



中山大学计算机学院

数据库系统原理

课程项目报告

医院挂号系统

0. 小组成员及分工

- 李骏唯：组长，学号21312378，负责后端代码和统筹协调
- 姜波：组员，学号21311290，负责数据库管理和测试
- 何叶：组员，学号21307223，负责前端代码

1. 应用需求介绍

1.1 背景

预约挂号是当前现代化医疗体系不可缺少的内容，有就医需求的人群可以通过在网上提前挂号，合理安排就诊计划和出行时间，能够有效提高患者和医院双方看病就诊的效率，实现线上就诊的智能化、信息化。同时，考虑到近年来全球性疫情的影响，尤其是新冠疫情和最近频发的甲流乙流等流感的严峻形势，这些卫生突发事件使得医疗资源的合理调配和医疗服务的高效运作变得尤为关键。在这些传染性较强的疾病面前，传统的挂号方式，比如说电话挂号或者更古老的线下挂号，让信息管理系统面临了很大的挑战，长时间的排队等待也容易造成人群聚集，增加疫情传播的风险。同时，及时准确的患者信息管理也是医院的一大紧急任务。

因此，医院挂号管理系统的建立和功能优化是一个很值得研究的课题，它可以为医疗机构提供了一种更为安全和高效的挂号和信息管理方式。通过系统化、电子化的挂号流程，患者可以在避免人群聚集的前

提下完成挂号，降低了流感感染风险。而系统内部的患者信息管理也有助于医护人员迅速获取患者的病历和就诊历史，为医疗决策提供有力支持。此外，系统还能够通过智能排班和信息管理，更好地应对患者大量涌入的情况，提前做好医生资源的调配，确保及时就诊。同时，系统中的病历填写和信息记录功能也有助于医生更全面、系统地了解患者的病情，提高诊疗的准确性和效率。

总的来说，医院挂号管理系统的背景与当前社会生活息息相关，还紧密结合了疫情防控和流感高发季节的实际需求。通过数字化和智能化手段，医院挂号管理系统在应对突发状况、提高医疗服务方面发挥着重要作用，成为医疗行业不可或缺的一部分。综合种种原因，我们选择实现一个简单的医院挂号管理系统。

1.2 功能需求

结合实体医院的挂号情况和方式以及我们本身的设计需求和水平，我们计划设计的系统能实现基本的预约挂号、就诊和结束就诊的功能。医院预约挂号管理系统的功能主要包括患者预约挂号、医生接诊并处理就诊以及系统后台管理三部分。

序号	角色	需求简述
1	患者	可以查看医院科室和医生信息，挂号预约，取消预约，查看病历，查看历史预约记录，个人信息管理
2	医生	可以查看挂号记录，接诊，填写病历
3	管理员	对数据库进行更新和维护

具体的功能需求：

1. 登录和注册功能：
 - 设定登录和注册功能，需要正确登录才能进入界面。
 - 设定医生、患者的用户权限，保障不同角色的合理访问权限。
2. 挂号功能：
 - 实现患者线上挂号，包括选择医生、科室和就诊日期以及时段。
 - 生成唯一的挂号记录，方便患者查询和医院管理。
3. 患者信息管理：
 - 记录患者的个人信息，包括用户名、密码等。
 - 记录患者的预约信息，并支持患者可以自行取消预约。
 - 整合患者的医疗档案，包括病情描述和处方等。

4. 医生个人介绍：

- 提供医生的个人资料，方便患者进行查询和了解。

5. 病历填写和管理：

- 给医生提供在线病历填写表单，包括患者症状、诊断结果和治疗方案。
- 记录医生对患者的诊断和治疗建议，形成完整的电子病历。
- 只允许患者在个人登录界面内查看，确保医疗信息的完整性和保密性。

6. 药品查询功能：

- 提供查询药房中的药品功能，方便医生了解，提高工作效率。

其中，在用户界面，用户（患者）可以在任意时间段内进入挂号系统，按照自身需求和喜好进行挂号预约。在医生界面，医生可以查看关于自己的预约记录，在上班时间内接诊，按照患者情况填写病历和开药。系统后台管理可以对平台数据进行存储、维护和更新，并对患者挂号预约数据进行收集整理。总体来说该系统设计需要满足用户预约挂号、医院医生和科室查询、就诊接诊等基本的功能需要，满足医院信息化、智能化就诊的基本要求，能为患者就诊和医生接诊提供更便利的服务。

2. 实验环境

2.1 技术栈

- 界面：web
- 前端：html + css + javaScript
- 后端：Python + Django

2.2 实验环境

- 虚拟机：Oracle VM VirtualBOX
- 虚拟机OS：CentOS Linux release 7.9.2009 (Core)
- 数据库系统：gaussdb (openGauss 5.1.0)
- 框架：Django 3.0.0

Windows下连接本地虚拟机数据库端口进行开发

2.3 环境配置

2.3.1 OpenGauss数据库配置

虚拟机网络配置：

启用两个虚拟机网卡，一个设置为 仅主机(Host-Only)网络，另一个设置为 网络地址转换(NAT)，在虚拟机执行

ifconfig

找到类似 192.168.56.10x 的ip地址：

```
[root@localhost /]# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.56.109 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::bacb:68c6:bf02:c87d prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:cd:0d:b9 txqueuelen 1000 (Ethernet)
        RX packets 20056 bytes 3059089 (2.9 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 17284 bytes 4303537 (4.1 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.3.15 netmask 255.255.255.0 broadcast 10.0.3.255
    inet6 fe80::a05a:2b72:331a:2345 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:61:87:e9 txqueuelen 1000 (Ethernet)
        RX packets 507 bytes 41786 (40.8 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 542 bytes 44701 (43.6 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 221734 bytes 78680707 (75.0 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 221734 bytes 78680707 (75.0 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:bd:b7:25 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

查看是否能够从主机ping通虚拟机。

openGauss部署：

1. 首先按照安装教程安装数据库： [极简版安装](#)。(需要创建一个操作系统的普通用户: omm, 数据库操作都在普通用户下进行)
2. 数据库连接配置，在 /opt/opengauss/data/single_node/postgresql.conf 配置中，修改以下值(可能需要root权限)

```
listen_addresses = '*' # 监听本地所有地址  
port = 26000           # 数据库服务的端口号，自行设置
```

3. 数据库的配置

数据库的初始化，启动与连接

```
# 初始化数据库  
gs_initdb -D /opt/opengauss/data/single_node -w "Gauss@123" --nodename='data'  
# 启动数据库  
gs_ctl start -D /opt/openGauss/data/single_node/ -Z single_node  
# 连接数据库，-p 后为第二部设置的port值  
gsql -d postgres -p 26000 -r
```

数据库中的用户

```
# 创建数据库用户(连接数据库后执行指令，注意分号)  
CREATE USER omm WITH PASSWORD "Gauss@123";  
# 创建数据库，属于omm用户  
CREATE DATABASE db_project OWNER omm;  
# 再创建一个非数据库持有者用户admin，后续Django连接数据库的用户不能是数据库的持有者  
CREATE USER admin WITH PASSWORD "Gauss@123";  
# 授予admin所有权限  
GRANT ALL PRIVILEGES ON db_project TO admin;  
  
# 以admin身份连接数据库，Django都是以admin进行操作，只有admin能看到  
gsql -d db_project -r -p 26000 -U admin
```

遇到的问题：

1. 执行以下启动时，数据库启动失败，显示 permission denied

```
gs_ctl restart -D /opt/software/openGauss/data/single_node -Z  
single_node -l logfile
```

由于普通用户的权限问题，可能无法写日志文件logfile，去掉 -l logfile 后能够正常启动。

2. 使用vscode的ssh连接虚拟机进行环境配置时，直接打开了根目录文件夹 / (便于查找文件)，由于虚拟机只分配了4G内存，查看内存占用时发现占了3G，没有有足够的内存来启动数据库，而且执行清理内存指令无法清除。

断开vscode的连接后使用命令行的ssh直接连接，发现内存占用正常，能够正常启动数据库(因为这个重新还配置了一遍环境QAQ)。

推荐完成环境配置后使用命令行的ssh进行连接虚拟机来启动数据库(或者直接使用虚拟机启动)。

2.3.2 Django 配置

安装Django 3.0.0

```
pip install Django==3.0.0
```

基本操作

```
#初始化项目  
django-admin startproject 项目名称  
#创建应用  
python manage.py startapp 应用名称  
#运行服务器  
python manage.py runserver  
#生成迁移文件  
python manage.py makemigrations  
#执行sql语句生成数据表  
python manage.py migrate
```

在创建完成项目与应用后，每次修改 `models.py` 都要执行生成迁移文件和生成数据表

```
python manage.py makemigrations  
python manage.py migrate
```

Django数据库与openGauss进行连接，修改项目中 `settings.py` 文件中的数据库配置：

```

DATABASES = {
    'default': {
        # 'ENGINE': 'django.db.backends.sqlite3',
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'db_project', #数据库名
        'USER': 'admin', #用户名
        'PASSWORD': 'Gauss@123', #密码
        'HOST': '192.168.56.109', #虚拟机ip
        'PORT': 26000 #openGauss数据口的端口
    }
}

```

3. 应用系统设计

3.1 角色设计

序号	身份	功能简述
1	患者	查看医院科室和医生信息, 挂号预约, 取消预约, 查看病历, 查看历史预约记录, 查看个人信息
2	医生	查看挂号记录, 接诊, 填写病历, 查看个人信息
3	管理员	对数据库进行更新和维护

3.2 界面设计

医院挂号系统的界面总体可分为三个部分。

1. 登录和注册界面

- 在登录界面提供用户名和密码输入表单, 设计“医生登录”和“患者登录”按钮区分不同身份的角色登录。
- 在登录界面提供注册按钮, 跳转到注册界面, 让患者填写个人信息并注册, 需要在后台检查输入信息是否合法。

2. 患者界面

患者登录后进入患者界面的首页, 提供导航栏跳转到相应的功能区。功能区分为以下五个:

- 首页：显示医院的介绍、联系方式等信息。
- 预约挂号：提供医生列表，表格的每行包括医生的姓名、所属科室和预约按钮。在顶部设计搜索栏，支持筛选医生名字和选择科室，方便患者快速定位想挂号的医生或科室。
 - 预约界面：患者点击预约按钮可跳转到该医生的预约界面，让患者选择就诊日期和就诊时段并提交。
 - 预约成功界面：患者预约成功后显示，提供返回键返回到挂号界面。
- 挂号记录：展示挂号记录的列表，每个挂号记录都展示医生姓名、时间、时段、状态等消息，支持患者取消预约（仅在“已挂号”状态下），支持查看病历（仅在“结束就诊”状态下）。
 - 查看病历界面：显示病历信息，提供返回键。
- 医生详情：医生详情页面提供每个医生的详细信息。在医生详情界面，以科室划分罗列出各医生名字，用户可以点击医生名字跳转到对应医生的详情页。医生的详情页包括职称、科室、科室位置、详细描述等该医生的信息，便于用户了解医生。
 - 医生详情页：展示医生的姓名、科室、职称、简介等信息。
- 个人信息：展示患者的个人信息，提供注销按钮。

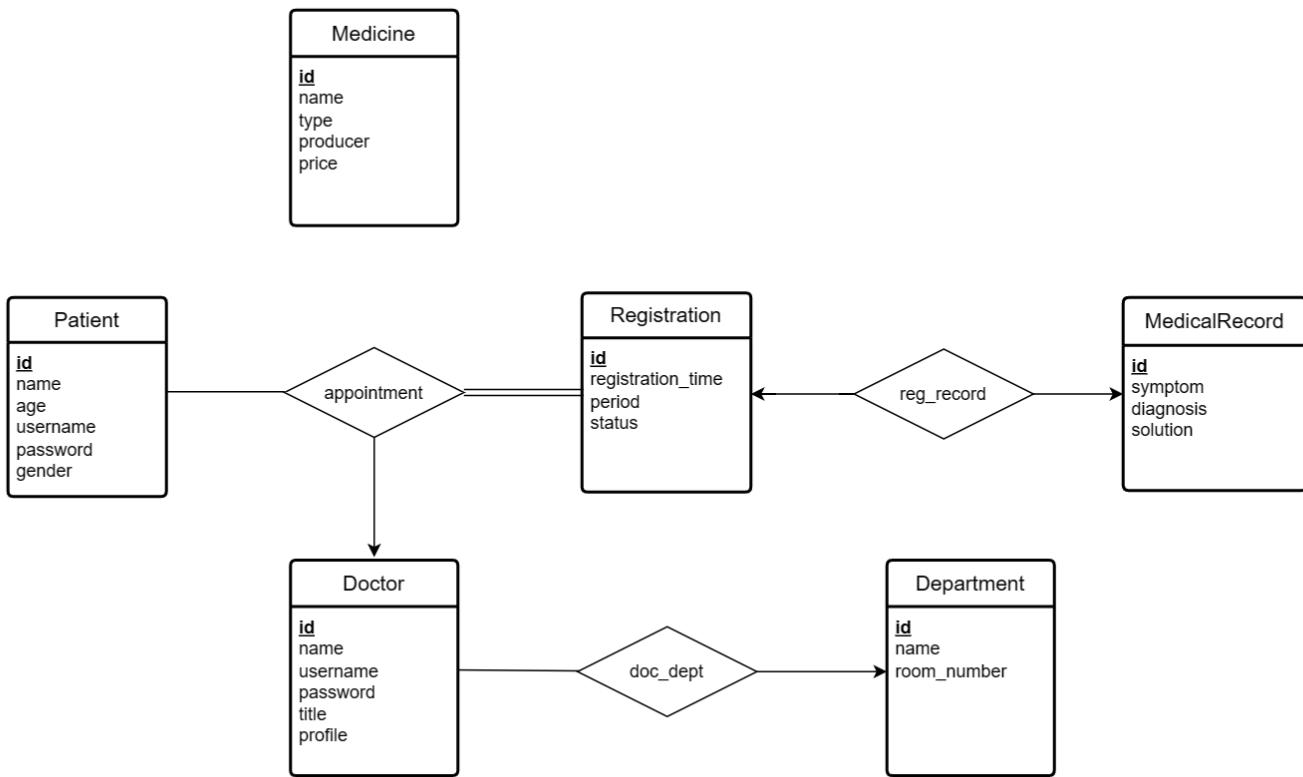
3. 医生界面

医生登录后进入医生界面的首页，提供导航栏跳转到相应的功能区。功能区分为以下四个：

- 首页：显示医院的介绍、联系方式等信息。
- 挂号记录：提供预约列表，每条预约包括患者姓名、预约时间、预约时段、状态右侧和操作。操作有“接诊”（仅在“已挂号”状态下）、填写病历（仅在“就诊中”状态下）、结束就诊（仅在“就诊中”状态下）。
 - 填写病历界面：显示患者姓名，提供填写症状、诊断结果、治疗方案的表单，提供保存按钮和返回上一页键。
- 药房：提供药品列表，在顶部设计搜索栏，支持基于药品名和药品类型的筛选。
- 个人信息：展示医生的个人信息，并提供注销按钮。

3.3 数据库设计/E-R图

基于上述功能需求和设计，E-R图设计如下：



说明：

- appointment联系集只有一个箭头指向Doctor，这是因为一个患者在一次预约中最多只能指定一个医生，而患者可以多次预约一个医生，也可以多次预约多个医生。
- Registration和appointment之间用双横线连接，即Registration必须全参与联系集，表明一个预约必须与一个医生和一个患者相关联。
- doc_dept与Doctor用实线连接，与Department用箭头连接，表明Doctor和Department是多对一的关系，即一个医生只能属于一个科室，一个科室可以有多个医生。
- reg_record与两个实体集都是箭头连接，表明二者是一对一的关系，即一次预约只能产生一份病历。

对应的数据库关系模式如下：

1. Patient (患者) :

- 用于存储患者的基本信息。
- 字段：
 - id : 标识符，主键。
 - name : 患者姓名。
 - age : 患者年龄。
 - username : 患者用户名，唯一性约束。

- password : 患者密码。
- gender : 患者性别，只允许在“男”或“女”中选择。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个患者需要一个唯一的标识符。
- 外键约束：无。
- 字段值约束：性别字段使用 Django 的 CHOICES 机制限制了固定的选项，是因为性别是有限且预定义的类别，通过提供选择可以确保数据的一致性和规范性。
- 唯一性约束：患者的用户名不允许重复。

2. Department (科室) :

- 用于存储医院各科室的信息。
- 字段：
 - id : 标识符，主键。
 - name : 科室名称。
 - room_number : 科室房间号。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个科室需要一个唯一的标识符。
- 外键约束：无。
- 字段值约束：无，因为科室的名称和房间号是自由文本，不需要提供预定义的选项。

3. Doctor (医生) :

- 用于存储医生的信息。
- 字段：
 - id : 标识符，主键。
 - name : 医生姓名。
 - username : 医生用户名，唯一标识。
 - password : 医生密码。
 - title : 医生职称。
 - dept : 医生所属科室，通过外键关联到科室模型。
 - profile : 医生简介。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个医生需要一个唯一的标识符。
- 外键约束：科室字段采用外键关联到科室模型，因为医生属于特定的科室，通过外键关联建立了医生与科室之间的关系。
- 字段值约束：无，因为职称字段是自由文本，不需要提供预定义的选项。

4. Registration (挂号记录) :

- 用于存储患者挂号记录的信息。
- 字段：
 - id : 标识符，主键。
 - doctor : 挂号的医生，通过外键关联到医生模型。
 - patient : 挂号的患者，通过外键关联到患者模型。
 - registration_time : 挂号时间。
 - period : 挂号时段，通过选择不同的时段来表示。

- `status` : 挂号记录的状态，在已挂号、已取消、就诊中、就诊结束中选择。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个挂号记录需要一个唯一的标识符。
- 外键约束：医生和患者字段采用外键关联到医生模型和患者模型，确保了挂号记录与医生和患者之间的关联。
- 字段值约束：状态和时段字段使用 Django 的 CHOICES 机制限制了固定的选项，是因为它们是有限且预定义的类别，通过提供选择可以确保数据的一致性和规范性。

5. MedicalRecord (病历) :

- 用于存储患者的病历信息。
- 字段：
 - `id` : 标识符，主键。
 - `symptom` : 症状描述。
 - `diagnosis` : 诊断结果。
 - `solution` : 治疗方案。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个病历需要一个唯一的标识符。
- 外键约束：挂号记录字段采用外键关联到挂号记录模型，建立了病历与挂号记录之间的联系。
- 字段值约束：无，因为病历的症状、诊断结果和治疗方案是自由文本，不需要提供预定义的选项。

6. Medicine (药物) :

- 用于存储药物的信息。
- 字段：
 - `id` : 标识符，主键。
 - `name` : 药物名称。
 - `type` : 药物类型，只允许在“中药”或“西药”中选择。
 - `producer` : 生产厂商。
 - `price` : 药物价格。
- 主键约束：默认使用 Django 自动生成的自增主键，因为每个药物需要一个唯一的标识符。
- 字段值约束：类型字段使用 Django 的 CHOICES 机制限制了固定的选项，是因为药物类型是有限且预定义的类别，通过提供选择可以确保数据的一致性和规范性。

在Django的 `models.py` 搭建数据库模型，代码如下：

```
from django.db import models

class Patient(models.Model):
    '''患者'''
    name = models.CharField(verbose_name='姓名', max_length = 20)
    age = models.IntegerField(verbose_name='年龄', null=True, blank=True)
    username = models.CharField(verbose_name='用户名', max_length = 20, unique=True)
    password = models.CharField(verbose_name='密码', max_length = 20)
    GENDER_CHOICES = [
        (0, '男'),
        (1, '女'),
    ]
    gender = models.SmallIntegerField(verbose_name='性别', choices=GENDER_CHOICES, null=True, bl

class Department(models.Model):
    '''科室'''
    name = models.CharField(verbose_name='科室名', max_length = 30)
    room_number = models.CharField(verbose_name='房间号', max_length = 20)

class Doctor(models.Model):
    '''医生'''
    name = models.CharField(verbose_name='姓名', max_length = 20)
    username = models.CharField(verbose_name='用户名', max_length = 20, unique=True)
    password = models.CharField(verbose_name='密码', max_length = 20)
    title = models.CharField(verbose_name='职称', max_length = 20)
    dept = models.ForeignKey(Department, verbose_name='科室', on_delete=models.CASCADE, related_
    profile = models.CharField(verbose_name='简介', max_length = 200)

class Registration(models.Model):
    '''挂号记录'''
    STATUS_CHOICES = [
        (0, '已挂号'),
        (1, '已取消'),
        (2, '就诊中'),
        (3, '就诊结束'),
    ]
    PERIOD_CHOICES = [
        (0, '8:00-10:00'),
        (1, '10:00-12:00'),
        (2, '14:00-16:00'),
        (3, '16:00-18:00'),
    ]
```

```

]

doctor = models.ForeignKey(Doctor, verbose_name='医生', on_delete=models.CASCADE, related_name='records')
patient = models.ForeignKey(Patient, verbose_name='患者', on_delete=models.CASCADE, related_name='records')
registration_time = models.DateTimeField(verbose_name='挂号时间')
period = models.SmallIntegerField(verbose_name='挂号时段', choices=PERIOD_CHOICES)
status = models.SmallIntegerField(verbose_name='状态', choices=STATUS_CHOICES)

class MedicalRecord(models.Model):
    '''病历'''
    symptom = models.CharField(verbose_name='症状', max_length = 200)
    diagnosis = models.CharField(verbose_name='诊断结果', max_length = 200)
    solution = models.CharField(verbose_name='治疗方案', max_length = 200)
    registration = models.ForeignKey(Registration, verbose_name='预约信息', on_delete=models.CASCADE)

class Medicine(models.Model):
    '''药物'''
    TYPE_CHOICES = [
        (0, '中药'),
        (1, '西药'),
    ]
    name = models.CharField(verbose_name='名称', max_length = 30)
    type = models.SmallIntegerField(verbose_name='类型', choices=TYPE_CHOICES)
    producer = models.CharField(verbose_name='生产厂商', max_length = 30)
    price = models.DecimalField(verbose_name='价格', max_digits=5, decimal_places=2)

```

总的来说，数据库设计实现了医院挂号系统的核心功能，包括患者、医生、科室、挂号记录、病历和药物等信息的管理和交互。通过外键关联，实现了这些模型之间的关系，为系统提供了有效的数据库结构。

4. 前端实现

- 整个前端网页按照设计风格可以大致分成三部分，分别是登陆注册界面、用户界面和医生界面。用户界面和医生界面的网页可以分为两部分，导航栏和内容主体。左边的导航栏用于实现网页跳转的交互，右边的容器存放对应的显示内容。
- 在实验报告中仅展示功能实现的代码，完整的关于CSS样式部分可以在提交的完整实验代码中查看。

4.1 登陆注册界面

4.1.1 登录模块 (login.html)

- 登陆模块的应用对象包括患者和医生，系统为不同的登陆身份设置了不同的使用权限。
- 在网页的登录表格内可以对用户账号进行登录和注册操作。如果先前已经创建了账号，用户可以直接输入账号和密码进行登录，后台会对输入的账号和密码通过调用数据库中的数据进行核对，核对成功则完成登录操作，进入相应的页面。如果密码错误或者输入的账号没注册过，会有错误提示。我们给患者提供了注册功能，可以点击注册按钮进入注册界面（医生账号由系统管理，不能自己注册）。
- 网页内容设计部分放在 login.css 中。

核心代码：

```
<div class="login">
    <div class="login-triangle"></div>
    <h2 class="login-header">登录</h2>
    <form class="login-container" method="post" novalidate>
        {% csrf_token %}
        {% for field in form %}
            <p>{{ field }}</p>
            <span class="error">{{ field.errors.0 }}</span>
        {% endfor %}
        <div class="container" style="margin-top: 20px;">
            <div class="row">
                <div class="col-md-6">
                    <button type="submit" name="login_type" value="doctor_login" class="btn btn-block b1">医生登陆</button>
                </div>
                <div class="col-md-6">
                    <button type="submit" name="login_type" value="patient_login" class="btn btn-success b2">患者登陆</button>
                </div>
            </div>
        </div>
    </form>
</div>
```

4.1.2 注册模块 (enroll.html)

- 这部分主要实现了一个简单的患者注册页面，包括基本的样式、表单验证和交互效果。使用了 Bootstrap 框架提供的样式和组件，使页面看起来更加美观，也能体现web的响应式功能。
- 用户通过向数据库提交表单的形式注册新账号。在 Django 中，表单通常由一个 Form 类表示，该类的实例被传递到模板中，以便在模板中生成表单元素。通过 `{% for field in form %}` 循环，模板根据表单的定义，为账号和密码字段生成相应的标签、输入框等 HTML 元素。`{ field.errors.0 }` 用来显示字段的错误信息，比如说账号和密码不能为空。

核心代码：

```
{% csrf_token %}  
<div class="container" style="background-color: #fff; padding: 20px;">  
    {% for field in form %}  
        <div class="form-group">  
            <label>{{field.label}}</label>  
            <div class="col-md-8 inputGroupContainer">  
                <div class="input-group">  
                    <span class="input-group-addon"><i class="glyphicon glyphicon-user"></i></span>  
                    {{field}}  
                </div>  
                <span class="error">{{field.errors.0}}</span>  
            </div>  
        </div>  
    {% endfor %}  
</div>
```

4.2 患者界面

4.2.1 主页模块 (home.html)

- 从导航栏可以知道当前页面的位置，点击不同页面导引可以跳转到相应的页面。右边是首页的内容主体，展示了医院简介、提供的服务以及医院的联系方式。医院简介是直接显示的，没有调用数据库，篇幅较长，可在实验完整代码中查看，这里不再展示。

核心代码（导航栏实现）：

```

<div class="menu-container">
    <!-- 菜单栏 -->
    <div class=" sidenav" style="bottom: 0;">
        <h5 style="color: #fff; margin: 30px; margin-top: 10.6px; font-size: 1.449em; letter-spacing: 0.1em; border-bottom: 1px solid #fff; padding-bottom: 5px; position: relative; z-index: 1;">患者中心</h5>
        <a href="/patient/home" class="active">首页</a>
        <a href="/patient/register" >预约挂号</a>
        <a href="/patient/personnel">医生介绍</a>
        <a href="/patient/registrations">挂号记录</a>
        <a href="/patient/personal" style="padding-top: 8px;">个人中心</a>
    </div>

```

4.2.2 预约挂号模块 (register.html)

- 左边是导航栏，右边是预约挂号的流程和功能实现。
- 用户可以通过在上方搜索栏输入想要挂号的医生姓名和选择对应的科室，了解排班的医生情况。点击预约按钮，跳转到预约界面就可以进行挂号预约。页面下方实现了分页功能，在展示内容较多时可以实现分页跳转。

核心代码：

- 搜索栏

```

<form method="get" action="/patient/register/">
    <label for="name" style="font-weight: bold;">医生姓名: </label>
    <input type="text" id="name" name="name" value="{{ name_query }}" style="margin-right: 10px; width: 150px; height: 35px; border-radius: 10px; border: 1px solid #ccc; padding: 5px; font-size: 14px; font-family: inherit; vertical-align: middle;">
    <label for="department" style="font-weight: bold;">科室: </label>
    <select id="department" name="department">
        <option value="" {{ if not dept_query }}selected{{ endif }} style="color: #999;">选择科室</option>
        {% for department in departments %}
            <option value="{{ department.id }}" {{ if department.id == dept_query }}selected{{ endif }} style="color: #000; font-weight: bold;">{{ department.name }}</option>
        {% endfor %}
    </select>
    <button type="submit" class="btn btn-primary" style="margin-left: 10px; background-color: #007bff; color: white; border: none; border-radius: 5px; padding: 5px 15px; font-size: 14px; font-weight: bold;">搜索

```

- 显示搜索结果

```


| 医生姓名 | 科室 | 操作                                                                              |
|------|----|---------------------------------------------------------------------------------|
|      |    | <a href="#" onclick="redirectToAppointment('123456789')">         姚明       </a> |
|      |    | <a href="#" onclick="redirectToAppointment('987654321')">         张伟       </a> |


```

4.2.3 医生详情模块

- 设计思路：

在预约挂号之前，用户可以通过导航栏的 医生介绍 索引跳转到 personnel.html 页面查看所有科室和医生信息。为了页面的简洁，我们仅以科室作为分类，显示每个科室下分管的所有医生。想要了解每个医生的信息详情可点击该医生的姓名跳转到详情介绍界面。

- 内容实现：

- 在医生介绍简介部分，使用 `{% for depart in departments %}` 循环语句将所有的科室和医生信息显示出来，点击姓名即可跳转详情连接。

```
<div class="container">
    <!-- 医生介绍部分 -->
    {% for depart in departments %}
        <h4 class="department-name">{{ depart.name }}</h4>
        <div>
            {% for doctor in doctors %}
                {% if doctor.dept == depart %}
                    <span class="doctor-name"><a href="/patient/profile/{{ doctor.id }}" style="color: #007bff; text-decoration: none; font-weight: bold;">{{ doctor.name }}</a></span>
                {% endif %}
            {% endfor %}
        </div>
    {% endfor %}
</div>
```

- 在详情介绍 profile.html 部分，通过传参，把对应的医生信息从数据库中读取并显示，内容包括医生的姓名、职称、所属科室、科室房间号以及相应的个人简介。在右上角提供了 返回上一页 按钮跳转。

```
<div class="container">
    <div class="col-12 text-center" style="background-color: #112434; ">
        <h3>详情介绍</h3></div>
    <div class="row"><div class="col-12 text-center">
        <h1>{{ doctor.name }}</h1>
    </div></div><hr>
    <div class="row mt-3" style="text-align: center; " >
        <div class="col-md-4"><div class="form-group">
            <label for="title"> 职称: </label>
            <p>{{ doctor.title }}</p>
        </div></div>
        <div class="col-md-4">
            <div class="form-group">
                <label for="department">所属科室:</label>
                <p>{{ doctor.dept.name }}</p>
            </div></div>
        <div class="col-md-4"><div class="form-group">
            <label for="department">科室房间号:</label>
            <p>{{ doctor.dept.room_number }}</p>
        </div></div></div>
        <hr><div class="row"><div class="col-12">
            <div class="form-group" style="text-align: center;">
                <label for="profile">简介:</label>
                <p style="font-size: 24px;">{{ doctor.profile }}</p>
            </div></div></div>
    </div>
```

4.2.4 提交预约和显示预约结果模块

- 与主页和预约挂号页面版图不同，这是单独设计的网页，主要实现了提交预约表单功能。用户点击想要预约的医生的 预约 按钮后就会跳转到这个 register_appoint.html 提交预约页面，选择想要预约的日期和时段，点击 提交预约 按钮后，将表单传给后端处理，就完成了一次预约。
- 预约完成后会跳转到 register_success.html 页面，表示预约成功。

核心代码：

```
<form method="post">
<h1>提交预约</h1>
<!-- 日期选择 -->
<div class="form-group">
    <label for="appointment_time">选择日期: </label>
    <input type="date" class="form-control" id="appointment_time" name="appointment_time">
</div>
<!-- 时段选择 -->
<div class="form-group">
    {% csrf_token %}
    <label for="period">选择时段: </label>
    <select class="form-control" id="period" name="period">
        {% for k, v in options %}
            <option value="{{ k }}>{{ v }}</option>
        {% endfor %}
    </select>
</div>
<div class="row justify-content-center">
    <button type="submit" class="btn btn-primary">提交预约</button>
    <a href="#" class="btn btn-primary" onclick="goBack()">返回</a>
</div>
</form>
```

- 此外，在前端使用JavaScript增加验证，如果选择的日期早于今天或选择的日期等于今天且选择的时段早于现在时间就会报错，并且不允许提交。

核心代码：

```
function validateDateTime() {
    var selectedDate = new Date(this.value);
    var today = new Date();

    // 设置选定日期和今天的时间部分为0
    selectedDate.setHours(0, 0, 0, 0);
    today.setHours(0, 0, 0, 0);

    if (selectedDate < today) {
        document.getElementById('date-error').innerText = '选择的日期必须不早于今天';
        this.setCustomValidity('选择的日期必须不早于今天');
    } else {
        document.getElementById('date-error').innerText = '';
        this.setCustomValidity('');
    }
}

document.getElementById('appointment_time').addEventListener('change', validateDateTime);

function validateTime() {
    var selectedDate = new Date(document.getElementById('appointment_time').value);
    var selectedPeriod = document.getElementById('period').value;
    var now = new Date();
    var currentHour = now.getHours();

    // 设置选定日期和今天的时间部分为0
    selectedDate.setHours(0, 0, 0, 0);
    now.setHours(0, 0, 0, 0);

    var periodStartHours = [10, 12, 16, 18];

    if (selectedDate.getTime() === now.getTime() && currentHour >= periodStartHours[selectedPeriod - 1]) {
        document.getElementById('time-error').innerText = '不能选择过去的时段';
        document.getElementById('period').setCustomValidity('不能选择过去的时段');
    } else {
        document.getElementById('time-error').innerText = '';
        document.getElementById('period').setCustomValidity('');
    }
}

// 绑定到两个事件
document.getElementById('period').addEventListener('change', validateTime);
document.getElementById('appointment_time').addEventListener('change', validateTime);
```

4.2.5 查看挂号记录模块

- 注册成功后，用户可以在首页点击 挂号记录 索引跳转到 `registrations.html` 页面查看自己的挂号记录。
- 页面内会显示用户的所有挂号记录，条目包括医生的姓名、挂号的时间、挂号时段以及这条预约的状态（已预约或预约已取消）。如果用户想取消某次预约，可以在对应条目里点击 取消预约 按钮取消；如果已经就诊完成，可以在结果栏里点击 查看详情 按钮查看医生反馈的就诊报告。
- 因为有的状态出现是互斥的，比如 取消预约 和 预约已取消，我们使用了 Django 自带的 ‘if else’ 判断语句进行条件判断，根据传入参数的值决定显示在页面的内容。

核心代码：

```





```

4.2.6 病历详情模块 (medical_record.html)

- 从挂号记录页面的 查看详情 按钮可以跳转到对应就诊记录里由医生填写的病历详情，通过传参的方式从数据库中得到相应数据并显示。提供了 返回上一页 按钮实现页面回退，病历里主要包含了患者姓名、医生姓名，以及由医生填写的症状、诊断结果和治疗方案。

核心代码：

```
<tbody>
<tr>
    <th style="width:100px;">患者姓名</th>
    <td>{{ medical_record.registration.patient.name }}</td>
</tr>
<tr>
    <th style="width:100px;">医生姓名</th>
    <td>{{ medical_record.registration.doctor.name }}</td>
</tr>
<tr>
    <th style="width:100px;">症状</th>
    <td>{{ medical_record.symptom }}</td>
</tr>
<tr>
    <th style="width:100px; padding:13px;">诊断结果</th>
    <td>{{ medical_record.diagnosis }}</td>
</tr>
<tr>
    <th style="width:100px; padding:13px;">治疗方案</th>
    <td>{{ medical_record.solution }}</td>
</tr>
</tbody>
```

4.2.7 个人中心模块 (personal.html)

- 我们提供了个人中心模块用来显示用户的的基础资料，包括姓名、用户名、密码、年龄和性别等个人信息。此外，如果不想要这个账户了，可以通过点击 注销 按钮注销账户，后端会把对应的登陆用户的记录删除。

核心代码：

```

<div class="container mt-5 patient-info-container">
<h2 class="text-center">个人中心</h2>
<div class="form-group">
    <label for="name">姓名: </label>
    <p>{{ patient.name }}</p>
</div>
<div class="form-group">
    <label for="username">用户名:</label>
    <p>{{ patient.username }}</p>
</div>
<div class="form-group">
    <label for="password">密码:</label>
    <p>{{ patient.password }}</p>
</div>
<div class="form-group">
    <label for="age">年龄:</label>
    <p>{{ patient.age }}</p>
</div>
<div class="form-group">
    <label for="gender">性别:</label>
    <p>{{ patient.get_gender_display }}</p>
</div>
<a href="#" class="btn btn-primary" onclick="goBack()">返回</a>
<a href="/logout" class="btn btn-warning">注销</a>
</div>

```

4.3 医生界面

4.3.1 首页模块 (home.html)

医生界面的首页和患者界面的几乎相同，只是导航栏的页面索引不一样。对于医生用户，我们提供了查看患者预约记录功能、查看药房信息以及个人中心界面。

```

<div class=" sidenav" style="bottom: 0;">
    <h5 style="color: #fff; margin: 30px; margin-top: 10.6px; font-size: 1.449em; letter-spacing: 1px;">
        <a href="/doctor/home" class="active">首页</a>
        <a href="/doctor/registrations">患者预约记录</a>
        <a href="/doctor/medicine">药房</a>
        <a href="/doctor/personal">个人中心</a>
    </h5>
</div>

```

4.3.2 查看患者预约记录模块 (registrations.html)

- 设计思路：
 - 这个界面主要用来给医生查看患者预约的记录和接诊。患者预约的信息包括患者姓名、预约时间、预约时段和状态，通过调用数据库数据传参显示。
 - 在用户提交预约后，医生可以在这里点击 接诊 按钮接诊，面诊结束后，点击 填写病历 按钮可以跳转到病历页面填写相关信息，最后点击 结束就诊 按钮，结束本次就诊。用户可以在自己的预约记录界面看到医生填写的病历信息。
- 核心代码：

```


| 患者姓名                            | 预约时间                                                   | 预约时段                                  | 状态                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 操作 |
|---------------------------------|--------------------------------------------------------|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| {{ registration.patient.name }} | {{ registration.registration_time date:"Y 年 m 月 d 日"}} | {{ registration.get_period_display }} | {{ registration.get_status_display }}</td><td> <div class="action-buttons"> {% if registration.status == 0 %} &lt;button class="btn btn-able" style="margin-right: 5px; margin-left: -35px;" onclick="acceptVisit(this, {{ registration.id }})"&gt;接诊&lt;/button&gt; {% else %} &lt;button class="btn btn-disable" style="margin-right: 5px; margin-left: -35px;"&gt;填写病历&lt;/button&gt; {% endif %}  {% if registration.status == 2 %} &lt;button class="btn btn-able" style="margin-right: 5px;" onclick="fillCase(this, {{ registration.id }})"&gt;填写病历&lt;/button&gt; {% else %} &lt;button class="btn btn-disable" style="margin-right: 5px;"&gt;结束就诊&lt;/button&gt; {% endif %} </div> |    |


```

4.3.3 填写病历模块 (medical_record.html)

医生可以通过点击预约记录的 填写病历 按钮进入相应患者的病历填写页面，然后通过提交表单的形式，填入患者的症状、诊断结果和治疗方案，点击 保存 后，将表单传递到后端处理。此外，设置了

.error 类用于显示表单字段验证过程中的错误信息，提高用户体验感和表单的交互性。

核心代码：

```
<form method="post" style="margin-top: 30px;">
  {% csrf_token %}
  <table><tbody><tr>
    <th style="width:100px;">患者姓名</th>
    <td><span class="name">{{ patient.name }}</span></td></tr>
  <tr><th style="width:100px;">症状</th><td>
    <textarea name="symptom">{{ form.symptom.value }}</textarea>
    <span class="error">{{ form.symptom.errors.0 }}</span>
  </td></tr>
  <tr><th style="width:100px;">诊断结果</th>
    <td><textarea name="diagnosis">{{ form.diagnosis.value }}</textarea>
    <span class="error">{{ form.diagnosis.errors.0 }}</span>
  </td></tr>
  <tr><th style="width:100px;">治疗方案</th>
    <td><textarea name="solution">{{ form.solution.value }}</textarea>
    <span class="error">{{ form.solution.errors.0 }}</span></td></tr>
  <tr><th style="width:100px;"></th>
    <td><button type="submit" class="btn btn-primary">保存</button></td></tr>
  </tbody></table>
</form>
```

4.3.4 药品查询模块 (medicine.html)

- 设计思路：

相当于设计了一个药房方便医生查询药品的相关信息。在上方的搜索栏处，医生用户可以通过输入药品名和选择药品类型查询想要了解的药品。医生用户的输入信息会通过 form 表单传到后端数据库，后台得到网页提交的表单后，查询数据库将符合要求的信息返回给网页。搜索栏下方可以显示筛选后的查询结果，结果包含了药物名、类型、生产厂商和价格。同时考虑到药物种类比较多，设置了一个分类器。

- 核心代码：

```

<div class="container" style="margin-top: 23px;">
<form method="get">
    <label for="name" style=" font-weight: bold;">药物名: </label>
    <input type="text" id="name" name="name" value="{{ name_query }}" style="margin-right: 10px;"

    <label for="type" style=" font-weight: bold;">类型: </label>
    <select id="type" name="type">
        <option value="" {% if not type_query %}selected{% endif %} style="color: #999;">选择类型
        {% for type in types %}
            <option value="{{ type.0 }}"{% if type.0 == type_query %}selected{% endif %}>
                {{ type.1 }}
            </option>
        {% endfor %}
    </select>
    <button type="submit" class="btn btn-primary">搜索</button>
</form>

<table class="table table-striped">
    <thead><tr>
        <th scope="col">药物名</th>
        <th scope="col">类型</th>
        <th scope="col">生产厂商</th>
        <th scope="col">价格</th>
    </tr></thead>
    <tbody>
        {% for medicine in medicines %}
        <tr>
            <td>{{ medicine.name }}</td>
            <td>{{ medicine.get_type_display }}</td>
            <td>{{medicine.producer}}</td>
            <td>{{ medicine.price }}￥</td>
        </tr>
        {% endfor %}
    </tbody>
</table>

```

4.3.5 医生个人中心模块 (personal.html)

跟患者的个人中心模块很相似，依然是实现了简单的用户个人信息显示功能，包括医生姓名、用户名、登陆密码、职称以及所属科室，通过从数据库中读取数据实现。通过 返回 按钮可以返回上一个界面，点击 注销 可以将该条医生信息从数据库删除。

核心代码：

```
<div class="container mt-5 doctor-info-container">
    <h2 class="text-center">个人中心</h2>
    <div class="form-group">
        <label for="name">姓名: </label>
        <p>{{ doctor.name }}</p></div>
    <div class="form-group">
        <label for="username">用户名:</label>
        <p>{{ doctor.username }}</p></div>
    <div class="form-group">
        <label for="password">密码:</label>
        <p>{{ doctor.password }}</p></div>
    <div class="form-group">
        <label for="title">职称:</label>
        <p>{{ doctor.title }}</p></div>
    <div class="form-group">
        <label for="dept">所属科室:</label>
        <p>{{ doctor.dept.name }}</p></div>
    <a href="#" class="btn btn-primary" onclick="goBack()">返回</a>
    <a href="/logout" class="btn btn-warning">注销</a>
</div>
```

5. 后端实现

5.1 登录注册界面

5.1.1 登录模块

- **LoginForm (登录表单) 类:**

使用Django的Form类继承定义登录表单LoginForm，表单包括username和password字段。

- **login (登录) 视图函数:**

对应于 login/login 页面，处理用户登录的逻辑。在login函数中，若用户以 GET 访问网页则提供表单。否则创建Loginform实例接收 POST 的表单数据，调用 is_valid 方法进行初步校验，然后根据用户点击按钮的 name 属性判断登录类型。接着从数据库查询出该用户，比对用户名和密码，如果不对输出调用 add_error 方法添加错误，否则将用户的id和类型存入session并重定向到home页面。

cookie和session机制：

我们的代码采用了cookie和session机制。Django的session机制是一种用于在Web应用程序中存储和检索用户数据的机制。它基于cookie或数据库，用于在不同的请求之间保持用户的状态信息。

在Django视图函数中，可以通过request对象的session属性来访问或设置session数据。设置session会在用户浏览器保存cookie，然后在数据库中保存对应session数据。之后每当用户访问页面时都会带上cookie，从而后台可以根据cookie在数据库种查到相应session信息。我们在用户登录时设置cookie和session。

使用 `request.session['key'] = 'value'` 来设置session数据，此处我们使用 `request.session["info"] = {'id': user.id, 'login_type': login_type}` 设置id和登陆类型的cookie。

cookie可帮助完成一些页面的逻辑。例如个人信息页面是需要知道当前登录者信息的，这时就必须用到cookie传递给后台，后台才能知道是谁正在使用，进而查询到他/她的个人信息返回给前端。

使用 `value = request.session.get('key')` 来获取session数据。

使用 `request.session.clear()` 来清除cookie。

```

class LoginForm(forms.Form):
    username = forms.fields.CharField(
        required=True,
        label="用户名",
        widget=forms.widgets.TextInput({"placeholder": "请输入用户名", "class": "form-control"})
    )
    password = forms.fields.CharField(
        required=True,
        label="密码",
        widget=forms.widgets.PasswordInput({"placeholder": "请输入密码", "class": "form-control"})
    )

def login(request):
    if request.method == "GET":
        form = LoginForm()
        return render(request, "login/login.html", {"form": form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        login_type = request.POST.get('login_type')
        user_model = Patient if login_type == 'patient_login' else Doctor
        user = user_model.objects.filter(**form.cleaned_data).first()

        if user:
            # 验证成功，跳转到主页
            request.session["info"] = {'id': user.id, 'login_type': login_type}
            if login_type == 'patient_login':
                return redirect("/patient/home/")
            else:
                return redirect("/doctor/home/")
        else:
            # 验证失败，增加错误并重新渲染
            form.add_error("password", "用户名或密码不正确")
            return render(request, "login/login.html", {"form": form})

    return render(request, "login/login.html", {"form": form})

```

5.1.2 注册模块

- **PatientForm (患者信息表单) 类:**

使用Django的ModelForm类继承定义患者模型表单PatientForm，用于构建Patient模型的表单。它定义了需要在表单中包含的字段 ('name', 'age', 'username', 'password', 'gender')，指定了

password字段的展示形式为密码输入框，并定义 `clean_username` 方法添加了验证逻辑，确保注册时的用户名在数据库中的唯一性。在初始化方法中，将姓名、用户名和密码字段标记为必填项。

- **enroll (注册) 视图函数:**

对应于 `login/enroll` 页面，处理用户注册的逻辑。当接收到POST请求时，创建PatientForm实例，调用 `is_valid` 方法验证用户输入，如果有效则调用 `save` 方法保存患者信息到数据库并重定向到登录页面。在GET请求下，创建空的PatientForm实例，用于渲染空的注册表单。

error机制:

我们在登录、注册、填写病历等地方添加了error机制，在用户提交含有非法值的表单时反馈错误信息给用户，以做到良好的用户交互。

前端在表单相应位置添加`{{field.errors.0}}`，当相应field无error时它不会被显示出来，当有error时会打印出第一个error值。

后端主要涉及到的三种方法：

- `clean_username`： 定义在Form或ModelForm里的方法，其中username可换成任一字段的名字。可以在该方法内定义初步判断逻辑，若违反则可发起 `ValidationError` 在该字段上添加 error信息。
- `is_valid`： Form或ModelForm的方法，在视图函数中使用，会根据字段特性和用户自定义的 `clean` 方法检验是否合法，返回布尔值。
- `add_error("password", "用户名或密码不正确")`： Form或ModelForm的方法，在视图函数中使用，可以在指定字段加上错误信息。

```

class PatientForm(forms.ModelForm):
    class Meta:
        model = Patient
        fields = ['name', 'age', 'username', 'password', 'gender']
        widgets = {
            'password': forms.PasswordInput(),
        }

    def clean_username(self):
        username = self.cleaned_data['username']
        # 检查数据库中是否存在相同的用户名
        if Patient.objects.filter(username=username).exists():
            raise ValidationError("该用户名已被使用, 请重新填写")
        return username

    def __init__(self, *args, **kwargs):
        super(PatientForm, self).__init__(*args, **kwargs)
        self.fields['name'].required = True
        self.fields['username'].required = True
        self.fields['password'].required = True

def enroll(request):
    if request.method == 'POST':
        form = PatientForm(request.POST)
        if form.is_valid():
            patient = form.save()
            return redirect('/login/')
    else:
        form = PatientForm()

    return render(request, "login/enroll.html", {"form": form})

```

5.1.3 注销模块

- **logout (注销) 视图函数:**

对应于 login/logout 链接, 当用户点击“注销”按钮触发, 清空当前用户的会话信息 (session), 然后重定向到登录页面。

```

def logout(request):
    request.session.clear()
    return redirect('/login/')

```

5.2 患者界面

5.2.1 主页模块

- **home (主页) 视图函数:**

对应于 `patient/home` 页面。当用户访问主页，返回主页的html。

```
def home(request):  
    return render(request, 'patient/home.html')
```

5.2.2 预约挂号模块

- **register 视图函数:**

对应于 `patient/register` 页面，即“预约挂号”界面。首先获取所有医生和科室的信息，然后根据请求中的查询值进行过滤。`name_query`用于按医生姓名过滤，`dept_query`用于按科室过滤，从数据库筛选出符合条件的项。页面还使用Django的Paginator进行结果分页，每页显示10个医生。最后，将医生列表数据、查询值和科室列表传递给注册页面进行渲染（查询值是为了在搜索栏中保存上次搜索的条件）。

分页机制：

Paginator是Django框架提供的用于分页的工具。它允许将查询结果分成固定数量的页面，以便在用户界面上进行分页显示。Paginator接受一个可迭代对象（例如QuerySet）和每页显示的记录数量作为参数，然后可以通过它的方法和属性来获取分页的信息。

在后端使用

```
paginator = Paginator(doctors, 10)  
page_number = request.GET.get('page')  
doctors = paginator.get_page(page_number)
```

Paginator会进行分页，每页10项，然后根据url参数'page'获取当前页码数，最后调用 `get_page` 方法获取分页后的列表，后面传递给前端。前端可用

```
page_obj.has_previous、page_obj.previous_page_number、page_obj.paginator.num_pages、  
page_obj.has_next、page_obj.next_page_number 等方法实现分页器。
```

```

def register(request):
    doctors = Doctor.objects.all()
    departments = Department.objects.all()

    # 获取查询值
    name_query = request.GET.get('name', '')
    dept_query = request.GET.get('department', '')

    if name_query:
        doctors = doctors.filter(name__contains=name_query)

    if dept_query:
        doctors = doctors.filter(dept=dept_query)

    # 分页
    paginator = Paginator(doctors, 10)
    page_number = request.GET.get('page')
    doctors = paginator.get_page(page_number)

    # 转换为int在前端进行比较
    dept_query = int(dept_query, base=10) if dept_query != '' else dept_query

    return render(request, 'patient/register.html', {'doctors': doctors, 'name_query': name_query})

```

- **register_success (预约成功) 视图函数:**

对应于 patient//register/success 页面，返回患者挂号成功的页面。

- **register_appoint (挂号预约) 视图函数:**

对应于 patient/register/appoint/<int:doctor_id>/ 页面，处理患者对医生的挂号预约逻辑。该函数首先在url中获取doctor_id参数（url传参），在数据库中通过doctor_id获取医生实例，然后根据请求方法进行相应的处理。

- 如果是POST请求，表示患者提交了挂号预约的表单。在这种情况下，从POST数据中获取预约时间(appointment_time)和时段(period)，将预约时间字符串转换为datetime对象，然后通过session获取患者实例，最后创建一个Registration实例，表示患者对医生的挂号预约，并保存到数据库中。最后，重定向到挂号成功页面。
- 如果是GET请求，表示患者正在访问挂号预约的页面。在这种情况下，获取挂号时段的选项(options)，然后渲染html页面，向页面传递医生实例和挂号时段的选项。

```

def register_sucess(request):
    return render(request, 'patient/register_success.html')

def register_appoint(request, doctor_id):
    doctor = get_object_or_404(Doctor, pk=doctor_id)

    if request.method == 'POST':
        appointment_time = request.POST.get('appointment_time')
        period = request.POST.get('period')

        # 将字符串时间转换为datetime对象
        appointment_datetime = datetime.strptime(appointment_time, '%Y-%m-%d')

        # 通过session获得患者实例
        patient = Patient.objects.get(id=request.session.get('info')['id'])

        # 创建挂号记录
        registration = Registration.objects.create(
            doctor=doctor,
            patient=patient,
            registration_time=appointment_datetime,
            period=period,
            status=0,
        )

    return redirect('/patient/register/success/')

options = Registration.PERIOD_CHOICES
return render(request, 'patient/register_appoint.html', {'doctor': doctor, 'options': options})

```

5.2.3 挂号记录模块

- **registrations (患者挂号记录) 视图函数:**

对应于 `patient/registrations` 页面，用于显示患者的挂号记录。首先，获取session的患者ID，然后从数据库中获取患者实例。接着，通过用该患者实例从数据库中筛选出该患者的所有挂号记录。使用Django的Paginator对挂号记录进行分页，每页显示10条记录。最后，将分页后的挂号记录传递给前端页面进行渲染。

- **cancel_registration (取消挂号) 视图函数:**

对应于 `patient/cancel_registration/<int:registration_id>` 链接，它在患者点击“取消预约”按钮时跳转，处理患者取消挂号的逻辑。首先，从url参数中获得`registration_id`并用它从数据库获取挂

号记录实例，然后检查该挂号记录的状态。如果状态为0（已挂号），则将状态改为1（已取消）并保存。最后，重定向到患者挂号记录页面。

- **medical_record (患者病历信息) 视图函数:**

对于 `patient/medical_record/<int:registration_id>/` 页面，显示患者的病历信息。函数首先接收 `session` 中的 `registration_id` 并用其在数据库中获取挂号记录实例，然后传递给前端页面。

```
def registrations(request):  
    patient_id = request.session.get('info')['id']  
    patient = Patient.objects.get(id=patient_id)  
    registrations = Registration.objects.filter(patient=patient)  
  
    # 分页  
    paginator = Paginator(registrations, 10)  
    page_number = request.GET.get('page')  
    registrations = paginator.get_page(page_number)  
  
    return render(request, 'patient/registrations.html', {'registrations': registrations})  
  
def cancel_registration(request, registration_id):  
    registration = get_object_or_404(Registration, pk=registration_id)  
    if registration.status == 0:  
        registration.status = 1  
        registration.save()  
  
    return redirect('/patient/registrations/')  
  
def medical_record(request, registration_id):  
    registration = get_object_or_404(Registration, pk=registration_id)  
    medical_record = MedicalRecord.objects.get(registration=registration)  
    return render(request, 'patient/medical_record.html', {'medical_record': medical_record})
```

5.2.4 医生详情模块

- **personnel (医院人员信息) 视图函数:**

对于 `patient/personnel` 页面，即“医生详情”界面。调用 `Doctor.objects.all()` 和 `Department.objects.all()` 获取所有医生和科室的信息，并将医生和科室的信息传递给前端。

- **profile (医生个人信息) 视图函数:**

对于 `patient/profile/<int:doctor_id>/` 页面，用于显示医生个人信息。首先通过 `url` 参数的 `doctor_id` 获取医生实例，如果医生存在，则将医生实例传递给 `patient/profile.html` 页面进行渲染。如果医生不存在，返回 404 页面。

```
def personnel(request):
    doctors = Doctor.objects.all()
    departments = Department.objects.all()
    return render(request, 'patient/personnel.html', {'doctors': doctors, 'departments': departments})

def profile(request, doctor_id):
    doctor = get_object_or_404(Doctor, pk=doctor_id)
    return render(request, 'patient/profile.html', {'doctor': doctor})
```

5.2.5 个人信息页面

- **personal (患者个人信息) 视图函数:**

对应于 patient/personal 页面，即“个人信息”页面。通过session的信息获取患者的ID，然后通过在数据库中获取患者实例，最后传递给前端页面。

```
def personal(request):
    patient_id = request.session.get('info')['id']
    patient = Patient.objects.get(id=patient_id)
    return render(request, 'patient/personal.html', {'patient': patient})
```

5.3 医生界面

5.3.1 首页模块

与患者界面的首页模块类似。

5.3.2 预约记录模块

- **registrations (医生挂号记录) 视图函数:**

对应于 doctor/registrations 页面，即“挂号记录”页面，用于显示医生的挂号记录。首先，通过网页session获取医生的ID并在数据库中获取该医生实例。接着，在数据库中筛选出该医生的所有挂号记录，并按照挂号时间和时段进行排序。使用Django的Paginator对挂号记录进行分页，每页显示10条记录。最后，将分页后的挂号记录传递给前端进行渲染。

```
def registrations(request):
    doctor_id = request.session.get('info')['id']
    doctor = Doctor.objects.get(id=doctor_id)
    registrations = Registration.objects.filter(doctor=doctor).order_by('registration_time', 'pe')

    # 分页
    paginator = Paginator(registrations, 10)
    page_number = request.GET.get('page')
    registrations = paginator.get_page(page_number)

    return render(request, 'doctor/registrations.html', {'registrations': registrations})
```

- **MedicalRecordForm (病历表单类) :**

使用Django的ModelForm类定义MedicalRecordForm，用于构建处理病历信息的表单。该表单关联的模型是MedicalRecord。在Meta类中使用exclude排除了registration字段。

- **medical_record (医生病历信息) 视图函数:**

对应于 `doctor/medical_record/<int:registration_id>/` 页面，该页面由医生点击“填写病历”按钮跳转，是医生填写病历单的页面。首先，通过url参数中的registration_id获取挂号记录实例。然后，尝试通过该挂号记录实例获取关联的病历信息并赋值给 `medical_record`，如果病历信息不存在，将 `medical_record` 设置为None。

在GET请求中，如果病历信息已存在，即 `medical_record` 不为None，将使用MedicalRecordForm初始化表单，并传递给页面显示。如果病历信息不存在，将form设置为None。

在POST请求中，如果表单数据有效，将表单保存，并将病历信息关联到挂号记录，最后重新渲染页面，向页面传递表单和患者实例。

```

class MedicalRecordForm(forms.ModelForm):
    class Meta:
        model = MedicalRecord
        exclude = ['registration']

def medical_record(request, registration_id):
    registration = get_object_or_404(Registration, pk=registration_id)

    try:
        medical_record = MedicalRecord.objects.get(registration=registration)
    except MedicalRecord.DoesNotExist:
        medical_record = None

    patient = registration.patient

    if request.method == 'GET':
        if medical_record == None:
            form = None
        else:
            form = MedicalRecordForm(instance=medical_record)
        return render(request, 'doctor/medical_record.html', {'form': form, 'patient': patient})

    form = MedicalRecordForm(request.POST, instance=medical_record)
    if form.is_valid():
        medical_record = form.save(commit=False)
        medical_record.registration = registration
        medical_record.save()
    return render(request, 'doctor/medical_record.html', {'form': form, 'patient': patient})

```

- **accept (接诊) 视图函数:**

医生点击“接诊”时触发。通过url的registration_id参数获取挂号记录实例，将该挂号记录的状态设置为2（就诊中），然后在数据库中该挂号记录。

- **finish (完成就诊) 视图函数:**

医生点击“接诊”时触发。通过url的registration_id参数获取挂号记录实例，将该挂号记录的状态设置为3（就诊结束），然后在数据库中更新该挂号记录。

```
def accept(request, registration_id):
    registration = get_object_or_404(Registration, pk=registration_id)
    registration.status = 2
    registration.save()
    return redirect('/doctor/registrations')

def finish(request, registration_id):
    registration = get_object_or_404(Registration, pk=registration_id)
    registration.status = 3
    registration.save()
    return redirect('/doctor/registrations')
```

5.3.3 药房模块

- **medicine (药房) 视图函数:**
 - 对应于doctor/medicine页面，即“药房”页面。medicine的处理逻辑和患者的挂号很像。首先，通过Medicine.objects.all()获取所有药物的信息，并通过Medicine.TYPE_CHOICES获取药物的类型选项。接着，通过GET请求获取查询参数name_query和type_query。
 - 如果存在name_query，则通过medicines.filter筛选出药物名称包含name_query的药物。如果存在type_query，则通过medicines.filter筛选出类型等于type_query的药物。
 - 使用Django的Paginator对药物信息进行分页，每页显示10条记录。最后，将分页后的药物信息、查询参数和药物类型选项传递给doctor/medicine.html页面进行渲染。

```

def medicine(request):
    medicines = Medicine.objects.all()
    types = Medicine.TYPE_CHOICES

    # 获取查询值
    name_query = request.GET.get('name', '')
    type_query = request.GET.get('type', '')

    if name_query:
        medicines = medicines.filter(name__contains=name_query)

    if type_query:
        medicines = medicines.filter(type=type_query)

    # 分页
    paginator = Paginator(medicines, 10)
    page_number = request.GET.get('page')
    medicines = paginator.get_page(page_number)

    # 转换为int在前端进行比较
    type_query = int(type_query, base=10) if type_query != '' else type_query

    return render(request, 'doctor/medicine.html', {'medicines': medicines, 'name_query': name_query})

```

5.3.4 个人信息模块

与患者界面的个人信息模块类似。

5.4 中间件

设计一个Django中间件，用于实现用户认证和权限控制，保证安全性，应实现以下功能：

1. 当用户输入根目录url时，跳转到登录界面
2. 当用户未登录时，不允许通过直接输入url进入到除登录或注册的其他界面
3. 当用户登录后，允许其通过url进入到界面，但患者只能进入患者相关界面，医生只能进入医生相关界面

具体设计思路：

定义 `AuthMiddleware` 类，该类继承 `MiddlewareMixin` 类。定义 `process_request` 方法，该方法是Django中间件的一个钩子函数，会在每个请求处理之前被调用。在 `settings.py` 的 `MIDDLEWARE` 列表注册该中间件。

1. 排除不需要登录就可以访问的页面：

- 通过 `request.path_info` 获取当前请求的URL路径。
- 如果当前请求的路径是 `/login/` 或 `/enroll/`，则直接通过，不进行后续认证和权限控制，允许用户访问登录和注册页面。

2. 读取session并检查登录状态：

- 通过 `request.session.get('info')` 读取session中的用户信息。如果用户信息不存在，说明用户未登录，将用户重定向到登录界面（`/login/`）。

3. 验证登录方式和URL权限：

- 如果用户已经登录（session中存在用户信息），则检查用户的登录方式（`login_type`）和当前请求的URL路径。
- 如果登录方式为患者登录（`patient_login`）且请求路径包含 `/patient/`，或者登录方式为医生登录（`doctor_login`）且请求路径包含 `/doctor/`，则允许用户访问，直接返回，不进行后续重定向。
- 如果以上条件不满足，说明用户的登录方式与请求的URL路径不匹配，将用户重定向到登录界面。

4. 未登录用户重定向：

- 如果用户未登录或登录方式不匹配，通过 `redirect('/login/')` 将用户重定向到登录页面。

代码如下：

```

class AuthMiddleware(MiddlewareMixin):
    def process_request(self, request):
        # 截获请求的url路径
        path = request.path_info

        # 排除不需登录就可以访问的页面
        if path == '/login/' or path == '/enroll/' or path == '/login' or \
            path == '/enroll' or path.startswith('/admin'):
            return

        # 读取session, 如果没有就回到登陆界面
        info = request.session.get('info')
        if not info:
            return redirect('/login/')

        # 如果登录方式符合请求url就允许登录
        login_type = info['login_type']
        if login_type == 'patient_login' and '/patient/' in path:
            return
        elif login_type == 'doctor_login' and '/doctor/' in path:
            return

        # 没有登陆过就回到登陆界面
        return redirect('/login/')

```

总而言之，该中间件实现了对用户登录状态和权限的简单检查，确保用户在访问需要登录的页面时必须先登录，并且只能访问与其登录方式相匹配的页面。

5.5 管理员

使用Django自带的管理系统 /admin。Django 的 /admin 是一个内置的管理界面，用于方便地管理数据库中的数据和进行后台操作，注册超级管理员即可使用。主要特点和功能包括：

- 数据管理：**提供了对应每个模型的数据管理界面，可以查看、添加、修改和删除数据库中的记录。
- 用户权限管理：**支持对用户和用户组进行管理，可以定义用户权限，控制其对系统的访问和操作权限。
- 模型可视化：**显示了项目中定义的所有模型，使开发者和管理员可以轻松查看和操作数据库模型。
- 后台操作：**提供了一些系统级别的后台操作，如修改站点设置、查看日志等。
- 定制性强：**可以通过自定义 admin.py 文件进行定制，包括调整显示字段、过滤器、搜索字段等，以适应项目需求。

使用 /admin 界面需要管理员权限，通常在开发和调试阶段，可以使用超级用户账户登录。这个管理界面为开发者和系统管理员提供了便捷的工具，用于快速查看和管理项目的数据。

6. 功能演示

6.1 登录与注册界面

- **登录界面：**

- 输入用户名和密码登录，有“医生登录”和“患者登录”两种登录方式

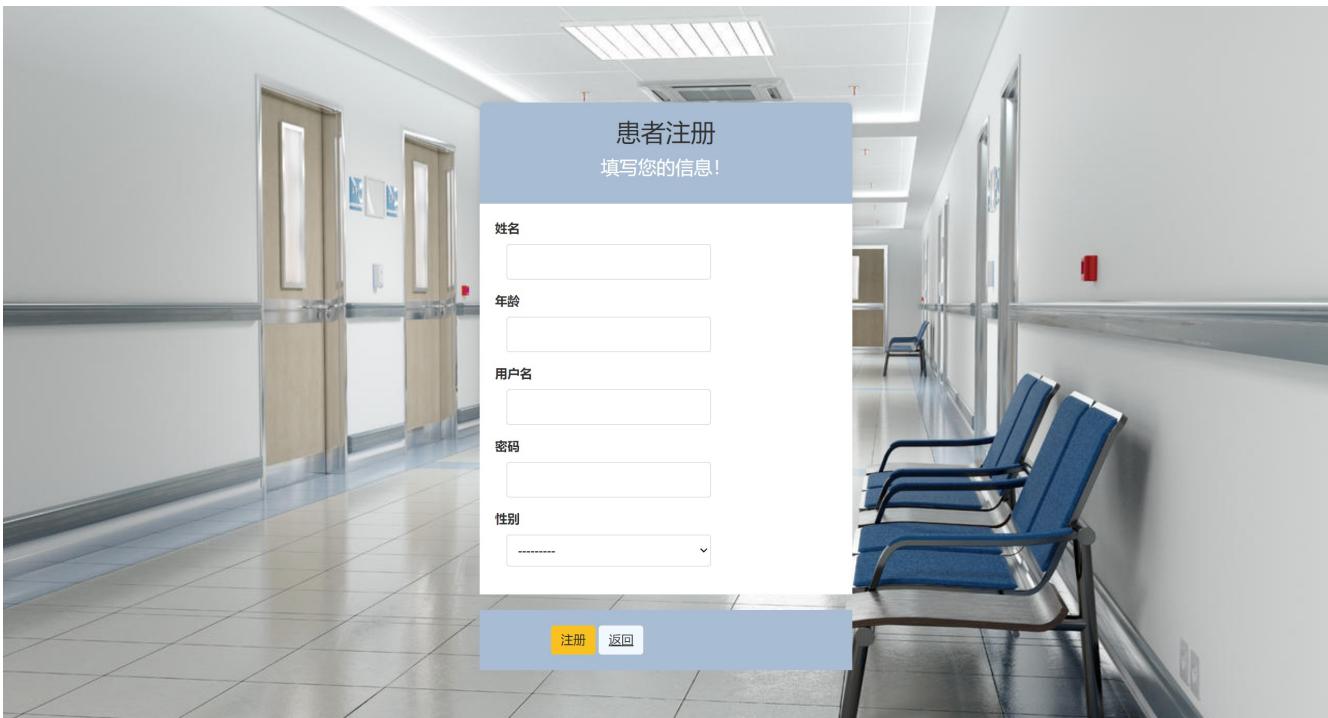


- 非法输入会有错误提示

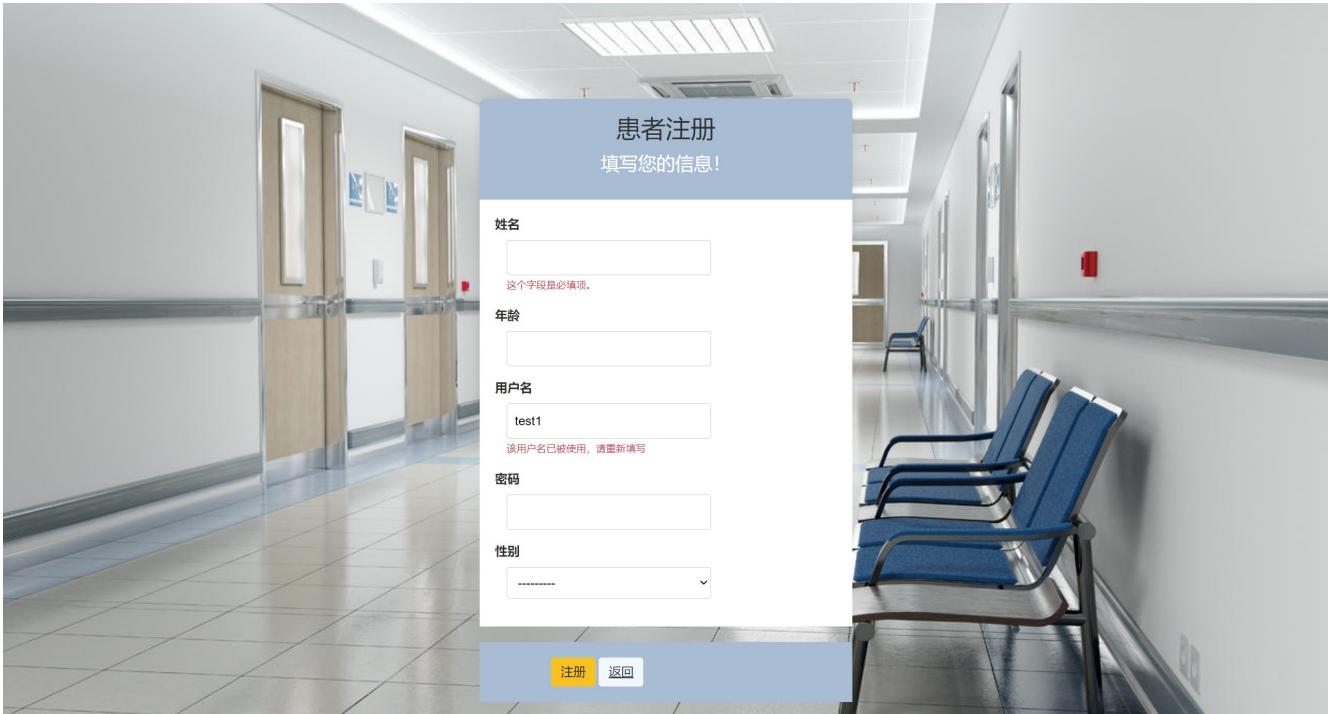


• 注册界面

- 填写个人信息注册



- 非法输入会有错误提示



6.2 患者界面

- 首页:

- 展示医院的基本信息

患者界面

- [首页](#)
- [预约挂号](#)
- [医生介绍](#)
- [挂号记录](#)
- [个人中心](#)

医院简介

我院拥有心血管内科、心血管外科、内分泌科、临床护理、呼吸内科、神经内科、眼科、肾病科、感染科、急诊科、医学影像科等国家临床重点专科建设项目11个，广东省临床重点学科18个，院士工作站5个，国家级培训中心17个，广东省防控、诊疗与研究中心42个。

拥有高级职称专家720人，国管、省管专家62人，博士硕士导师142人，广东省学术技术带头人32人，国家级学术团体常委以上79人，广东省医学会专业分会主委、副主任116人。先后荣获国家自然科学基金等项目53项，省部级科研项目397项，获广东省科技进步一等奖2项。发表学术论文3856篇，其中SCI文章近300篇，中华核心400余篇。

医院秉承“仁爱、博学、严谨、卓越”的医院精神，坚持“人民医院服务人民”的宗旨，始终服务在人民群众生命健康的前线。先后被评为全国百姓放心示范医院、全国卫生系统先进集体、全国医院文化建设先进单位、等荣誉称号。连续三届蝉联全国文明单位。连续8年荣获行风评议先进单位，被评为“人民群众满意医院”。

我们的服务

健康体检

健康医学部2005年成立，于2012年9月17日搬入新门诊大楼9-11层，总面积达5400平方米。配有专用通道、直达电梯，体检区域与门诊患者分开，避免交叉感染。宽敞、明亮、整洁。现代化的体检大厅充分改善了体检者的受检环境。医院聚集了大批临床经验丰富、专业知识渊博、工作严谨的医学人才。在完成日常诊治的同时，轮流参加健康体检工作，为受检者提供高品质的体检服务。

远程医疗

无论您身在何处，我们的远程医疗服务将为您提供方便快捷的医疗咨询和治疗选择。您可以在家中与专业医生进行实时在线咨询，获取医学建议，甚至进行远程诊断。我们的远程医疗团队由经验丰富的医生和护理人员组成，他们将倾听您的症状，提供专业的医学建议，并协助您进行治疗计划。无论是常规的健康咨询、慢性病管理，还是急症病症的诊疗都能够满足您的需求。

专科专病

我院专门开设了疑难病会诊中心，用来解决病情复杂、诊断困难的医学难题，为患者提供更深入、更专业的医学建议。包括但不限于胰腺疾病疑难会诊、胸部疾病疑难病会诊、炎性肠病疑难病会诊、垂体疾病疑难病会诊等。

联系我们:

地址：广州市番禺区大学城至善医院
E-mail：info@example.com
电话总机：020-12345678

- 预约挂号:

- 可以通过医生姓名或者科室来筛选医生（支持模糊搜索），点击对应医生的预约按钮进入预约界面

患者界面

医生姓名: 科室: 搜索

医生姓名	科室	操作
穆璐	内科	<input type="button" value="预约"/>
赵雪	精神科	<input type="button" value="预约"/>
程旭	儿科	<input type="button" value="预约"/>
陈倩	内科	<input type="button" value="预约"/>
张三	口腔	<input type="button" value="预约"/>
张鹏	骨科	<input type="button" value="预约"/>
陈婷	外科	<input type="button" value="预约"/>
施玲	妇科	<input type="button" value="预约"/>
刘秀梅	耳鼻咽喉科	<input type="button" value="预约"/>
余丽	传染科	<input type="button" value="预约"/>

«首页 上一页 第1/2页 下一页 尾页»

- 选择日期和时段后提交预约

提交预约

选择日期:

选择时段:

- 选择日期和时段非法会报错

提交预约

选择日期:

年/月/日



选择时段:

10:00-12:00

! 请填写此字段。

提交预约

返回

提交预约

选择日期:

2024/01/13



选择的日期必须不早于今天

选择时段:

10:00-12:00

提交预约

返回

提交预约

选择日期：

2024/01/14



选择时段：

10:00-12:00



不能选择过去的时段

提交预约

返回

- 预约成功后跳转到成功页面

预约成功！

[返回主页](#)

• 医生介绍：

- 按科室展示所有医生，点击医生的名字可以进入医生详情界面

患者界面

- 首页
- 预约挂号
- 医生介绍**
- 挂号记录
- 个人中心

口腔	张三 朱慧
骨科	张鹏
内科	穆璐 陈倩
外科	陈婷
妇科	施玲
儿科	程旭
耳鼻咽喉科	刘秀梅
精神科	赵雪
传染科	余丽

- 医生详情界面可以看到医生的职称，科室，科室房间号，以及简介信息

详情介绍

朱慧

职称: 副主任	所属科室: 口腔	科室房间号: 101
------------	-------------	---------------

简介:
口腔科副主任

返回上一页

- **挂号记录:**

- 可以看到已经预约的挂号记录，当医生未开始接诊时（状态为已挂号），可以点击取消预约来取消

患者界面

- 首页
- 预约挂号
- 医生介绍
- 挂号记录
- 个人中心

医生姓名	挂号时间	挂号时段	状态	操作	诊断结果
穆璐	2023年12月31日	8:00-10:00	已取消	预约已取消	
张三	2023年12月29日	8:00-10:00	已取消	预约已取消	
张三	2023年12月29日	8:00-10:00	已取消	预约已取消	
张三	2023年12月31日	8:00-10:00	就诊结束		查看详情
张三	2023年12月30日	10:00-12:00	就诊结束		查看详情
穆璐	2024年01月07日	8:00-10:00	已取消	预约已取消	
张三	2023年12月16日	8:00-10:00	就诊中		
张三	2024年01月03日	16:00-18:00	已挂号	取消预约	
施玲	2024年01月07日	8:00-10:00	已挂号	取消预约	
张三	2023年12月16日	14:00-16:00	已挂号	取消预约	

[« 首页](#) [上一页](#) [第 1 / 1 页](#) [下一页](#) [尾页 »](#)

- 当医生填写好病历，并结束就诊后，可以点击查看详情来查看诊断结果

病例详情

[返回上一页](#)

患者姓名	test1
医生姓名	施玲
症状	患者不分场合表现为经常困乏思睡，出现不同程度、不可抗拒的入睡。过多的睡眠引起显著痛苦或职业、社交等社会功能和生活质量的下降。有认知功能方面的改变，表现为近事记忆减退，思维能力下降，学习新事物能力下降。
诊断结果	嗜睡症
治疗方案	严格作息时间，白天有意识地让患者小睡，养成良好的生活习惯。多运动要多参加体育活动，每天不少于一小时，使自己的身心得到兴奋。心理调节要有积极的生活态度，每天给自己制定好生活学习计划，认真努力完成等。

- 个人中心：
 - 可以看到患者的个人信息，可进行注销账号操作

个人中心

姓名: test1

用户名: test1

密码: 123456

年龄: 1

性别: 男

[返回](#) [注销](#)

6.3 医生界面

• 首页:

- 展示医院的基本信息

医生界面

首页 患者预约记录 药房 个人中心

医院简介

我院拥有心血管内科、心脑血管外科、内分泌科、临床护理、呼吸内科、神经内科、眼科、肾病科、感染科、急诊科、医学影像科等国家临床重点专科建设项目11个，广东省临床重点学科18个，院士工作站5个，国家级培训中心17个，广东省质控、诊疗与研究中心42个。

拥有一级职称专家720人，国管、省管专家62人，博士硕士导师142人，广东省学术技术带头人32人，国家级学术团体常委以上79人，广东省医学会专业分会主委、副主委116人。先后荣获国家自然科学基金项目53项，省部级科研项目397项，获广东省科技进步一等奖2项，发表学术论文3856篇，其中SCI文章近300篇，中华核心400余篇。

医院秉承“仁爱、博学、严谨、卓越”的医院精神，坚持“人民医院服务人民”的宗旨，始终服务于人民群众生命健康的前线。先后被评为全国百姓放心示范医院、全国卫生系统先进集体、全国医院文化建设先进单位、等荣誉称号。连续三届蝉联全国文明单位，连续8年荣获行风评议先进单位，被评为“人民群众满意医院”。

我们的服务

健康体检

健康医学部2005年成立，于2012年9月17日搬入新门诊大楼9-11层，总面积达5400平方米。配有专用通道、直达电梯，体检区域与门诊患者分开，避免交叉感染。宽敞、明亮、整洁、现代化的体检大厅充分改善了体检者的受检环境。医院聚集了大批临床经验丰富、专业知识渊博、工作严谨的医学人才。在完成日常诊治的同时，轮流参加健康体检工作，为受检者提供高品质的体检服务。

远程医疗

无论您身在何处，我们的远程医疗服务将为您提供方便快捷的医疗咨询和治疗选择。您可以在家中与专业医生进行实时在线咨询，获取医学建议，甚至进行远程诊断。我们的远程医疗团队由经验丰富的医生和护理人员组成，他们将倾听您的症状，提供专业的医学建议，并协助您进行治疗计划。无论是常规的健康咨询、慢性病管理，还是急性病症的诊疗都能够满足您的需求。

专科专病

我院专门开设了疑难病会诊中心，用来解决病情复杂、诊断困难的医学难题，为患者提供更深入、更专业的医学建议。包括但不限于胰腺疾病疑难病会诊、胸部疾病疑难病会诊、炎性肠病疑难病会诊、垂体疾病疑难病会诊。

联系我们：

地址：广州市番禺区大学城至善医院
E-mail：info@example.com
电话总机：020-12345678

• 患者预约记录：

- 按照预约时间，预约时段顺序对患者预约记录排序，不同状态下不同的操作按钮可用

医生界面

- 首页
- 患者预约记录**
- 药房
- 个人中心

患者预约记录

患者姓名	预约时间	预约时段	状态	操作
test1	2023年12月16日	8:00-10:00	就诊中	<button>接诊</button> <button>填写病例</button> <button>结束就诊</button>
test1	2023年12月16日	14:00-16:00	已挂号	<button>接诊</button> <button>填写病例</button> <button>结束就诊</button>
test1	2023年12月29日	8:00-10:00	已取消	<button>接诊</button> <button>填写病例</button> <button>结束就诊</button>
test1	2023年12月29日	8:00-10:00	已取消	<button>接诊</button> <button>填写病例</button> <button>结束就诊</button>
test1	2023年12月30日	10:00-12:00	就诊结束	<button>接诊</button> <button>填写病例</button> <button>结束就诊</button>

« 首页 上一页 第1/2页 下一页 尾页 »

- 点击“填写病历”后可以填写病历，点击“保存”可以保存或创建病历

填写病历

[返回上一页](#)

患者姓名	test1
症状	患者不分场合表现为经常困乏思睡，出现不同程度、不可抗拒的入睡。过多的睡眠引起显著痛苦或职业、社交等社会功能和生活质量的下降。有认知功能方面的改变，表现为近事记忆减退，思维能力下降，学习新事物能力下降。
诊断结果	嗜睡症
治疗方案	严格作息时间，白天有意识地让患者小睡，养成良好的生活习惯。 多运动 要多参加体育活动，每天不少于一小时，使自己的心身得到兴奋。 心理调节 要有积极的生活态度，每天给自己制定好生活学习计划，认真努力完成等。
<input type="button" value="保存"/>	

• 药房：

- 可以根据药物名（支持模糊搜索）或药物类型来筛选药物

医生界面

- 首页
- 患者预约记录
- 药房**
- 个人中心

药物名:	类型:	搜索
选择类型		
中药		
西药		
来氟米特片	西药	苏州长征-欣凯制药有限公司 79.00¥
对乙酰氨基酚口服溶液	西药	中美上海施贵宝制药有限公司 13.50¥
复方醋酸甲羟孕酮胶囊	西药	北京康必得药业有限公司 178.00¥
布洛芬混悬液	西药	上海强生制药有限公司 41.00¥
复方对乙酰氨基酚片	西药	拜耳医药保健有限公司 21.00¥
盐酸多奈哌齐片	西药	重庆植恩药业有限公司 25.00¥
复方氯唑沙宗胶囊	西药	万特制药(海南)有限公司 13.40¥
正骨水	中药	广西玉林制药集团有限责任公司 22.00¥
消痛贴膏	中药	西藏奇正藏药股份有限公司 15.00¥
连花清瘟胶囊	中药	石家庄以岭药业股份有限公司 18.00¥

« 首页 上一页 第 1 / 2 页 下一页 尾页 »

• 个人中心:

- 可以看到医生的个人信息，可进行账号注销操作

个人中心

姓名:	<input type="text" value="张三"/>
用户名:	<input type="text" value="zhangsan"/>
密码:	<input type="text" value="zhangsan123"/>
职称:	<input type="text" value="主任"/>
所属科室:	<input type="text" value="口腔"/>
<input type="button" value="返回"/> <input style="background-color: #ffcc00; border: 1px solid #ffcc00; color: black; font-weight: bold;" type="button" value="注销"/>	

7. 视频展示

展示1（基础功能展示）：

输入网址 - 注册 - 患者登录 - 首页 - 医生详情 - 查看医生信息 - 预约挂号 - 预约挂号搜索 - 点击预约 - 填写预约信息并提交 - 返回预约挂号界面（观察状态变化） - 查看挂号记录 - 取消预约 - 重新再挂一个号 -

查看个人信息 - 注销 - 医生登录 - 首页 - 接诊 - 填写病历 - 保存病历并返回 - 结束就诊 - 药房查询药物 -
查看个人信息 - 注销

| 视频见 video/display1.mp4

展示2 (错误机制和中间件机制) :

登录输入为空 - 登录输入用户名或密码错误 - 注册为空 - 注册用户名已使用 - 在未登录的情况下试图输入url到患者界面

| 视频见 video/display2.mp4