### 6.3 功能实现

#### 6.3.1 主要类的定义

主要类的定义已经在代码中体现，以下是其功能概述和扩展说明：

1. **User 类**：管理用户身份验证、角色分配和基础信息。
2. **Patient 类**：患者信息管理，用于与处方和账户关联。
3. **Doctor 类**：医生信息管理，涉及开具处方和查看患者病历。
4. **Prescription 类**：处方管理，负责记录患者的药品使用信息及其状态。
5. **Task 类**：任务管理，用于跟踪从处方到煎药完成的操作流程。

#### 6.3.2 系统的关键性代码

##### 用户注册与登录

* **注册模块**：  
  实现患者的注册功能，自动生成关联的患者和用户记录。

@app.route('/register', methods=['GET', 'POST'])  
def register():  
 if request.method == 'POST':  
 username = request.form['username']  
 password = request.form['password']  
 name = request.form['name']  
 contact\_number = request.form['contact\_number']  
  
 # 创建患者和用户记录  
 new\_patient = Patient(name=name, contact\_number=contact\_number)  
 db.session.add(new\_patient)  
 db.session.flush()  
 new\_user = User(username=username, password=password, role='patient', role\_id=new\_patient.patient\_id)  
 db.session.add(new\_user)  
 db.session.commit()  
 flash("注册成功！")  
 return redirect(url\_for('login'))  
 return render\_template('register.html')

* **登录模块**：  
  验证用户身份，依据角色跳转到对应的仪表盘。

@app.route('/login', methods=['GET', 'POST'])  
def login():  
 if request.method == 'POST':  
 username = request.form['username']  
 password = request.form['password']  
 user = User.query.filter\_by(username=username, password=password).first()  
 if user:  
 session['user\_id'] = user.uuid  
 session['role'] = user.role  
 return redirect(url\_for('dashboard'))  
 flash("用户名或密码错误！")  
 return render\_template('login.html')

### 6.4 安全性设计

#### 用户与权限划分

1. **用户角色**：
   * 患者：可查看个人信息和处方状态。
   * 医生：负责开具处方，查看和管理患者处方。
   * 工人：完成煎药相关任务。
   * 管理员：管理用户、任务和系统配置。
2. **权限控制**：
   * 基于角色的路由保护：
   * @app.route('/dashboard')  
     def dashboard():  
      if 'user\_id' not in session:  
      return redirect(url\_for('login'))  
      role = session['role']  
      if role == 'admin':  
      return redirect(url\_for('admin\_users'))  
      elif role == 'doctor':  
      return redirect(url\_for('doctor\_prescriptions'))  
      elif role == 'patient':  
      return redirect(url\_for('patient\_prescriptions'))  
      elif role == 'worker':  
      return redirect(url\_for('worker\_tasks'))  
      flash('角色未识别', 'danger')  
      return redirect(url\_for('login'))

#### 日志表设计

记录用户操作日志的触发器：

CREATE TABLE OperationLogs (  
 log\_id INT IDENTITY(1,1) PRIMARY KEY,  
 operation\_time DATETIME NOT NULL DEFAULT GETDATE(),  
 user\_id NVARCHAR(36),  
 operation NVARCHAR(200)  
);  
  
CREATE TRIGGER trg\_log\_operations  
ON prescriptions  
AFTER INSERT, UPDATE  
AS  
BEGIN  
 DECLARE @user NVARCHAR(36)  
 SET @user = SYSTEM\_USER  
 INSERT INTO OperationLogs (user\_id, operation)  
 VALUES (@user, '更新或插入处方');  
END;

### 6.5 完整性设计

#### 6.5.1 触发器设计

* **确保任务按阶段顺序更新**：
* CREATE TRIGGER trg\_task\_sequence  
  ON tasks  
  FOR UPDATE  
  AS  
  BEGIN  
   IF EXISTS (  
   SELECT \* FROM inserted i  
   WHERE i.form\_time IS NOT NULL AND i.receive\_time IS NULL  
   )  
   BEGIN  
   ROLLBACK TRANSACTION;  
   THROW 50001, '必须完成收方后才能进行配方', 1;  
   END  
  END;
* **自动计算处方完成时间**：
* CREATE TRIGGER trg\_prescription\_completion  
  ON prescriptions  
  AFTER UPDATE  
  AS  
  BEGIN  
   UPDATE prescriptions  
   SET expected\_pickup\_time = DATEADD(HOUR, 2, date)  
   WHERE prescription\_id IN (SELECT prescription\_id FROM inserted);  
  END;

#### 6.5.2 存储过程设计

* **添加新处方**：
* CREATE PROCEDURE AddPrescription  
   @patient\_id INT,  
   @doctor\_id INT,  
   @amount FLOAT,  
   @usage\_instructions NVARCHAR(MAX)  
  AS  
  BEGIN  
   INSERT INTO prescriptions (patient\_id, doctor\_id, amount, usage\_instructions, status)  
   VALUES (@patient\_id, @doctor\_id, @amount, @usage\_instructions, '待配方');  
  END;
* **分配任务**：
* CREATE PROCEDURE AssignTask  
   @task\_id INT,  
   @worker\_id INT,  
   @task\_type NVARCHAR(20)  
  AS  
  BEGIN  
   UPDATE tasks  
   SET   
   CASE   
   WHEN @task\_type = 'receive' THEN receive\_worker\_id = @worker\_id  
   WHEN @task\_type = 'formulate' THEN form\_worker\_id = @worker\_id  
   WHEN @task\_type = 'decoction' THEN decoction\_worker\_id = @worker\_id  
   END  
   WHERE task\_id = @task\_id;  
  END;

通过以上设计，系统在功能实现、安全性和数据完整性方面提供了全面支持，保障了中药代煎业务的高效稳定运行。