

《HIS 医疗管理系统设计说明书（2025 级）》

（需求功能与系统约定）

1 系统概述与设计目标

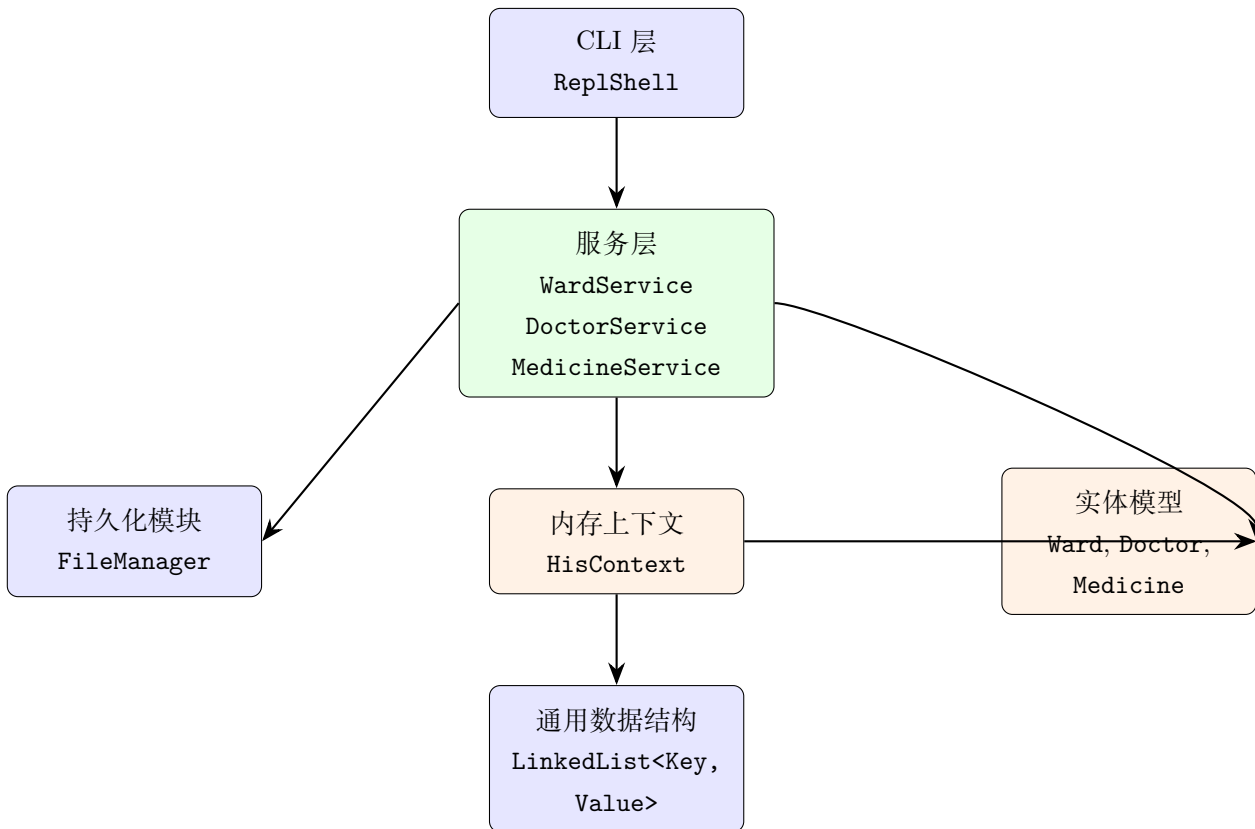
本系统为一个面向小型医院的轻量级医疗信息管理系统（Hospital Information System, HIS），采用 C++ 纯链表实现。通过模块化设计，实现业务逻辑与交互界面的解耦，保障系统的高扩展性与可维护性。

- **业务闭环**：实现完整的挂号、诊疗、检查、住院、开药等医疗生命周期。
- **数据规模**：支持至少 100 名门诊患者、30 名住院患者、20 名医生、5 个科室及 20 类药品。
- **底层要求**：全程基于手写链表实现内存数据流转，采用文本文件（JSON 数组）持久化。
- **鲁棒性**：具备极强的异常拦截能力，从容应对测试工程师的极端输入场景（防崩溃设计）。

2 系统架构与项目目录结构

系统采用分层架构设计（CLI 交互层 → Service 逻辑层 → Model 实体层 → Storage 持久层）。

- **CLI 层**：命令解析与结果输出
- **Service 层**：业务规则处理
- **Model 层**：数据结构定义
- **Storage 层**：数据存储与文件持久化



load 逻辑：WardService 先清空 HisContext.wards 再读文件填充，通常不造成累积内存泄漏。

图 1: HIS 系统模块架构图

项目工程文件结构如下：

```

his-project/
  CMakeLists.txt          # 根构建配置
  data/                   # 原始数据目录（评测时展示的文件）
    patients.txt          # 【存储JSON数组的文本文件】至少30条记录
    doctors.txt
    wards.txt             # 病房信息
    medicines.txt         # 药品库
  include/               # 头文件目录（.h / .hpp）
    models/              # 1. 纯数据实体层（Entity）
      patient.h
      doctor.h
      medicine.h
    utils/               # 2. 底层工具与驱动层
      linked_list.hpp    # 【核心】全手写的泛型链表 template<T>
      file_manager.h     # 文件读写封装
      e2hang_json.h      # 你的 C++20 JSON 解析库头文件
    core/               # 3. 核心业务层（his-core）
      his_context.h      # 全局上下文（持有所有内存链表的实例）
      patient_service.h  # 处理挂号、看诊等纯逻辑
      report_service.h   # 报表统计算法
    cli/                # 4. 交互表现层（his-cli）
      repl_shell.h       # 交互式命令行主循环
      format_printer.h   # 负责把链表数据打印成漂亮的 ASCII 表格
  src/                  # 源文件目录（.cpp）=> 源文件目录（实现）
    main.cpp             # 程序唯一入口：初始化、启动REPL和退出保存
    models/              # 实体类方法实现
      patient.cpp        # 患者属性操作及 JSON 序列化实现
      doctor.cpp         # 医生属性操作
      ward.cpp           # 病房/床位逻辑实现
      medicine.cpp       # 药品多名称匹配逻辑实现
    core/               # his-core: 核心业务实现（无输入输出）
      his_context.cpp    # 全局单例：管理所有内存链表的加载与保存
      patient_service.cpp # 实现挂号、诊疗记录、病历生成逻辑
      inpatient_service.cpp # 实现住院分配、床位流转逻辑
      pharmacy_service.cpp # 实现处方发药、库存增减、入库逻辑
      report_service.cpp # 实现多维度报表统计与数据分析算法
    utils/              # 基础设施实现
      file_manager.cpp   # 处理 .txt 文件的物理读写
      input_sanitizer.cpp # 【鲁棒性核心】针对测试工程师的输入清洗
      e2hang_json.cpp    # 你的 C++20 JSON 解析库实现
    cli/                # his-cli: 命令行交互实现
      repl_shell.cpp     # 交互式 Shell 主循环（REPL）
      command_parser.cpp # 指令分发器：将输入映射到 Service 函数
      table_printer.cpp  # 美化工具：将结果格式化为 ASCII 表格
  tests/                # 测试用例目录
    test_linked_list.cpp # 必须先确保链表绝对可靠
    test_core_logic.cpp  # 灌入异常数据测试业务层鲁棒性

```

3 数据实体与约定

3.1 通用数据约定

- **唯一性**：所有系统实体均采用**唯一 ID** 作为数据主标识。
- **宽容性**：允许患者姓名重复，系统必须在底层支持重名/同名情况下的精准区分。
- **存储规约**：所有内存态数据全程挂载并依托于**自定义泛型链表**进行流转；数据的磁盘持久化统一采用 **JSON 数组**格式存储于对应的 **TXT** 文件中。

3.2 患者 (Patient)

- **实体属性**：PatientID (唯一主键)、姓名、年龄、性别、联系方式、当前状态 (门诊 / 住院 / 已出院)。
- **系统约定**：一名患者的生命周期内可对应多条医疗记录；系统查询接口必须支持按患者姓名的**模糊查询**。

3.3 医生 (Doctor)

- **实体属性**：DoctorID (唯一主键)、姓名、所属科室 ID、职称 (主任 / 副主任 / 主治 / 住院医师)、出诊时间安排。
- **系统约定**：每位医生有且仅隶属于一个明确的科室；为保证医院各科室的接诊流转能力，约定**每个科室至少包含 3 名注册医生**。

3.4 科室 (Department)

- **实体属性**：DepartmentID (唯一主键)、科室名称。

3.5 医疗记录 (MedicalRecord) 【核心】

- **实体属性**：RecordID (唯一主键)、关联的 PatientID、关联的 DoctorID、业务类型 (挂号 / 看诊 / 检查 / 住院)、发生时间、详细结构化信息。
- **系统约定**：一条合法的医疗记录**必须同时强关联**当前存在的患者与医生；为保障防篡改性与溯源严肃性，医疗记录**不可直接修改**，必须采用“撤销废除 + 重建新单”的机制。

3.6 病房与床位 (Ward & Bed)

- **病房属性**：WardID (唯一主键)、病房类型 (普通 / 特殊 / ICU 等)、关联科室 ID、最大床位容量。
- **床位属性**：BedID (唯一主键)、所属病房 ID、当前状态 (空闲 / 已占用)、绑定的患者 ID。
- **系统约定**：同一时间段内，一个物理床位**仅能属于一名患者**；系统须支持基于患者住院/出院动作的**床位动态分配与自动释放**。

3.7 药品 (Medicine)

- **实体属性**：MedicineID (唯一主键)、通用名、商品名、别名、当前库存数量、所属科室、单价。
- **系统约定**：检索接口必须支持**多名称混合匹配**；在处方发药与出库操作中引入强校验拦截，任何情况下**药品库存均不得为负**。

4 核心功能需求

4.1 数据管理功能

- (1) 增加操作
 - 操作对象：添加患者、医生、医疗记录、药品信息。
 - 约束要求：支持单条命令行输入与 TXT 文件批量导入；自动校验合法性。
- (2) 修改操作
 - 操作对象：支持修改患者基本信息、修改药品属性信息。
 - 业务约定：医疗记录属于核心溯源数据，**不可直接修改**。
- (3) 删除操作
 - 操作对象：删除患者档案、删除医疗记录、删除下架药品。
 - 业务约束：必须进行严谨的**依赖检查**（例如：若患者当前仍在住院，则系统必须拦截并拒绝删除操作）。

4.2 查询功能

系统需支持灵活的多维度数据检索：

- 按患者查询：查询指定患者完整的历史就诊记录与生命周期。
- 按医生查询：查询特定医生的挂号列表与诊疗情况。
- 按科室查询：查询该科室的人员构成与整体运转信息。
- 按药品查询：根据名称查询药品的实时库存与单价信息。
- 技术要求：必须支持按姓名/药品名称的**模糊查询**，且支持**排序处理**。

4.3 住院管理功能

- 业务操作：为患者分配床位、办理出院处理、实时更新床位状态。
- 系统约定：办理住院时若无空闲床位，系统需自动拒绝；办理出院后系统需自动释放占用的床位资源。

4.4 药房管理功能

- 业务操作：药品入库（补给）、药品出库、凭医生处方发药。
- 系统约定：发药需强校验，**库存不足时明确禁止发药**并警报提示；处方开药数量须设上限。

4.5 报表与统计功能

系统需支持以下维度的核心统计报表：

- 医院营业额统计：综合汇总各项检查费用、药品开销与住院费用。
- 医生工作量统计：统计各医生/各科室的接诊患者数量与频次。
- 当前住院患者报表：生成当前全院或指定科室仍在住院的患者清单。
- 床位使用率统计：计算并展示各病房/科室床位的实时占用百分比。
- 药品库存统计：一览当前所有药品的库存余量状况及高频消耗趋势。

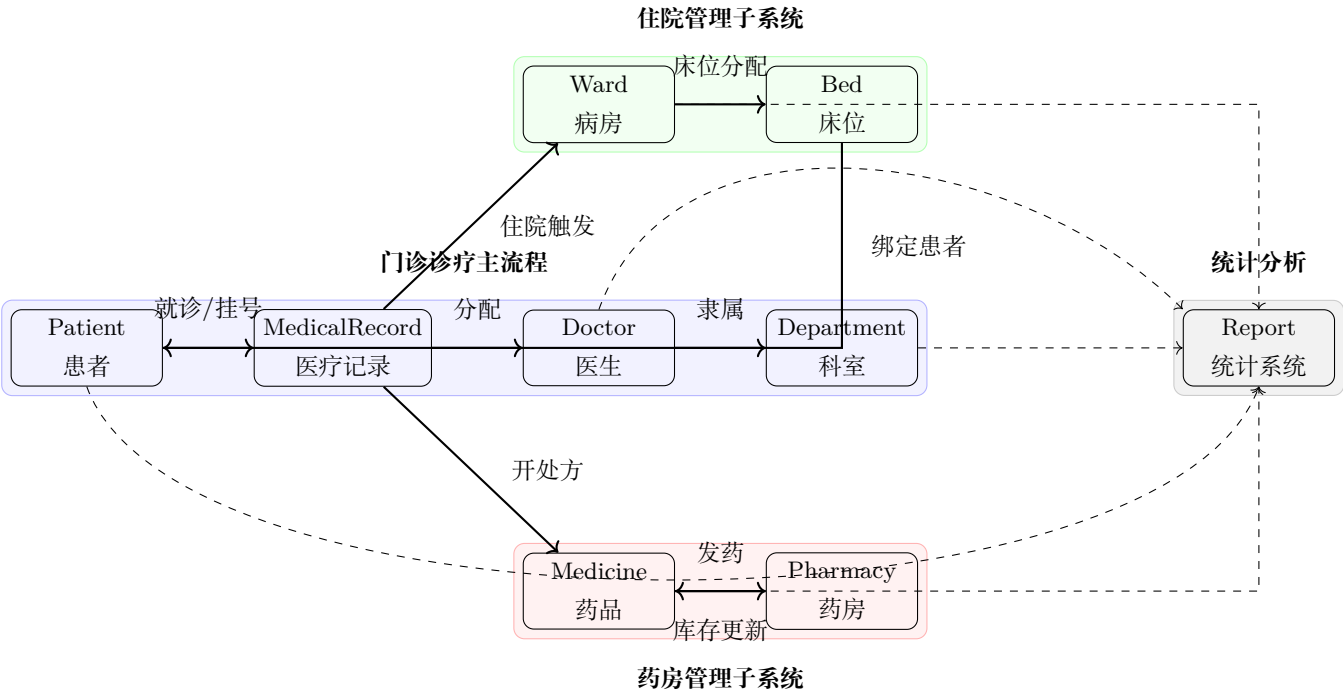
4.6 业务流程管理

- 住院管理：执行床位检索 → 分配绑定 → 状态更新 → 出院释放。无空床时业务拒绝。
- 药房管理：处理入库补给、处方强校验发药（库存不足禁止发药）、单次开药限量限制。

4.7 报表与统计（多权限视角）

支持格式化的 ASCII 表格打印输出：医院综合营业额、医生工作量排行、住院患者一览表、全院/科室床位使用率统计、药品库存消耗趋势。

5 系统核心交互流程图（完整业务流）



6 交互层 (CLI) 与鲁棒性设计

6.1 REPL 命令行交互

系统封装了交互式 Shell，支持如 `his > add patient`、`his > admit P001`、`his > report doctor` 等语义化指令，参数输入灵活，错误提示友好。

6.2 极致鲁棒性（防崩溃拦截）

结合 `input_sanitizer.cpp` 提供全方位输入保护：

- **输入异常**：拦截非法字符（防命令注入）、阻断空输入与超长字符串、自动纠正格式错误。
- **业务异常**：妥善处理无空床、库存穿透、非法 ID 关联引用、并发重名等业务死锁状态。

7 数据持久化与扩展机制

系统根目录 `data/` 下维持 `patients.txt` 等存储媒介。采用 JSON 数组格式。遵循“启动时全量挂载，执行关键操作或退出时序列化落盘”的策略。后期可进一步无缝拓展：历史数据预测分析（如病床流转预测）、医生智能推荐以及多角色权限 (RBAC) 认证系统。