# </> DocsGen

TECHNICAL INTELLIGENCE REPORT

## HospitalManagement.py

Generated on: 2026-01-09 22:32

# Executive Summary

This code defines the structure and core interactions within a hospital setting, focusing on staff, patients, and hospital operations. It outlines the relationships and basic functionalities required to manage hospital personnel and patients.

# Purpose & Responsibility

**PURPOSE:**

The goal of this code is to create a foundational structure for managing hospital staff, patients, and departments.

**Responsibility:** It aims to facilitate basic operations such as hiring staff and prescribing medication.

# Key Capabilities

- Define hospital staff roles (doctors and nurses)
- Manage patient information
- Handle hospital operations, including staff hiring

# Class: Staff

**PURPOSE:**

Represents a general staff member in the hospital.

## Constructor: __init__(self, staff_id: int, name: str)

**LOGIC FLOW:**

1. Assigns the provided `staff_id` to the instance variable `self.staff_id`.
2. Assigns the provided `name` to the instance variable `self.name`.

# Class: Doctor

**PURPOSE:**

Represents a doctor, a type of staff member with a specialty.

## Constructor: __init__(self, staff_id: int, name: str, specialty: str)

**LOGIC FLOW:**

1. Calls the constructor of the parent class `Staff` with `staff_id` and `name`.
2. Assigns the provided `specialty` to the instance variable `self.specialty`.
3. Initializes an empty list `self.patients` to keep track of the doctor's patients.

## Method: prescribe_medication(self, patient, medicine: str)

**LOGIC FLOW:**

1. Returns a string indicating that the doctor with `self.name` prescribed `medicine` for the patient with `patient.name`.

# Class: Nurse

**PURPOSE:**

Represents a nurse, a type of staff member with a shift.

## Constructor: __init__(self, staff_id: int, name: str, shift: str)

**LOGIC FLOW:**

1. Calls the constructor of the parent class `Staff` with `staff_id` and `name`.
2. Assigns the provided `shift` to the instance variable `self.shift`.

# Class: Patient

**PURPOSE:**

Represents a patient in the hospital.

## Constructor: __init__(self, patient_id: int, name: str, illness: str)

**LOGIC FLOW:**

1. Assigns the provided `patient_id` to the instance variable `self.patient_id`.
2. Assigns the provided `name` to the instance variable `self.name`.
3. Assigns the provided `illness` to the instance variable `self.illness`.

# Class: Hospital

**PURPOSE:**

Represents a hospital with staff members and departments.

## Constructor: __init__(self, hospital_name: str)

**LOGIC FLOW:**

1. Assigns the provided `hospital_name` to the instance variable `self.hospital_name`.
2. Initializes an empty list `self.staff_members` to keep track of the hospital's staff.
3. Initializes a list `self.departments` with the names of the hospital's departments: "Emergency", "Radiology", "Surgery".

## Method: hire_staff(self, member: Staff)

**LOGIC FLOW:**

1. Adds the provided `member` to the list `self.staff_members`.

# Source Code Analysis

HospitalManagement.py

CODE

```
class Staff:
    def __init__(self, staff_id: int, name: str):
        self.staff_id = staff_id
        self.name = name

class Doctor(Staff):
    def __init__(self, staff_id: int, name: str, specialty: str):
        super().__init__(staff_id, name)
        self.specialty = specialty
        self.patients = []

    def prescribe_medication(self, patient, medicine):
        return f"Doctor {self.name} prescribed {medicine} for {patient.name}"

class Nurse(Staff):
    def __init__(self, staff_id: int, name: str, shift: str):
        super().__init__(staff_id, name)
        self.shift = shift

class Patient:
    def __init__(self, patient_id: int, name: str, illness: str):
        self.patient_id = patient_id
        self.name = name
        self.illness = illness

class Hospital:
    def __init__(self, hospital_name: str):
        self.hospital_name = hospital_name
        self.staff_members = []
        self.departments = ["Emergency", "Radiology", "Surgery"]

    def hire_staff(self, member: Staff):
        self.staff_members.append(member)
```