实验三报告

> 学号：<3225706084>
>
> 姓名：<陈芯悦>
>
> 指导老师：<张凯斌老师>
>
> 实验日期：<2025-03-30>

一、实验目的

- 复习软件工程的基本概念和方法论；
  - 软件生命周期与开发方法论；
    - 结构化分析与设计（SAD）
    - 面向对象分析与设计（OOAD）
- 掌握OOAD与UML图的对应关系；
  - 注意：UML图只是OOAD中的一部分（代码相关的部分），并不是OOAD的全部。例如：
    - 需求分析除了UML图外，还有文档说明；
    - 总体设计除了UML图外，还有UI设计、数据库设计等；
    - 详细设计除了UML图外，还有算法实现（流程图、N-S图、伪代码）、UI的具体实现、数据库的具体实现等；
- 完成教科书中关系数据库实例的UML建模练习；

二、实验内容

- 阅读教科书的第9章“数据管理”的第9.4节“关系数据库的开发”；
- 根据理论课所讲内容和软件工程的相关概念，完成教科书上关系数据库实例的UML建模练习；

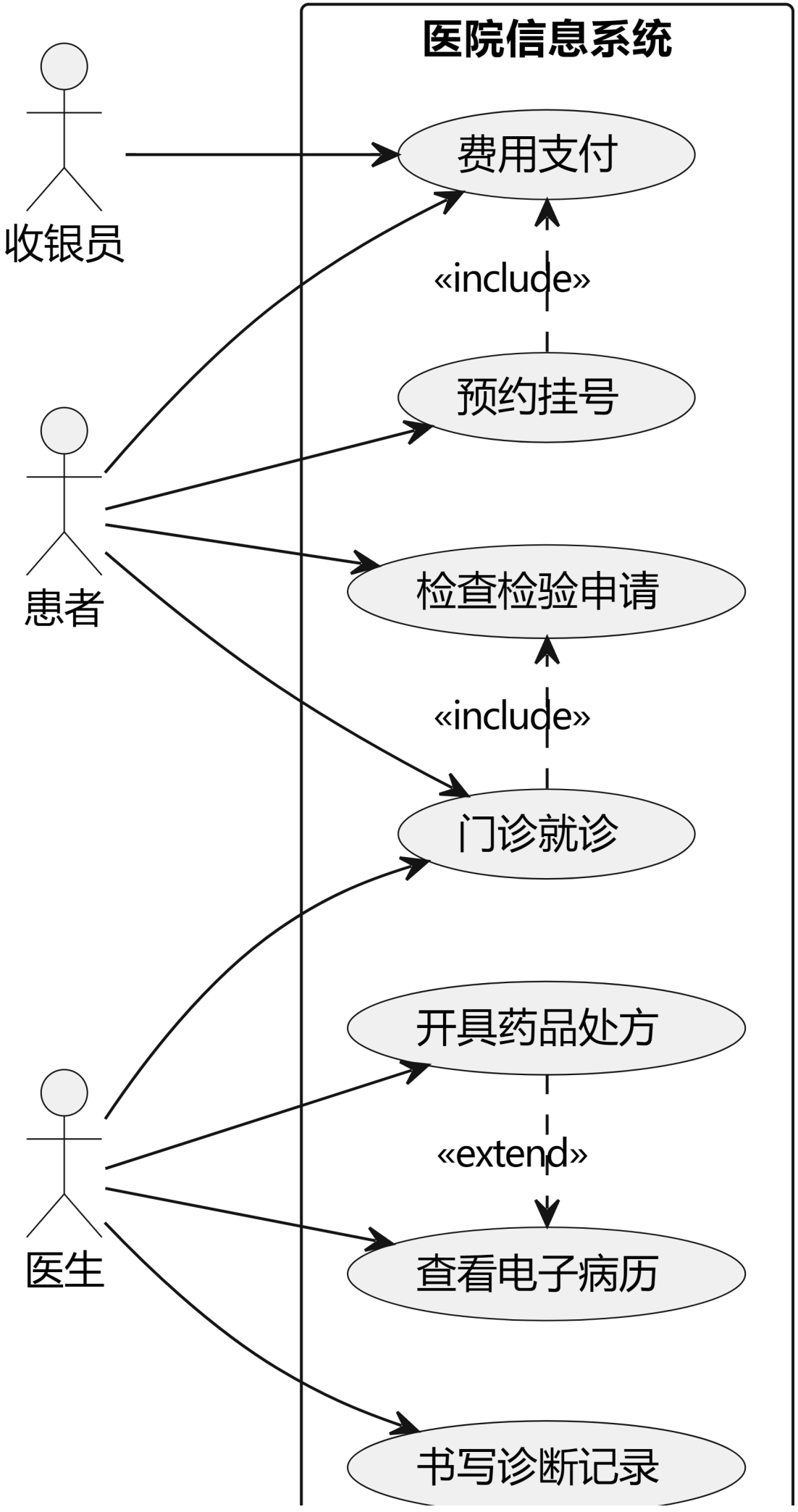
三、实验要求

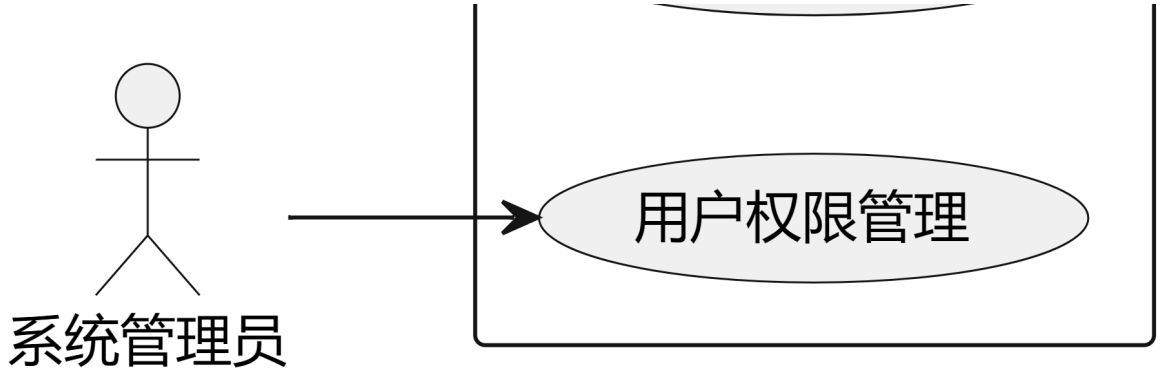
- 需求分析：完成用例图和用例规约；
- 总体设计：完成类图（静态视图）和活动图（动态视图）；
- 详细设计：完成详细类图和包图；
- 撰写并提交实验报告；
  - 实验步骤部分需严格按模板要求撰写；
  - 提交时删除掉上文中的“UML提纲.png“图片；

四、实验步骤

1. 需求分析

1.1 用例图


医院信息系统
收银员
患者
医生
费用支付
«include»
预约挂号
检查检验申请
«include»
门诊就诊
开具药品处方
«extend»
查看电子病历
书写诊断记录



1.2 用例规约

用例1：预约挂号

字段	内容
用例名	患者预约挂号服务
执行者	患者（主）、收银员（辅助）
前置条件	1. 系统在线且可访问 2. 患者账号已注册并验证
触发事件	患者通过线上/线下渠道发起挂号请求
主成功场景	1. 患者选择科室/医生/时间 2. 系统验证号源有效性 3. 生成挂号单（含挂号ID） 4. 推送挂号凭证至患者端
扩展场景	号源不足时： - 推荐其他时段/医生 - 加入候补队列
最小保证	挂号操作记录（含失败尝试）必须写入审计日志
后置条件	患者账户关联有效挂号记录
优先级	高
频度	每日数千次
输入	科室代码、医生工号、就诊时间
输出	电子挂号单（含二维码）、短信提醒
异常	1. 支付超时 2. 医生临时停诊 3. 系统并发错误
附加信息	支持医保卡/电子健康卡绑定

用例2：门诊就诊

字段	内容
用例名	医生接诊患者
执行者	医生（主）、患者（配合）
前置条件	1. 患者已完成签到 2. 医生工作站登录成功
触发事件	叫号系统分配患者至诊室
主成功场景	1. 调取患者电子病历 2. 进行临床问诊检查 3. 生成初步诊断记录
扩展场景	需要检查时： - 开具检查申请单 - 同步推送检验科
最小保证	问诊过程全程录音录像（加密存储）
后置条件	电子病历新增就诊记录
优先级	最高
频度	每位医生每日50-100次
输入	患者主诉、体征数据
输出	诊断记录（XML格式）、检查申请单
异常	1. 病历调取失败 2. 系统卡顿影响问诊
附加信息	集成合理用药知识库

用例3：检查检验申请

字段	内容
用例名	执行医学检验检查
执行者	检验科医师（主）、患者（配合）
前置条件	1. 医生已开具申请单 2. 患者完成缴费
触发事件	检验科接收电子申请单
主成功场景	1. 扫码核对患者身份 2. 执行检验项目（如血常规） 3. 上传结构化报告

字段	内容
扩展场景	异常结果时： - 触发危急值预警 - 自动通知主治医生
最小保证	原始检测数据保留15年
后置条件	检验报告关联电子病历
优先级	中
频度	每日全院约2000次
输入	生物样本、检验项目代码
输出	LIS检验报告（PDF+HL7格式）
异常	1. 样本污染 2. 设备故障 3. 结果传输失败
附加信息	支持ISO15189质控标准

用例4：开具药品处方

字段	内容
用例名	医生开具电子处方
执行者	医生（主）、药师（审核）
前置条件	1. 诊断记录已保存 2. 患者无药物过敏记录
触发事件	医生提交处方请求
主成功场景	1. 选择药品目录 2. 系统验证配伍禁忌 3. 生成数字签名处方 4. 推送至药房系统
扩展场景	毒麻药品需： - 二级医师审核 - 公安系统备案
最小保证	处方修改留痕（区块链存证）
后置条件	处方进入药房处理队列
优先级	高
频度	每日约5000次
输入	药品编码、剂量、用法

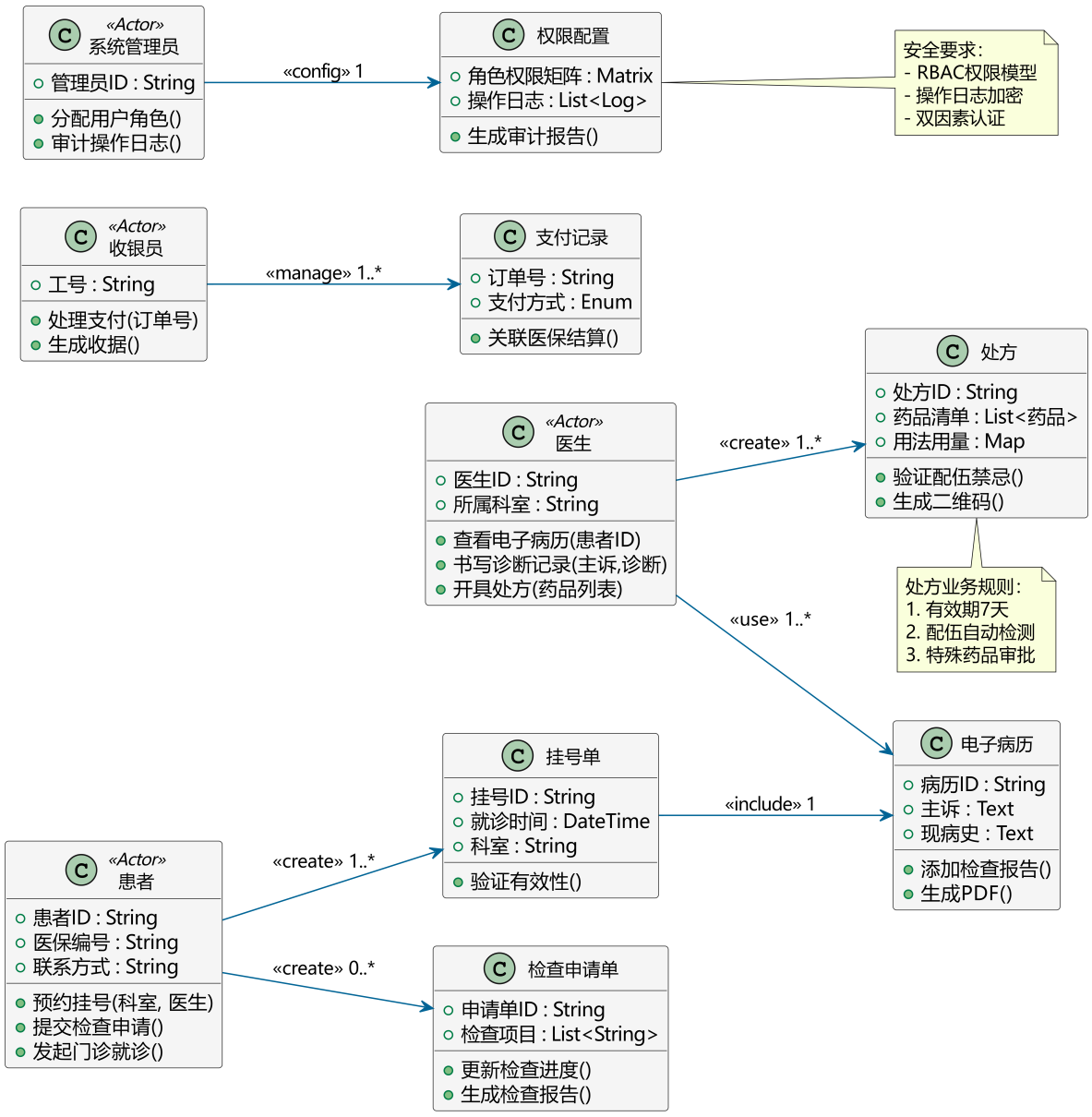
字段	内容
输出	电子处方（含CA签名）、用药指导单
异常	1. 库存不足 2. 医保目录限制
附加信息	对接国家药品监管码系统

用例5：用户权限管理

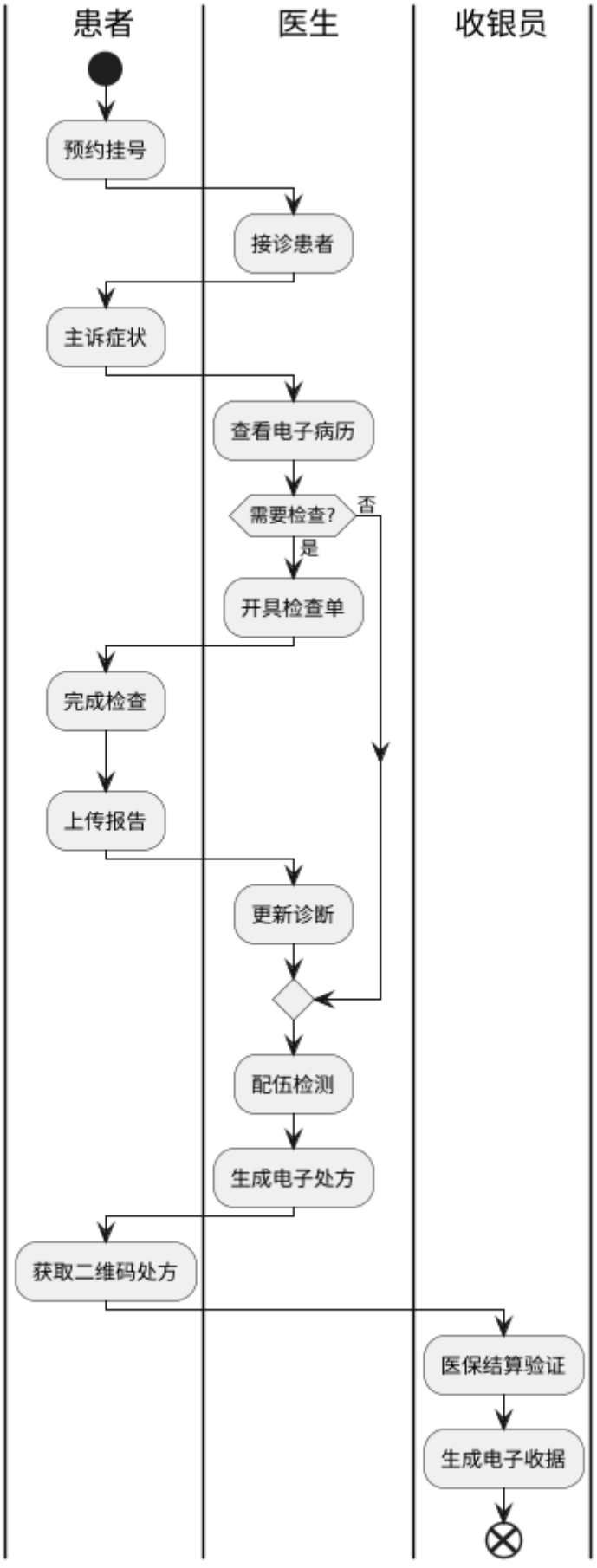
字段	内容
用例名	管理系统访问权限
执行者	系统管理员
前置条件	管理员通过双因素认证
触发事件	新员工入职/角色变更
主成功场景	1. 选择用户/角色 2. 分配权限组 3. 生成审计日志
扩展场景	权限冲突时： - 触发风险预警 - 要求上级审批
最小保证	权限变更记录不可篡改
后置条件	用户获得新权限配置
优先级	中
频度	每日约20次
输入	员工工号、权限代码
输出	权限分配确认书、操作日志
异常	1. 越权操作 2. 权限继承冲突
附加信息	符合等保2.0三级要求

2. 总体设计

2.1 类图（静态视图）

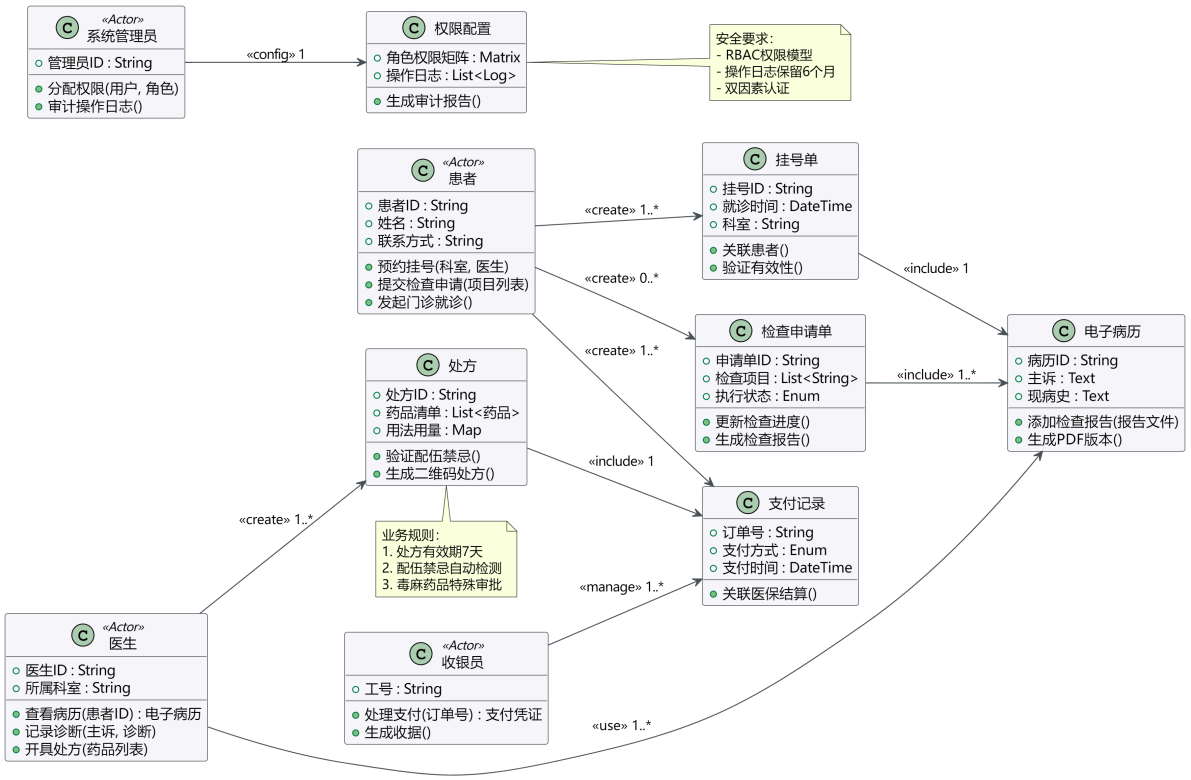

«Actor»
系统管理员
管理员ID : String
分配用户角色()
审计操作日志()
«config» 1
权限配置
角色权限矩阵 : Matrix
操作日志 : List<Log>
生成审计报告()
安全要求:
- RBAC权限模型
- 操作日志加密
- 双因素认证
«Actor»
收银员
工号 : String
处理支付(订单号)
生成收据()
«manage» 1..*
支付记录
订单号 : String
支付方式 : Enum
关联医保结算()
处方
处方ID : String
药品清单 : List<药品>
用法用量 : Map
验证配伍禁忌()
生成二维码()
«Actor»
医生
医生ID : String
所属科室 : String
查看电子病历(患者ID)
书写诊断记录(主诉,诊断)
开具处方(药品列表)
«create» 1..*
«use» 1..*
处方业务规则:
1. 有效期7天
2. 配伍自动检测
3. 特殊药品审批
电子病历
病历ID : String
主诉 : Text
现病史 : Text
添加检查报告()
生成PDF()
挂号单
挂号ID : String
就诊时间 : DateTime
科室 : String
验证有效性()
«include» 1
«Actor»
患者
患者ID : String
医保编号 : String
联系方式 : String
预约挂号(科室, 医生)
提交检查申请()
发起门诊就诊()
«create» 1..*
«create» 0..*
检查申请单
申请单ID : String
检查项目 : List<String>
更新检查进度()
生成检查报告()

2.2 活动图（动态视图）

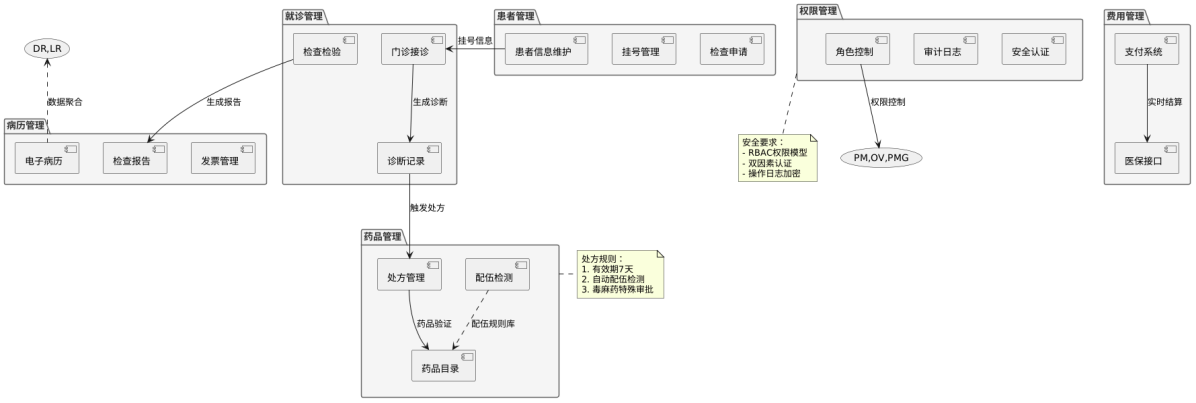

患者
医生
收银员
预约挂号
接诊患者
主诉症状
查看电子病历
需要检查?
否
是
开具检查单
完成检查
上传报告
更新诊断
配伍检测
生成电子处方
获取二维码处方
医保结算验证
生成电子收据

3. 详细设计

3.1 详细类图



3.2 包图



4. 编码

4.1 代码实现

```
//预约门诊界面
import { BookingInfo } from '../models/BookingModel';
import { bookingStore } from '../services/BookingStore';
import router from '@ohos.router';
import promptAction from '@ohos.promptAction';

interface RouterError {
  code: number;
  message?: string;
}

@Entry
```

```
@Component
struct PatientBookingPage {
  @State name: string = '';
  @State gender: string = '男';
  @State age: number = 0;
  @State symptoms: string = '';
  @State bookingId: string = '';

  build() {
    Column() {
      // 在 Row() 组件中添加病例管理入口按钮
      Row() {
        Text('医院挂号')
          .fontSize(24)
          .margin({ top: 20 })

        Button('医生入口')
          .margin({ top: 20, left: 20 })
          .backgroundColor('#4CAF50')
          .onClick(() => {
            router.pushUrl({
              url: 'pages/DoctorPage'
            })
              .then(() => {
                console.info('跳转到医生页面成功');
              })
              .catch((err: object) => {
                const routerError = err as RouterError;
                console.error(`跳转失败: ${routerError.code},
${routerError.message || '未知错误'}`);
                promptAction.showToast({
                  message: '跳转失败，请重试'
                });
              });
          })

        Button('病例管理')
          .margin({ top: 20, left: 20 })
          .backgroundColor('#2196F3')
          .onClick(() => {
            router.pushUrl({
              url: 'pages/MedicalRecordPage'
            })
              .then(() => {
                console.info('跳转到病例管理页面成功');
              })
              .catch((err: object) => {
                const routerError = err as RouterError;
                console.error(`跳转失败: ${routerError.code},
${routerError.message || '未知错误'}`);
                promptAction.showToast({
                  message: '跳转失败，请重试'
                });
              });
          })
      }
```

```
      .width('90%')
      .justifyContent(FlexAlign.SpaceBetween)

      // 添加查询预约状态区域
      Column() {
        Text('查询预约状态')
          .fontSize(16)
          .margin({ top: 20, bottom: 10 })
          .fontColor('#666666')

        Row() {
          TextInput({ placeholder: '请输入预约号' })
            .width('70%')
            .height(40)
            .margin({ right: 10 })
            .onChange((value: string) => {
              this.bookingId = value;
            })

          Button('查询')
            .width('20%')
            .height(40)
            .backgroundColor('#2196F3')
            .onClick(() => {
              this.checkBooking();
            })
        }
        .width('90%')
        .margin({ bottom: 20 })
      }
      .width('100%')
      .backgroundColor('#F5F5F5')
      .padding({ top: 10, bottom: 10, left: 10, right: 10 })  // 修改这里

      Divider()
        .width('90%')
        .color('#EEEEEE')
        .margin({ top: 10, bottom: 10 })  // 修改这里

      Text('新建预约')
        .fontSize(16)
        .margin({ top: 10, bottom: 10 })  // 修改这里
        .fontColor('#666666')

      TextInput({ placeholder: '请输入姓名' })
        .width('90%')
        .margin(10)
        .onChange((value: string) => {
          this.name = value;
        })

      Select([{ value: '男' }, { value: '女' }])
        .width('90%')
        .margin(10)
        .selected(0)
        .onSelect((_: number, value: string) => {
```

```
          this.gender = value;
        })

      TextInput({ placeholder: '请输入年龄' })
        .width('90%')
        .margin(10)
        .type(InputType.Number)
        .onChange((value: string) => {
          this.age = parseInt(value) || 0;
        })

      TextArea({ placeholder: '请描述病症（50字以内）' })
        .width('90%')
        .height(100)
        .margin(10)
        .maxLength(50)
        .onChange((value: string) => {
          this.symptoms = value;
        })

      Button('提交预约')
        .width('90%')
        .margin({ top: 20 })
        .backgroundColor('#4CAF50')
        .onClick(() => {
          if (this.validateForm()) {
            this.submitBooking();
          }
        })
    }
    .width('100%')
    .height('100%')
    .padding({ left: 16, right: 16, top: 16, bottom: 16 })  // 修改这里
    .backgroundColor('#FFFFFF')
  }

  private checkBooking(): void {
    if (!this.bookingId.trim()) {
      promptAction.showToast({
        message: '请输入预约号'
      });
      return;
    }

    const booking = bookingStore.getBooking(this.bookingId);
    if (!booking) {
      promptAction.showToast({
        message: '未找到该预约记录'
      });
      return;
    }

    router.pushUrl({
      url: 'pages/WaitingPage',
      params: { bookingId: this.bookingId }
    })
```

```
      .then(() => {
        console.info('跳转到等待页面成功');
      })
      .catch((err: object) => {
        const routerError = err as RouterError;
        console.error(`跳转失败: ${routerError.code}, ${routerError.message || '未
知错误'}`);
        promptAction.showToast({
          message: '跳转失败，请重试'
        });
      });
  }

  private validateForm(): boolean {
    if (!this.name.trim()) {
      promptAction.showToast({ message: '请输入姓名' });
      return false;
    }
    if (this.age <= 0 || this.age > 150) {
      promptAction.showToast({ message: '请输入有效年龄' });
      return false;
    }
    if (!this.symptoms.trim()) {
      promptAction.showToast({ message: '请描述病症' });
      return false;
    }
    return true;
  }

  private submitBooking(): void {
    const booking = new BookingInfo(
      Date.now().toString(),
      this.name,
      this.gender,
      this.age,
      this.symptoms
    );

    bookingStore.addBooking(booking);
    router.pushUrl({
      url: 'pages/WaitingPage',
      params: { bookingId: booking.id }
    })
      .then(() => {
        console.info('跳转到等待页面成功');
      })
      .catch((err: object) => {
        const routerError = err as RouterError;
        console.error(`跳转失败: ${routerError.code}, ${routerError.message || '未
知错误'}`);
        promptAction.showToast({
          message: '跳转失败，请重试'
        });
      });
  }
}
```

```
//医生门诊界面
import { BookingInfo, BookingStatus, Medicine, Prescription } from
'../models/BookingModel';
import { bookingStore } from '../services/BookingStore';
import router from '@ohos.router';
import promptAction from '@ohos.promptAction';
import common from '@ohos.app.ability.common';

@Entry
@Component
struct DoctorPage {
  @State bookingList: Array<BookingInfo> = [];
  @State selectedBooking: BookingInfo | null = null;
  @State doctorAdvice: string = '';
  @State totalFee: string = '';
  @State medicines: Array<Medicine> = [];
  @State showAddMedicine: boolean = false;
  @State newMedicine: Medicine = new Medicine('', '', '', 0);
  dialogController: CustomDialogController = new CustomDialogController({
    builder: this.AddMedicineDialog.bind(this),
    autoCancel: true,
    alignment: DialogAlignment.Center
  });

  aboutToAppear() {
    this.refreshBookings();
  }

  refreshBookings() {
    this.bookingList = bookingStore.getAllBookings();
  }

  @Builder
  AddMedicineDialog() {
    Column() {
      Text('添加药品')
        .fontSize(20)
        .margin({ bottom: 20 })

      TextInput({ placeholder: '药品名称' })
        .width('90%')
        .margin({ bottom: 10 })
        .onChange((value: string) => {
          this.newMedicine.name = value;
        })

      TextInput({ placeholder: '用药剂量' })
        .width('90%')
        .margin({ bottom: 10 })
        .onChange((value: string) => {
          this.newMedicine.dosage = value;
        })

      TextInput({ placeholder: '用药频率' })
```

```
        .width('90%')
        .margin({ bottom: 10 })
        .onChange((value: string) => {
          this.newMedicine.frequency = value;
        })

      TextInput({ placeholder: '药品价格' })
        .width('90%')
        .margin({ bottom: 20 })
        .type(InputType.Number)
        .onChange((value: string) => {
          this.newMedicine.price = parseFloat(value) || 0;
        })

      Row() {
        Button('取消')
          .onClick(() => {
            this.dialogController.close();
          })
        Button('确定')
          .onClick(() => {
            if (this.validateMedicine()) {
              this.medicines.push(new Medicine(
                this.newMedicine.name,
                this.newMedicine.dosage,
                this.newMedicine.frequency,
                this.newMedicine.price
              ));
              this.newMedicine = new Medicine('', '', '', 0);
              this.dialogController.close();
            }
          })
      }
      .justifyContent(FlexAlign.SpaceAround)
      .width('100%')
    }
    .padding(20)
  }

  build() {
    Column() {
      Column() {
        Text('医生工作台')
          .fontSize(24)
          .fontWeight(FontWeight.Bold)
          .margin({ top: 20, bottom: 20 })

        List({ space: 10, initialIndex: 0 }) {
          ForEach(this.bookingList, (booking: BookingInfo) => {
            ListItem() {
              Column() {
                Row() {
                  Text(`患者：${booking.patientName}`)
                    .fontSize(16)
                  Text(`状态：${booking.status === BookingStatus.WAITING ? '等待中'
: '待缴费'}`)
```

```
                    .fontSize(16)
                    .margin({ left: 20 })
                }
                .width('100%')
                .padding(10)
                .backgroundColor(this.selectedBooking?.id === booking.id ?
'#E8F0FE' : '#FFFFFF')
              }
              .onClick(() => {
                this.selectedBooking = booking;
                this.doctorAdvice = '';
                this.totalFee = '';
                this.medicines = [];
              })
            }
          })
        }
        .width('90%')
        .height(200)
        .border({ width: 1, color: '#CCCCCC' })

        if (this.selectedBooking) {
          this.buildPatientInfo()
        }

        Button('返回首页')
          .width('80%')
          .margin({ top: 20 })
          .backgroundColor('#8F8F8F')
          .onClick(() => {
            router.back();
          })
      }
      .width('100%')
      .height('100%')
      .padding(20)
    }
  }

  @Builder
  private buildPatientInfo() {
    Column() {
      Text('患者信息：')
        .fontSize(18)
        .fontWeight(FontWeight.Bold)
        .margin({ top: 20, bottom: 10 })

      if (this.selectedBooking) {
        Text(`姓名：${this.selectedBooking.patientName}`)
        Text(`性别：${this.selectedBooking.gender}`)
        Text(`年龄：${this.selectedBooking.age}`)
        Text(`病症：${this.selectedBooking.symptoms}`)
          .margin({ bottom: 20 })

        if (this.selectedBooking.status === BookingStatus.WAITING) {
          Button('添加药品')
```

```
            .width('80%')
            .onClick(() => {
              this.dialogController.open();
            })

          List({ space: 10, initialIndex: 0 }) {
            ForEach(this.medicines, (medicine: Medicine, index: number) => {
              ListItem() {
                Row() {
                  Text(`${medicine.name} - ${medicine.dosage} -
${medicine.frequency}`)
                  Text(`￥${medicine.price}`)
                    .margin({ left: 20 })
                  Button('删除')
                    .fontSize(14)
                    .margin({ left: 20 })
                    .onClick(() => {
                      this.medicines.splice(index, 1);
                    })
                }
                .width('100%')
                .padding(10)
              }
            })
          }
          .width('90%')
          .height(100)

          TextArea({ placeholder: '请输入医嘱' })
            .width('80%')
            .height(100)
            .margin({ top: 20 })
            .onChange((value: string) => {
              this.doctorAdvice = value;
            })

          TextInput({ placeholder: '请输入总费用' })
            .width('80%')
            .type(InputType.Number)
            .margin({ top: 20 })
            .onChange((value: string) => {
              this.totalFee = value;
            })

          Button('确认开具处方')
            .width('80%')
            .margin({ top: 20 })
            .onClick(() => {
              this.handlePrescription();
            })
        }
      }
    }
    .width('90%')
  }
```

```
  private validateMedicine(): boolean {
    if (!this.newMedicine.name.trim()) {
      promptAction.showToast({ message: '请输入药品名称' });
      return false;
    }
    if (!this.newMedicine.dosage.trim()) {
      promptAction.showToast({ message: '请输入用药剂量' });
      return false;
    }
    if (!this.newMedicine.frequency.trim()) {
      promptAction.showToast({ message: '请输入用药频率' });
      return false;
    }
    if (this.newMedicine.price <= 0) {
      promptAction.showToast({ message: '请输入有效的药品价格' });
      return false;
    }
    return true;
  }

  private validatePrescription(): boolean {
    if (this.medicines.length === 0) {
      promptAction.showToast({ message: '请至少添加一种药品' });
      return false;
    }
    if (!this.doctorAdvice.trim()) {
      promptAction.showToast({ message: '请输入医嘱' });
      return false;
    }
    if (!this.totalFee || parseFloat(this.totalFee) <= 0) {
      promptAction.showToast({ message: '请输入有效的总费用' });
      return false;
    }
    return true;
  }

  private handlePrescription() {
    if (!this.validatePrescription() || !this.selectedBooking) {
      return;
    }

    try {
      const prescription = new Prescription(
        this.medicines,
        this.doctorAdvice,
        parseFloat(this.totalFee)
      );
      this.selectedBooking.prescription = prescription;
      this.selectedBooking.status = BookingStatus.PENDING_PAYMENT;
      bookingStore.updateBooking(this.selectedBooking);
      this.refreshBookings();
      promptAction.showToast({ message: '处方已开具' });
    } catch (error) {
      promptAction.showToast({ message: '开具处方失败，请重试' });
    }
  }
```

```
}
```

```
//病例管理系统
import { MedicalRecord } from '../models/MedicalRecordModel';
import { medicalRecordStore } from '../services/MedicalRecordStore';
import router from '@ohos.router';
import promptAction from '@ohos.promptAction';

interface SelectOption {
  text: string;
  value: string;
}
interface ValidationRule {
  condition: boolean;
  message: string;
}

@Entry
@Component
struct MedicalRecordPage {
  @State searchKeyword: string = '';
  @State records: Array<MedicalRecord> = [];
  @State isEditing: boolean = false;
  @State currentRecord: MedicalRecord | null = null;
  @State currentUser: string = '陈医生';
  @State currentTime: string = '2025-04-07 19:23:22';

  aboutToAppear() {
    this.loadRecords();
  }

  loadRecords() {
    this.records = this.searchKeyword
      ? medicalRecordStore.searchRecords(this.searchKeyword)
      : medicalRecordStore.getAllRecords();
  }

  private getGenderOptions(): Array<SelectOption> {
    return [
      { text: '男', value: '男' },
      { text: '女', value: '女' }
    ];
  }

  private copyRecord(record: MedicalRecord): MedicalRecord {
    return {
      id: record.id,
      patientName: record.patientName,
      age: record.age,
      gender: record.gender,
      symptoms: record.symptoms,
      doctorAdvice: record.doctorAdvice,
      totalAmount: record.totalAmount,
      visitDate: record.visitDate,
      doctorId: record.doctorId,
```

```
      doctorName: record.doctorName,
      isCompleted: record.isCompleted,
      paymentStatus: record.paymentStatus  // 添加支付状态
    };
  }

  private createNewRecord(): MedicalRecord {
    return {
      id: '',
      patientName: '',
      age: 0,
      gender: '男',
      symptoms: '',
      doctorAdvice: '',
      totalAmount: 0,
      visitDate: this.currentTime,
      doctorId: this.currentUser,
      doctorName: this.currentUser,
      isCompleted: false,
      paymentStatus: 'unpaid'  // 添加支付状态，新建时默认为未支付
    };
  }

  @Builder
  RecordItem(record: MedicalRecord) {
    Column() {
      Row() {
        Text(record.patientName)
          .fontSize(18)
          .fontWeight(FontWeight.Bold)
        Text(`${record.age}岁`)
          .fontSize(16)
          .margin({ left: 10 })
        Text(record.gender)
          .fontSize(16)
          .margin({ left: 10 })
      }.width('100%')

      Text(`症状：${record.symptoms}`)
        .fontSize(14)
        .margin({ top: 5 })
        .width('100%')

      Text(`医嘱：${record.doctorAdvice}`)
        .fontSize(14)
        .margin({ top: 5 })
        .width('100%')

      Row() {
        Text(`就诊金额：¥${record.totalAmount.toFixed(2)}`)
          .fontSize(14)
        Blank()
        Button('编辑')
          .width(80)
          .height(30)
          .backgroundColor('#2196F3')
```

```
          .margin({ right: 10 })
          .onClick(() => {
            this.currentRecord = this.copyRecord(record);
            this.isEditing = true;
          })
        Button('删除')
          .width(80)
          .height(30)
          .backgroundColor('#F44336')
          .onClick(() => {
            this.showDeleteConfirm(record.id);
          })
      }
      .width('100%')
      .margin({ top: 10 })
    }
    .width('100%')
    .padding(10)
    .backgroundColor('#FFFFFF')
    .borderRadius(8)
  }

  private showDeleteConfirm(id: string) {
    promptAction.showDialog({
      title: '确认删除',
      message: '是否确认删除该病例？',
      buttons: [
        { text: '取消', color: '#666666' },
        { text: '确认', color: '#FF0000' }
      ]
    }).then((result) => {
      if (result.index === 1 && medicalRecordStore.deleteRecord(id)) {
        promptAction.showToast({ message: '删除成功' });
        this.loadRecords();
      }
    });
  }

  private validateRecord(): boolean {
    if (!this.currentRecord) {
      promptAction.showToast({ message: '病例数据无效' });
      return false;
    }

    // 分别验证每个字段
    if (!this.currentRecord.patientName?.trim()) {
      promptAction.showToast({ message: '请输入患者姓名' });
      return false;
    }

    if (!this.currentRecord.age || this.currentRecord.age <= 0) {
      promptAction.showToast({ message: '请输入有效年龄' });
      return false;
    }

    if (!this.currentRecord.symptoms?.trim()) {
```

```
      promptAction.showToast({ message: '请输入症状描述' });
      return false;
    }

    if (!this.currentRecord.doctorAdvice?.trim()) {
      promptAction.showToast({ message: '请输入医嘱' });
      return false;
    }

    if (!this.currentRecord.totalAmount || this.currentRecord.totalAmount <= 0) {
      promptAction.showToast({ message: '请输入有效金额' });
      return false;
    }

    return true;
  }

  private saveRecord(): boolean {
    if (!this.validateRecord() || !this.currentRecord) return false;

    if (this.currentRecord.id) {
      if (medicalRecordStore.updateRecord(this.currentRecord)) {
        promptAction.showToast({ message: '更新成功' });
      } else {
        promptAction.showToast({ message: '更新失败' });
        return false;
      }
    } else {
      this.currentRecord.id = Date.now().toString();
      this.currentRecord.visitDate = this.currentTime;
      medicalRecordStore.addRecord(this.currentRecord);
      promptAction.showToast({ message: '添加成功' });
    }

    this.isEditing = false;
    this.currentRecord = null;
    this.loadRecords();
    return true;
  }

  @Builder
  EditDialog() {
    Stack() {
      // 遮罩层
      Column()
        .width('100%')
        .height('100%')
        .backgroundColor('#000000')
        .opacity(0.5)
        .onClick(() => {
          this.isEditing = false;
          this.currentRecord = null;
        })


      // 弹窗内容
```

```
      Column() {
        Text(this.currentRecord ? '编辑病例' : '新增病例')
          .fontSize(20)
          .fontWeight(FontWeight.Bold)
          .margin({ bottom: 20 })


        TextInput({ placeholder: '患者姓名', text: this.currentRecord?.patientName
|| '' })
          .width('100%')
          .height(40)
          .margin({ bottom: 10 })
          .onChange((value: string) => {
            if (this.currentRecord) {
              this.currentRecord.patientName = value;
            }
          })

        TextInput({ placeholder: '年龄', text: this.currentRecord?.age.toString()
|| '' })
          .width('100%')
          .height(40)
          .margin({ bottom: 10 })
          .type(InputType.Number)
          .onChange((value: string) => {
            if (this.currentRecord) {
              this.currentRecord.age = parseInt(value) || 0;
            }
          })

        Select(this.getGenderOptions())
          .width('100%')
          .margin({ bottom: 10 })
          .selected(this.currentRecord?.gender === '女' ? 1 : 0)
          .onSelect((index: number, value: string) => {
            if (this.currentRecord) {
              this.currentRecord.gender = value;
            }
          })

        TextArea({ placeholder: '症状描述', text: this.currentRecord?.symptoms ||
'' })
          .width('100%')
          .height(80)
          .margin({ bottom: 10 })
          .onChange((value: string) => {
            if (this.currentRecord) {
              this.currentRecord.symptoms = value;
            }
          })

        TextArea({ placeholder: '医嘱', text: this.currentRecord?.doctorAdvice ||
'' })
          .width('100%')
          .height(80)
          .margin({ bottom: 10 })
```

```
          .onChange((value: string) => {
            if (this.currentRecord) {
              this.currentRecord.doctorAdvice = value;
            }
          })

        TextInput({ placeholder: '就诊金额', text:
this.currentRecord?.totalAmount.toString() || '' })
          .width('100%')
          .height(40)
          .margin({ bottom: 20 })
          .type(InputType.Number)
          .onChange((value: string) => {
            if (this.currentRecord) {
              this.currentRecord.totalAmount = parseFloat(value) || 0;
            }
          })

        Row() {
          Button('取消')
            .width(120)
            .height(40)
            .backgroundColor('#CCCCCC')
            .margin({ right: 20 })
            .onClick(() => {
              this.isEditing = false;
              this.currentRecord = null;
            })

          Button('保存')
            .width(120)
            .height(40)
            .backgroundColor('#4CAF50')
            .onClick(() => {
              if (this.saveRecord()) {
                this.isEditing = false;
              }
            })
        }
        .justifyContent(FlexAlign.Center)
        .width('100%')
      }
      .width('90%')
      .padding(20)
      .backgroundColor('#FFFFFF')
      .borderRadius(16)
    }
    .width('100%')
    .height('100%')
    .position({ x: '5%', y: 0 })
    .zIndex(999)
  }

  build() {
    Stack() {  // 将最外层的 Column 改为 Stack
      Column() {
```

```
      // 顶部栏
      Row() {
        Column() {
          Text('病例管理系统')
            .fontSize(24)
            .fontWeight(FontWeight.Bold)
          Text(`当前用户：${this.currentUser}`)
            .fontSize(14)
            .margin({ top: 4 })
        }
        Blank()
        Button('返回')
          .backgroundColor('#CCCCCC')
          .onClick(() => router.back())
      }
      .width('100%')
      .padding(16)
      .backgroundColor('#F5F5F5')

      // 搜索和新增
      Row() {
        TextInput({
          placeholder: '搜索病例...',
          text: this.searchKeyword
        })
          .width('55%')  // 修改宽度
          .height(40)
          .margin({ right: 8 })
          .onChange((value: string) => {
            this.searchKeyword = value;
            this.loadRecords();
          })

        Button('搜索全部')  // 新增搜索全部按钮
          .width('20%')
          .height(40)
          .backgroundColor('#2196F3')
          .margin({ right: 8 })
          .onClick(() => {
            this.searchKeyword = '';
            this.loadRecords();
          })

        Button('新增')
          .width('20%')
          .height(40)
          .backgroundColor('#4CAF50')
          .onClick(() => {
            this.currentRecord = this.createNewRecord();
            this.isEditing = true;
          })
      }
      .width('90%')
      .margin({ top: 16, bottom: 16 })
      // 病例列表
      List() {
```

```
          ForEach(this.records, (record: MedicalRecord) => {
            ListItem() {
              this.RecordItem(record)
            }.margin({ bottom: 10 })
          })
        }
        .width('90%')
        .height('70%')
      }
      .width('100%')
      .height('100%')
      .backgroundColor('#F0F0F0')

      // 编辑对话框
      if (this.isEditing) {
        this.EditDialog()
      }
    }
  }
}
```

```
//查询/等待门诊结果
import { BookingInfo, BookingStatus } from '../models/BookingModel';
import { bookingStore } from '../services/BookingStore';
import router from '@ohos.router';
import promptAction from '@ohos.promptAction';

@CustomDialog
struct ModifySymptomDialog {
  @State newSymptoms: string = '';
  controller?: CustomDialogController = undefined;
  currentBooking?: BookingInfo = undefined;
  onSymptomConfirm: (value: string) => void = () => {};

  build() {
    Column() {
      Text('修改病症')
        .fontSize(20)
        .fontWeight(FontWeight.Bold)
        .margin({ bottom: 20 })

      TextArea({
        placeholder: '请输入新的病症描述（50字以内）',
        text: this.currentBooking?.symptoms || ''
      })
        .width('90%')
        .height(100)
        .maxLength(50)
        .onChange((value: string) => {
          this.newSymptoms = value;
        })

      Row() {
        Button('取消')
          .onClick(() => {
```

```
            if (this.controller) {
              this.controller.close();
            }
          })
        Button('确定')
          .onClick(() => {
            if (this.newSymptoms.trim()) {
              this.onSymptomConfirm(this.newSymptoms);
              if (this.controller) {
                this.controller.close();
              }
            } else {
              promptAction.showToast({ message: '请输入病症描述' });
            }
          })
      }
      .width('90%')
      .justifyContent(FlexAlign.SpaceAround)
      .margin({ top: 20 })
    }
    .padding(20)
  }
}

@Entry
@Component
struct WaitingPage {
  @State booking: BookingInfo | null = null;
  @State newSymptoms: string = '';
  private bookingId: string = '';
  dialogController: CustomDialogController = new CustomDialogController({
    builder: ModifySymptomDialog({
      controller: undefined,
      currentBooking: undefined,
      onSymptomConfirm: (symptoms: string): void => {
        this.handleModifySymptoms(symptoms);
      }
    })
  });

  aboutToAppear(): void {
    const params = router.getParams() as Record<string, string>;
    if (params && typeof params.bookingId === 'string') {
      this.bookingId = params.bookingId;
      this.refreshBooking();
    }
  }

  refreshBooking(): void {
    if (this.bookingId) {
      const foundBooking = bookingStore.getBooking(this.bookingId);
      if (foundBooking) {
        this.booking = foundBooking;
        this.newSymptoms = foundBooking.symptoms;

        // 更新对话框控制器
```

```
        this.dialogController = new CustomDialogController({
          builder: ModifySymptomDialog({
            controller: undefined,
            currentBooking: foundBooking,
            onSymptomConfirm: (symptoms: string): void => {
              this.handleModifySymptoms(symptoms);
            }
          })
        });
      }
    }
  }

  handleModifySymptoms(symptoms: string): void {
    if (this.booking) {
      this.booking.symptoms = symptoms;
      bookingStore.updateBooking(this.booking);
      this.refreshBooking();
      promptAction.showToast({ message: '修改成功' });
    }
  }

  build() {
    Column() {
      Text('预约等待')
        .fontSize(24)
        .fontWeight(FontWeight.Bold)
        .margin({ top: 20, bottom: 20 })

      if (this.booking) {
        Column() {
          Text(`预约号：${this.booking.id}`)
            .margin({ bottom: 10 })
          Text(`姓名：${this.booking.patientName}`)
            .margin({ bottom: 10 })
          Text(`状态：${this.booking.status === BookingStatus.WAITING ? '等待中' :
'待缴费'}`)
            .margin({ bottom: 10 })
          Text(`病症描述：${this.booking.symptoms}`)
            .margin({ bottom: 10 })

          if (this.booking.status === BookingStatus.PENDING_PAYMENT &&
this.booking.prescription) {
            Column() {
              Text('处方信息：')
                .fontWeight(FontWeight.Bold)
                .margin({ top: 20, bottom: 10 })
              Text(`医嘱：${this.booking.prescription.doctorAdvice}`)
              Text(`总费用：${this.booking.prescription.totalFee}元`)
                .margin({ top: 10 })
            }
            .margin({ bottom: 20 })
          }

          Button('修改病症')
            .width('80%')
```

```
            .margin({ bottom: 10 })
            .onClick(() => {
              this.dialogController.open();
            })

          Button('取消预约')
            .width('80%')
            .backgroundColor('#FF0000')
            .onClick(() => {
              bookingStore.removeBooking(this.bookingId);
              router.back();
            })

          Button('返回首页')
            .width('80%')
            .margin({ top: 10 })
            .backgroundColor('#8F8F8F')
            .onClick(() => {
              router.back();
            })
        }
        .padding(20)
      }
    }
    .width('100%')
    .height('100%')
  }
}
```

4.2 结果验证


04:27
医院挂号
医生入口
病例管理
查询预约状态
请输入预约号
查询
新建预约
陈女士
女
21
咳嗽发烧
提交预约
Code Linter


04:33
100
预约等待
预约号：1744057987602
姓名：陈女士
状态：等待中
病症描述：咳嗽发烧
修改病症
取消预约
返回首页
RecordPage.ets
dels/Booking
Store';
=>
Code Linter


RecordPage.ets
04:33
100
医院挂号
医生入口
病例管理
查询预约状态
17440579876O2
查询
新建预约
陈女士
女
21
咳嗽发烧
来自剪贴板
17440579876O2
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
符号
中/英
123
完成
下是新消息
Code Linter


RecordPage.ets
04:34
100
医生工作台
患者：陈女士　状态：等待中
患者：李爷爷　状态：等待中
返回首页
Code Linter

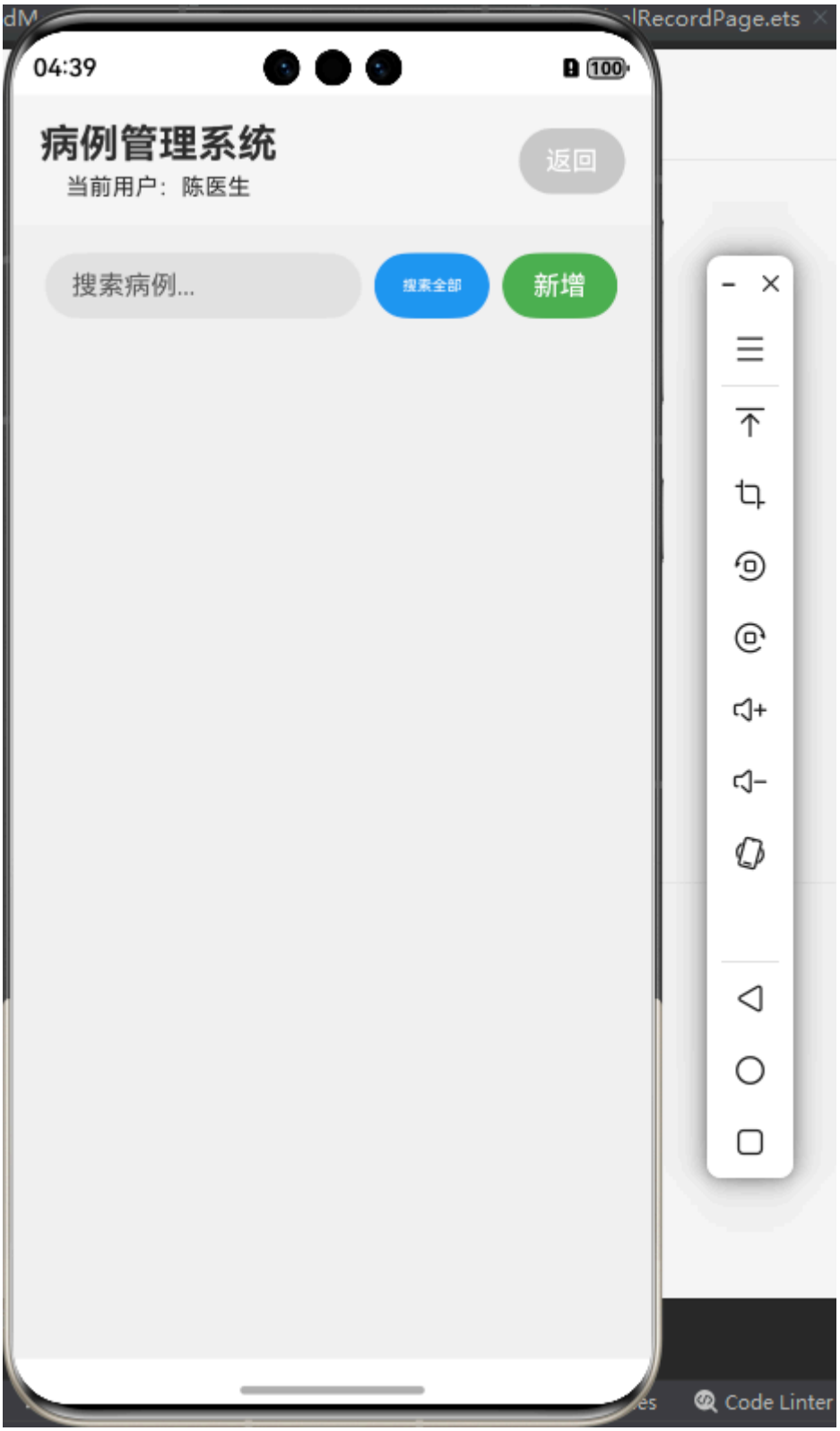

RecordPage.ets
04:36
100
医生工作台
患者：陈女士　状态：等待中
患者：李爷爷　状态：等待中
患者信息：
姓名：陈女士
性别：女
年龄：21
病症：咳嗽发烧
添加药品
请输入医嘱
请输入总费用
Code Linter


RecordPage.ets
04:35
100
医生工作台
患者：陈女士　状态：等待中
添加药品
药品名称
用药剂量
用药频率
药品价格
取消
确定
Code Linter


RecordPage.ets
04:38
100
医生工作台
患者：陈女士　状态：等待中
患者：李爷爷　状态：等待中
患者信息：
姓名：陈女士
性别：女
年龄：21
病症：咳嗽发烧
添加药品
止咳糖浆 - 20ml/次 - 一天3次　￥20
感冒药 - 一次1包 - 一天3次　￥25
多喝热水
50
Code Linter


RecordPage.ets
04:38
100
医生工作台
患者：陈女士　状态：待缴费
患者：李爷爷　状态：等待中
患者信息：
姓名：陈女士
性别：女
年龄：21
病症：咳嗽发烧
返回首页
Code Linter


RecordPage.ets
04:39
100
预约等待
预约号：1744057987602
姓名：陈女士
状态：待缴费
病症描述：咳嗽发烧
处方信息：
医嘱：多喝热水
总费用：50元
修改病症
取消预约
返回首页
Code Linter


RecordPage.ets
04:39
100
病例管理系统
当前用户：陈医生
返回
搜索病例...
搜索全部
新增
Code Linter


RecordPage.ets
04:40
100
病例管理系统
当前用户：陈医生
返回
搜索病例...
搜索全部
新增
编辑病例
陈女士
21
女
咳嗽发烧
多喝热水
50
取消
保存
Code Linter


RecordPage.ets
04:40
100
病例管理系统
当前用户：陈医生
返回
搜索病例...
搜索全部
新增
陈女士 21岁 女
症状：咳嗽发烧
医嘱：多喝热水
就诊金额：¥50.00
编辑
删除
Code Linter