`, `out_date`, `deposit_amount`
- PK: `hosp_id`
- FK: `record_id` → medicalrecord.record_id； `ward_id` → ward.ward_id； `hosp_doctor_id` → doctor.doctor_id

nursetask

- Attributes: `task_id`, `hosp_id`, `type`, `time`, `status`, `detail`, `medicine_snapshot`, `service_fee`, `created_by`
- PK: `task_id`
- FK: `hosp_id` → hospitalization.hosp_id； `created_by` → doctor.doctor_id

payment

- Attributes: `payment_id`, `type`, `amount`, `status`, `time`, `patient_id`, `reg_id`, `pres_id`, `exam_id`, `hosp_id`
- PK: `payment_id`
- FK: `patient_id` → patient.patient_id； `reg_id` → registration.reg_id； `pres_id` → prescription.pres_id； `exam_id` → examination.exam_id； `hosp_id` → hospitalization.hosp_id

3.2 索引定义

索引的本质是一种**优化查询效率的数据结构**，其核心意义是通过牺牲少量的存储空间和数据更新性能，换取查询效率的数量级提升。我们的基本表设计采取了**严格的范式设计**，清除了已知的全部依赖，因此存在大量的关联查询。因此，索引的存在是关键且必要的。在此，我们罗列出**除了主键索引之外**的其他索引。

useraccount

- index: `username`

department

- index: `dept_name` **UNIQUE**

doctor

- index: `dept_id`

nurse

- index: `phone`

patient

- index: `phone` **UNIQUE**、`id_number` **UNIQUE**

ward

- index: `dept_id`

nurseschedule

- index: `nurse_id`、`ward_id`

registration

- index: `doctor_id`、`patient_id`

medicalrecord

- index: `reg_id` **UNIQUE**

examination

- index: `record_id`

medicine

- index: `name` **UNIQUE**

prescription

- index: `record_id` **UNIQUE**

prescriptiondetail

- index: `medicine_id`、`pres_id`

hospitalization

- index: `uq_hosp_active_record_id` **UNIQUE**、`hosp_doctor_id`、`record_id`、`ward_id`

nursetask

- index: `hosp_id`

payment

- index: `exam_id`、`hosp_id`、`patient_id`、`pres_id`

4. 系统的安全性设计，不同人员的外模式及相关权限

4.1 安全性设计

- 登录/注册：注册时使用 `get_password_hash()` 保存 bcrypt 摘要，登录时 `verify_password()` 校验，再由 `create_access_token()` 生成 30 分钟有效的 HS256 JWT，Token 载荷含手机号 `sub` 与 `role` 字段。
- 认证中间件 `OAuth2PasswordBearer` 解码 JWT 并从 `sub` 解析手机号，失效或伪造 Token 直接 401；`get_current_user()` 基于手机号二次查询数据库确保账号仍存在、状态有效。
- 账号状态控制：登录流程会检查 `UserAccount.status`，被禁用的账号阻断登录。
- 权限由依赖注入强制：如 `get_current_admin_user()` 仅允许 `UserRole.ADMIN` 访问管理员接口，deps.py；医生、护士、药师各自有专门的 `Depends` 校验，详见下方外模式部分。
- 数据层约束（唯一键、外键）防止越权串改他人档案，例如 `Patient.phone` 同时唯一且外键指向账号，实现登录身份与患者档案的一一绑定。

4.2 外模式与权限

- **管理员外模式**：`/api/*` 下的大部分后台管理能力（批量导入医护账号、科室与病房维护、账单统计等）都要求 `Depends(get_current_admin_user)`，只对 `UserRole.ADMIN` 暴露。管理员还能创建/禁用医生、护士、药师账号，控制其登录入口。
- **患者外模式**：`/api` 路由下的患者接口依赖 `get_current_user_phone`，通过 Token 中手机号定位唯一档案，允许自助维护个人资料、挂号、查看病历、缴费状态等。少数接口（按患者 ID 查询）进一步要求 `UserRole.ADMIN` 或 `UserRole.DOCTOR` 才能访问。
- **医生外模式**：`/api/doctor/*` 接口统一依赖 `get_current_doctor()`，不仅验证 Token，还校验 `UserAccount.role == DOCTOR` 且存在医生档案，医生可管理排班、接诊、病历、检查、住院与护士任务等数据。
- **护士外模式**：`/api/nurse/*` 通过 `get_current_nurse()` 限定 `UserRole.NURSE`，普通护士仅能查看与自己排班相关的病房/任务；`get_head_nurse()` 进一步限制护士长专属操作（排班编排、全院病房视图等）。
- **药师外模式**：`/api/pharmacy/*` 中涉及库存或采购的接口都依赖 `get_current_pharmacist()`，仅 `UserRole.PHARMACIST` 可新增/采购药品、调整库存；开具处方的接口则复用医生依赖，确保只有接诊医生能生成或查看自己的处方。
- **公共接口**：`/auth` 下的注册、登录、改密等对所有用户开放。

5.存储过程、触发器和函数的代码说明

下面给出相关的示例代码及说明。

存储过程是对表**操作**的封装，函数是对数据**计算**的封装。我们在本项目中并未刻意使用这两类设计。

触发器：挂号完成自动生成缴费项目

```
CREATE TRIGGER trg_registration_completed
    AFTER UPDATE ON registration
    FOR EACH ROW
    BEGIN
        IF OLD.status <> 'FINISHED' AND NEW.status = 'FINISHED' THEN
            INSERT INTO payment(type, amount, patient_id, reg_id, pres_id, time)
```

```
            SELECT 'PRESCRIPTION', pres.total_amount, NEW.patient_id, NEW.reg_id,
pres.pres_id, NOW()
            FROM prescription pres
            JOIN medicalrecord mr ON mr.record_id = pres.record_id
            WHERE mr.reg_id = NEW.reg_id;

            INSERT INTO payment(type, amount, patient_id, reg_id, exam_id, time)
            SELECT 'EXAM', FLOOR(RAND() * 151) + 50, NEW.patient_id, NEW.reg_id,
exam.exam_id, NOW()
            FROM examination exam
            JOIN medicalrecord mr ON mr.record_id = exam.record_id
            WHERE mr.reg_id = NEW.reg_id;
        END IF;
    END;
```

6.实现过程中主要技术和主要模块的论述

主要技术：

Bcrypt：

一种专门为密码存储设计的密码哈希函数（Password Hashing Function）。它由 Niels Provos 和 David Mazières 在 1999 年基于 Blowfish 加密算法设计而成。内置Halt，适应强，十分方便。

HS256：

一种极为常见的数字签名算法。当服务器生成一个带有 HS256 签名的 JWT 时，过程如下：

1. **准备数据**：将 Header（算法信息）和 Payload（用户信息）进行 Base64 编码。
2. **拼接字符串**：将编码后的 Header 和 Payload 用点（`.`）连接。
3. **计算签名**：使用**服务器私有的密钥**对拼接后的字符串进行 HMAC-SHA256 运算。

   > 公式：
   > $$Signature = HMAC\_SHA256(Base64(Header) + "." + Base64(Payload), Secret\_Key)$$

4. **生成令牌**：将生成的签名附加在末尾，形成完整的 JWT。

当客户端带着这个令牌回来时，服务器用**同一个密钥**再次计算签名。如果计算结果一致，说明数据没有被篡改。

TCP/IP 协议栈：

在最底层的网络传输中，任何一个 TCP 连接都必须包含源 IP 地址和目的 IP 地址。同时业界使用了一个标准的HTTP请求头，X-Forwarded-For。解析其后的内容即可获取用户IP。

VUE3：

组合式API，高性能前端框架。小体积，易于上手和维护。

FastAPI：

目前 Python 生态中最受瞩目、增长最快的现代 Web 框架之一。它由 Sebastián Ramírez 于 2018 年发布，旨在解决开发效率与高性能不可兼得的痛点。性能极致，自动生成交互式文档，自动撰写校验逻辑，易于上手。

MySQL：

全球最受欢迎的开源关系型数据库管理系统（RDBMS）。可以高效选择最优的SQL执行方式。支持数据库的基本功能。

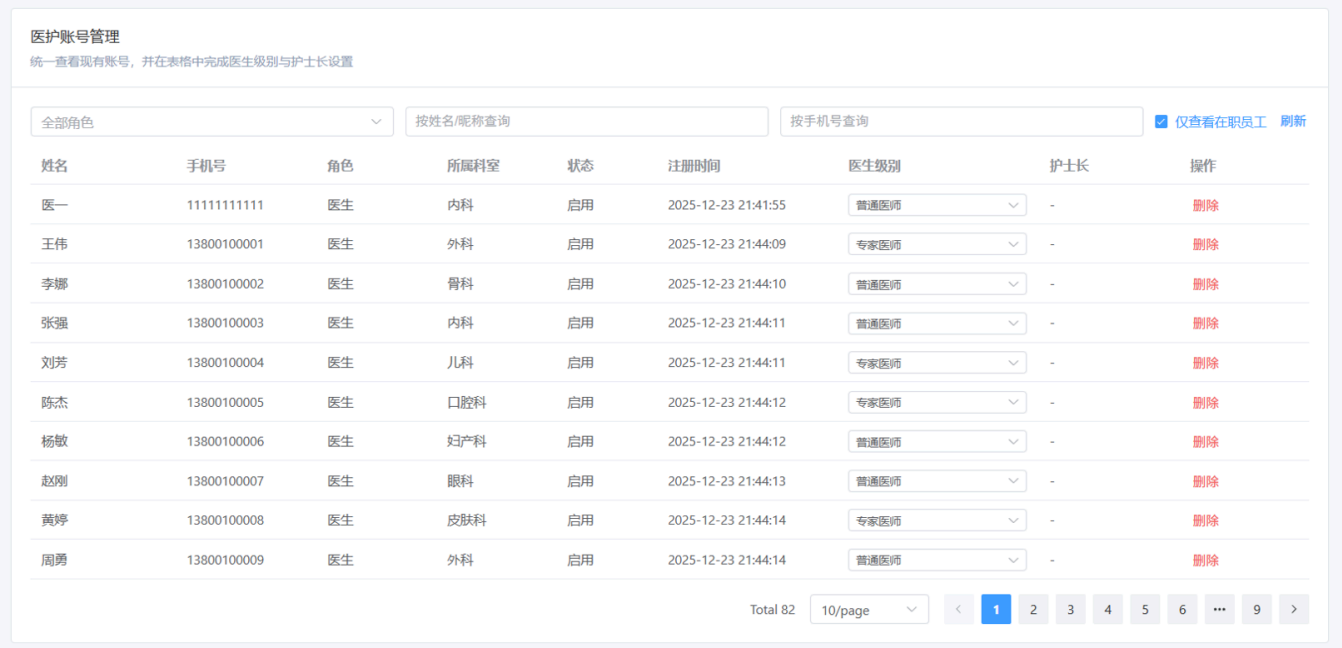
主要模块：

- 认证与审计：统一登录、角色鉴权，记录操作审计日志。
- 用户与科室：UserAccount 统一账号，Doctor/Nurse/Patient 档案，Department/Ward 管理。
- 挂号与就诊：Registration 挂号流转，MedicalRecord 病历编写，医生接诊流程。
- 处方与检查：Prescription/PrescriptionDetail 开具与查询，Examination 记录，支付关联。
- 药房与库存：Medicine 库存管理，近 30 天用量与建议补货，一键补足，低库存预警。
- 住院与护理：Hospitalization 住院管理，NurseTask 护理任务生成/提醒，护士长查看过期/即将到期任务。
- 支付与营收：Payment 统一费用记录，状态统计，院长营收看板（总额、笔数、类型占比）。
- 触发器集成：注册完成时自动生成处方/检查缴费记录，任务/库存等可选低层校验与预警。
- 文档导出：转院单生成规范版式 DOCX；数据分析图表可导出不同粒度钻取的PDF。

7.若干展示系统功能的运行实例

7.1 数据管理

7.1.1 增删改查医护帐号



7.1.2 文件批量导入数据

7.2 数据展示

7.2.1 支持多索引查询

7.2.2 基于时间排序

缴费查询
查看并完成未缴费项

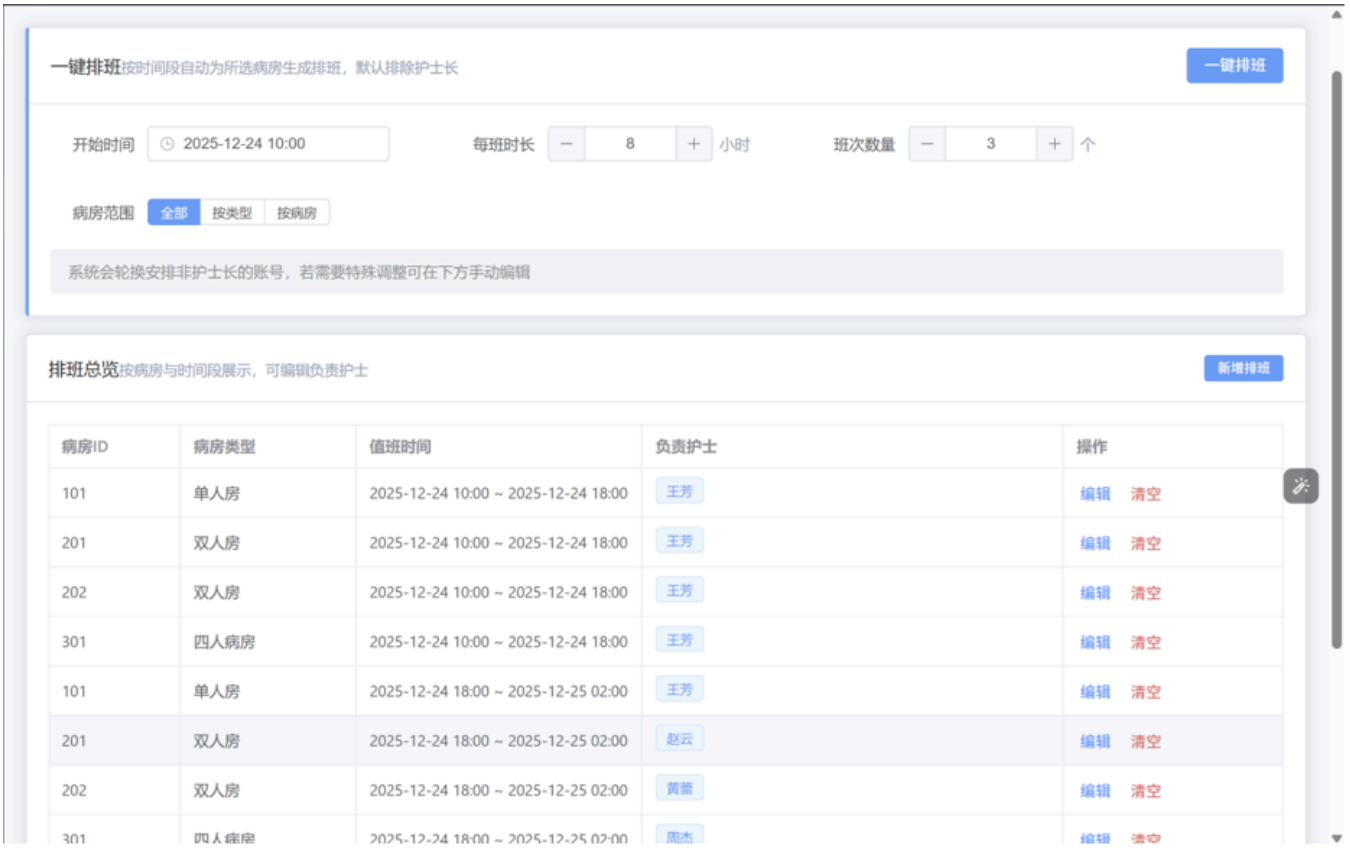
类型	金额	时间	状态	操作	附加信息
挂号费	¥50	2025-12-24 10:44:48	未缴费	缴费	挂号费用
住院费	¥64	2025-12-24 10:43:50	已缴费	-	**住院：** 双人房 入院：2025-12-24 10:30:18 出院：2025-12-24 10:43:50 时长：0.2 小时 合计 ¥64.00 查看费用详情
挂号费	¥10	2025-12-24 10:43:01	已退费	-	挂号费用
挂号费	¥10	2025-12-24 10:31:19	已取消	-	挂号费用
挂号费	¥10	2025-12-24 10:28:24	已缴费	-	挂号费用
检查费	¥5	2025-12-24 10:26:33	已缴费	-	**检查：** CT 结果：正常 时间：2025-12-24 10:25:54
处方费	¥70	2025-12-24 10:26:31	已缴费	-	**处方：** 云南白药气雾剂 × 1 （一日三次）
挂号费	¥50	2025-12-24 01:19:19	已缴费	-	挂号费用

7.2.3 数据推荐

7.3 业务功能

7.3.1 支持看诊全流程

7.3.2 护士长便捷排班和手动调整

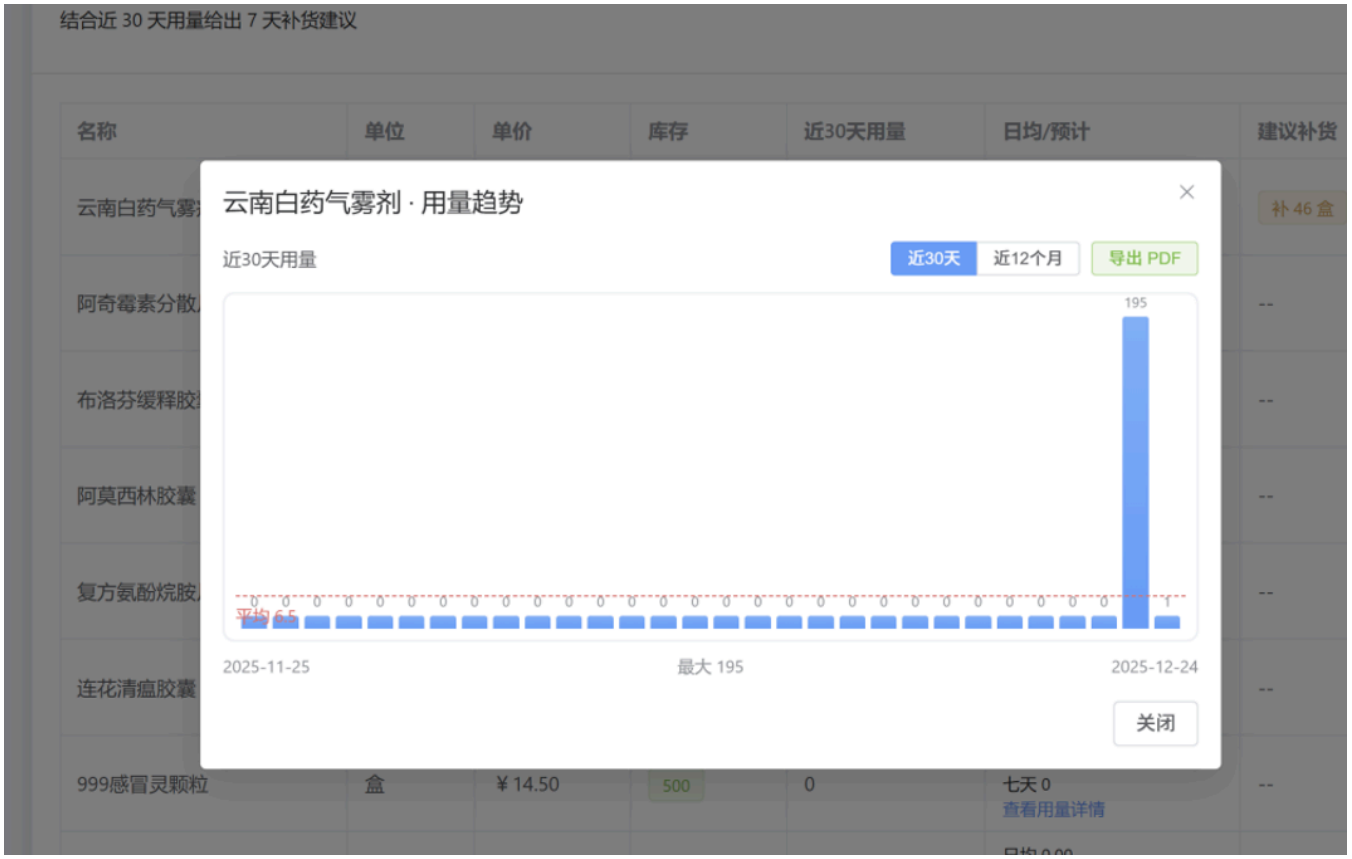


7.3.3 护士任务提醒警告

7.4 统计分析

7.4.1 营收统计图

7.4.2 药品使用统计与预测

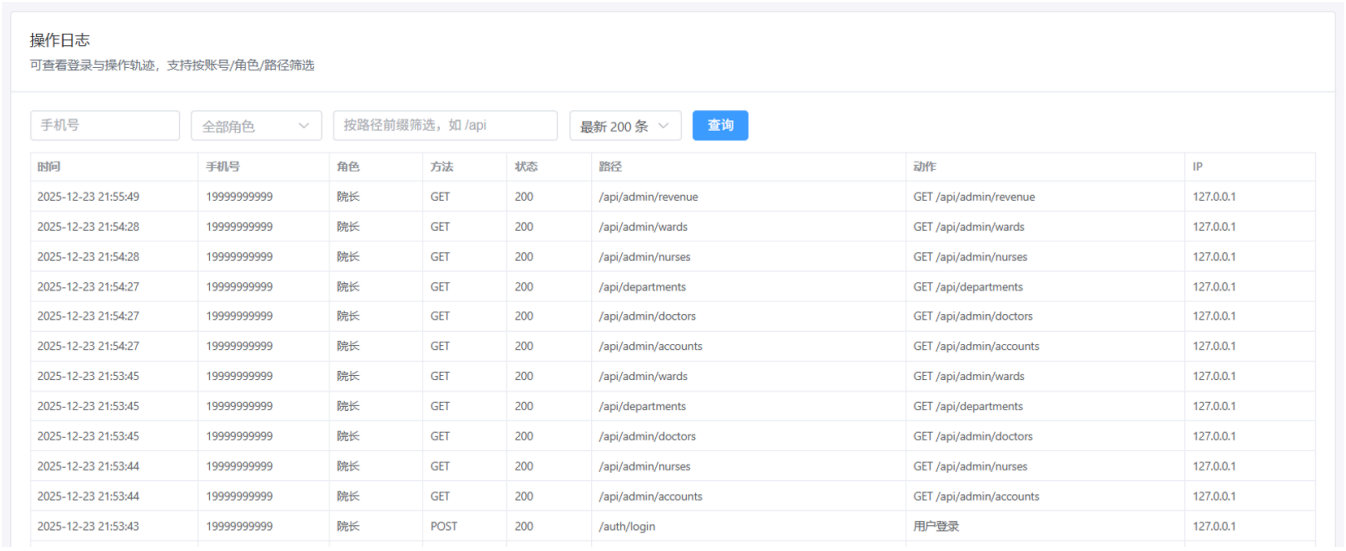


7.5 安全防护

7.5.1 支持多类别用户权限统一管理

```python
@router.post("/login", response_model=Token)
async def login(
    form_data: OAuth2PasswordRequestForm = Depends(),
    session: AsyncSession = Depends(get_session),
    request: Request = None,
):
    # A. 根据手机号查询用户 (form_data.username 对应前端传来的手机号)
    result = await session.execute(select(UserAccount).where(UserAccount.phone == form_data.username))
    user = result.scalars().first()
```

7.5.2 访问日志检查



操作日志

可查看登录与操作轨迹，支持按账号/角色/路径筛选

手机号 | 全部角色 | 按路径前缀筛选，如 /api | 最新 200 条 | 查询

时间	手机号	角色	方法	状态	路径	动作	IP
2025-12-23 21:55:49	19999999999	院长	GET	200	/api/admin/revenue	GET /api/admin/revenue	127.0.0.1
2025-12-23 21:54:28	19999999999	院长	GET	200	/api/admin/wards	GET /api/admin/wards	127.0.0.1
2025-12-23 21:54:28	19999999999	院长	GET	200	/api/admin/nurses	GET /api/admin/nurses	127.0.0.1
2025-12-23 21:54:27	19999999999	院长	GET	200	/api/departments	GET /api/departments	127.0.0.1
2025-12-23 21:54:27	19999999999	院长	GET	200	/api/admin/doctors	GET /api/admin/doctors	127.0.0.1
2025-12-23 21:54:27	19999999999	院长	GET	200	/api/admin/accounts	GET /api/admin/accounts	127.0.0.1
2025-12-23 21:53:45	19999999999	院长	GET	200	/api/admin/wards	GET /api/admin/wards	127.0.0.1
2025-12-23 21:53:45	19999999999	院长	GET	200	/api/departments	GET /api/departments	127.0.0.1
2025-12-23 21:53:45	19999999999	院长	GET	200	/api/admin/doctors	GET /api/admin/doctors	127.0.0.1
2025-12-23 21:53:44	19999999999	院长	GET	200	/api/admin/nurses	GET /api/admin/nurses	127.0.0.1
2025-12-23 21:53:44	19999999999	院长	GET	200	/api/admin/accounts	GET /api/admin/accounts	127.0.0.1
2025-12-23 21:53:43	19999999999	院长	POST	200	/auth/login	用户登录	127.0.0.1

7.5.3 IP校验


当前 IP 注册过于频繁，请稍后再试
医院管理系统
统一登录入口 · 支持患者 / 医生 / 护士 / 药师 / 管理员
立即登录
快速注册
手机号
12046295634
用户名
q
密码
确认密码
注册并登录

7.5.4 密码机制

phone	username	password_hash	role	register_time	status
1	hzy	$2b$12$XbK2/PBjhrxYDcAZ6GArhe.r6hRgBLqIq0Xu8EwcbQIDPTXseS.rm	PATIENT	2025-12-02 10:20:40	启用
10236457323	432	$2b$12$v.rli4kldKqISRz6V4L6jeHiALq8NzOmKy9u2uRzrUC0QTXUafP.K	PATIENT	2025-12-22 17:54:10	启用
11	1	$2b$12$7JOyMy1IsSfAqQRGGoSNw.liw73gVSoOxURPnP2opsPDatMw6UknK	PATIENT	2025-12-22 01:38:21	启用
111	11	$2b$12$26ElyuL..RISpUtO9./16./32Q6/XVjv9aOSbkNpoDu7ANo2P2G8e	PATIENT	2025-12-22 01:38:31	启用
11111111111	1111	$2b$12$4W/bNcXxVrW/S5yBIfjfh.HrTKeS02kllDRGRhsWYwa62V1d9WSBK	DOCTOR	2025-12-19 23:43:13	启用
11111111112	hh	$2b$12$0ZWlYOKjg/n.PmIJJnsSJui2meZ.iZfeNforubzQdsQsH5bQG.9.2	NURSE	2025-12-21 22:41:05	启用
11423569459	afewf	$2b$12$fvuu/DGV55wA6.c/j0Sls.oKNQ122jlnDtZJCJGQQAZHPx4AFwElq	PATIENT	2025-12-22 17:59:34	启用
12345678901	hushi	$2b$12$D8wmT71cdS91ZnoFaGGqe.goLtjrJuQz0aE7fQND3BKo8aRkkzJm2	NURSE	2025-12-02 10:28:54	启用
12345678910	doctor	$2b$12$uOJgEyTqiAUi7oKFlv4glOmvSuiNsupQyElTKbPoXwRnF8Neu6cey	DOCTOR	2025-12-02 10:18:55	启用
16534795362	djsalf	$2b$12$3cIB1kvGco.t8iOkdvHBveeYOj/WaVVQ/RbxeRv0SPZ3DhWrGjAWK	PATIENT	2025-12-22 17:59:22	启用
18612105277	yaoshi	$2b$12$/BTVz7wrXXGhpnO2CB4v2OG8pcTILgOKUvC8KQKDPpZNsFF0Xo7xu	PHARMACIST	2025-12-02 10:29:34	启用
19876543210	yes	$2b$12$mcv6x.pnRkY6CXMDrShFdeTHWZt7J0JZnYf2B/3q1dYvnzPrX0BHu	PATIENT	2025-12-22 01:41:30	启用
19999999999	院长	$2b$12$Q5IY.zxwa7EHJRAk0qnX5.6syB7riHR.8i5pM0Q2qO7K1B/fdB7ju	ADMIN	2025-12-02 10:13:15	启用

8.源程序简要说明

各文件、函数前文已详细介绍，不再赘述。

运行方式具体如下，以windows为例：

1.启动本地 MySQL 服务：

例如在 Windows：

```
mysql -u root -p
```

2.创建数据库：

```
CREATE DATABASE IF NOT EXISTS hms_db;
```

3.数据库导入：

在项目根目录的文件`.env`中编辑：

```
DATABASE_URL=mysql+aiomysql://root:你的密码@127.0.0.1:3306/hms_db
```

4.运行后端

```
pip install -r requirements.txt                 # 首次需要install
uvicorn app.main:app --reload --port 8001       # 默认绑定 http://127.0.0.1:8001
```

默认初始化一个院长账号：手机号`19999999999`，密码`Director@123`

5.运行前端

```
cd HMS\frontend
npm install          # 首次需要
npm run dev          # 默认 http://127.0.0.1:5173
```

9.收获和体会

该项目是我们做过有史以来最为复杂，功能最为全面的项目。我们从未设想过完成如此完善的功能链条，如此复杂的实体的关系。多轮迭代，终于优异地完成了挑战。

我们团队通过 Git 进行了规范化的协作开发，在分支管理与代码合并的过程中，深刻体会到了版本控制对于多人项目开发效率的保障作用。相比于 Git 的掌握，更重要的是，本次实践中，我们亲手打通了从底层数据库建模到后端逻辑处理，再到前端界面展示的完整链路，真正厘清了一个真实项目在各层级间的运作逻辑与数据流转，使我们对全栈开发有了全局性的认识。

我们深刻意识到，数据库设计是整个项目的基石，必须精心与仔细推敲，因为任何初始阶段的逻辑疏忽都可能在后期开发中引发连锁反应。同时，我们也坚信高效沟通的力量。面对逻辑复杂的攻坚环节，与其独自闷头苦干，积极与小组成员交流探讨往往能避免大量无效的重复劳动。

优秀的系统不是代码的简单堆砌，而是源于严谨的架构设计与紧密的团队协作。