Challenge April-2026
Agentic Medical Services

A solution with DT5GL/SQL/Python/Claude
by Jack Jansonius – 13 May 2026

Problem Statement (from the website):

This challenge aims to explore how LLMs orchestrate rule-based decision services. Consider the following three loosely coupled medical services that should be used by an LLM to help a doctor determine a therapy for a patient with Acute Sinusitis.

1. Determine Medication and Dosing

- If the patient is 18 years old or older, the therapy of choice is Amoxicillin.
- If the patient is younger than 18, the therapy of choice is Cefuroxime.
- If the patient is penicillin-allergic, the therapy of choice is Levofloxacin.

- For patients between 15 and 60, the dose is 500 mg every 24 hours for 14 days.
- For patients younger than 15 or older than 60, the dose is 200 mg every 24 hours for 14 days.
- If the patient's creatinine level (PCr) > 1.4 and creatinine clearance (CCr) < 50, the dose is 200 mg every 24 hours for 14 days.

2. Determine Creatinine Clearance

Creatinine clearance (CCr) should be calculated using this formula:
CCr [ml/min] = ((140 – age) × Lean Body Weight [kg])/ (PCr × 72)

3. Drug Interaction Rules

Check if the patient is on active medications. If so, check for possible conflicts between recommended and active medications using the file ConflictingMedications.csv. All detected conflicts should generate the appropriate warnings.
Tell your LLM to use these three decision services when answering any therapy request for a patient with Acute Sinusitis. Here is a possible request: "*I have a patient diagnosed with Acute Sinusitis. He is 58 years old, weighs 78 kg, and has a creatinine level of 1.85. Keep in mind that he is Penicillin-allergic and takes Coumadin.*"

Problem analysis

The rules for this challenge do remind us of the expert systems of the 1980s, in which knowledge was recorded in the form of if-then rules to arrive at a solution to a problem based on *successive information gathering*. The challenge was to ask exactly the right number of questions, *in order of relevance*[1], that would contribute to solving the problem at hand.

This challenge arises precisely in the three rules for determining the dosage:

1. For patients between 15 and 60, the dose is **500** mg every 24 hours for 14 days.
2. For patients younger than 15 or older than 60, the dose is **200** mg every 24 hours for 14 days.
3. If the patient's creatinine level (PCr) > 1.4 and creatinine clearance (CCr) < 50, the dose is **200** mg every 24 hours for 14 days.

It is clear to see here: rule 3 overrides rule 1 but not rule 2. Therefore, rule 2 must be evaluated first, and 'age < 15' and 'age > 60' are *the most relevant* conditions. So if a person is 14 or 61 years old, the conditions creatinine level (PCr) > 1.4 and creatinine clearance (CCr) < 50 are *irrelevant* (as indicated by a dash in the table below).

Overall, the decision table with the conditions *in order of relevance* is therefore:

```
Table 2: Determine Dosage
If:                                      | 0| 1| 2| 3| 4|
Age < 15                                 | Y| N| N| N| N|
Age > 60                                 | -| Y| N| N| N|
Patient_Creatinine_Level > 1.4           | -| -| Y| Y| N|
Patient_Creatinine_Clearance < 50.0      | -| -| Y| N| -|
Then:
Dosage is Half                           | X| X| X|  |  |
Dosage is Full                           |  |  |  | X| X|
# .......
```

If the rules for this challenge are implemented as an expert system, then for a person aged 14 or under or 61 or over, the system will not ask for the specified PCr, CCr and weight (for calculating the CCr).

For a person aged between 15 and 60, in addition to age, the patient's creatinine level (PCr) will be requested; however, if this is 1.4 or lower, the formula for creatinine clearance (CCr) will not be applied and therefore the patient's weight required for this calculation will not be requested either.

To demonstrate all this, I first created an expert system-style prototype in which the application asks the user for everything; even whether there is a conflict between the recommended and the active medication.

I then expanded this simple interactive application so that, based on an identification number, the patient's name and date of birth are retrieved from a database table and potential conflicts between the recommended and active medication are automatically checked using the ConflictingMedications.csv file.

---

[1] The fact that *relevance* is a key concept in both human and artificial (decision-making) intelligence will not be explored further here. However, this problem analysis has resulted in an article on LinkedIn: https://www.linkedin.com/pulse/stop-treating-decision-tables-like-programs-jack-jansonius-jfpce/

Version 1: expertsystem-style prototype:

```
# Quick expertsystem-style prototype

rTable 0: Determine patient therapy
If:                                     | 0| 1| 2|
'Doctor has determined Acute Sinusitus' | Y| Y| N|
Medication <> ""                        | Y| Y| -|
Dosage is Half                          | Y| N| -|
Dosage is Full                          | -| Y| -|
Then:
Therapy is Half_Dosage                  | X|  |  |
Therapy is Full_Dosage                  |  | X|  |
Therapy is Not_Applicable               |  |  | X|
# .......

Proposition: 'Doctor has determined Acute Sinusitus'
Askable_using: "**?"


Table 1: Determine Medication
If:                                     | 0| 1| 2|
'Patient is allergic to Penicillin'     | Y| N| N|
Age >= 18                               | -| Y| N|
Then:
Medication = "Levofloxacin"             | X|  |  |
Medication = "Amoxicillin"              |  | X|  |
Medication = "Cefuroxime"               |  |  | X|
# .......

Proposition: 'Patient is allergic to Penicillin'
Askable_using: "**?"


Table 2: Determine Dosage
If:                                     | 0| 1| 2| 3| 4|
Age < 15                                | Y| N| N| N| N|
Age > 60                                | -| Y| N| N| N|
Patient_Creatinine_Level > 1.4          | -| -| Y| Y| N|
Patient_Creatinine_Clearance < 50.0     | -| -| Y| N| -|
Then:
Dosage is Half                          | X| X| X|  |  |
Dosage is Full                          |  |  |  | X| X|
# .......


rTable 3: Drug interaction
If:                                     | 0| 1|
'Doctor has determined Acute Sinusitus' | Y| Y|
Medication = "Levofloxacin"             | Y| -|
Medication = "Cefuroxime"               | -| Y|
Active_Medication is Coumadin           | Y| Y|
Then:
'Drug interaction warning'              | X| X|
# .......
```

```
GoalAttribute: Therapy

Case: Full_Dosage
Print: "------------------------------------------------------------"
Print: "Recommended Medication: %s." Medication
Print: "500mg every 24 hours for 14 days"
Print: "------------------------------------------------------------"
Print: "Calculated Creatinine Clearance: %s" Patient_Creatinine_Clearance.getvalue

Case: Half_Dosage
Print: "------------------------------------------------------------"
Print: "Recommended Medication: %s." Medication
Print: "250mg every 24 hours for 14 days"
Print: "------------------------------------------------------------"
Print: "Calculated Creatinine Clearance: %s" Patient_Creatinine_Clearance.getvalue

Case: Not_Applicable
Print: "------------------------------------------------------------"
Print: "Sorry, this decision model can handle only Acute Sinusitus "
Print: "------------------------------------------------------------"


GoalProposition: 'Drug interaction warning'
Print: "Warning: Coumadin and %s can result in reduced effectiveness of Coumadin."
Medication

Attribute: Active_Medication
Askable_using: "Is the patient on medication now?"

Attribute: Age
Askable_using: "What is the years of age of the patient?"

Attribute: Weight    Type: Real
Askable_using: "What is the weight of the patient?"

Attribute: Patient_Creatinine_Level
Askable_using: "What is the creatine level of the patient?"

Attribute: Patient_Creatinine_Clearance
Equals: round((140 - Age) * Weight / (Patient_Creatinine_Level*72), 2)
```

Testruns version 1

PS C:\Users\jjans\dt5gl2504> .\dt.exe[2] -source:sinusitis_therapy_v1.txt -nti

'Doctor has determined Acute Sinusitus' (y/n)? *> y
'Patient is allergic to Penicillin' (y/n)? *> y
"What is the years of age of the patient?"
(int)> 14

Recommended Medication: Levofloxacin.
250mg every 24 hours for 14 days

Calculated Creatinine Clearance: None

"Is the patient on medication now?"
1. Coumadin
2. Otherwise
> 1

Warning: Coumadin and Levofloxacin can result in reduced effectiveness of Coumadin.

PS C:\Users\jjans\dt5gl2504> .\dt.exe -source:sinusitis_therapy_v1.txt -nti
'Doctor has determined Acute Sinusitus' (y/n)? *> y
'Patient is allergic to Penicillin' (y/n)? *> n
"What is the years of age of the patient?"
(int)> 58

"What is the creatine level of the patient?"
(real)> 1.4

Recommended Medication: Amoxicillin.
500mg every 24 hours for 14 days

Calculated Creatinine Clearance: None

2 DT.exe is Python code (Version 3.6), compiled to C, and runs directly under Windows (without pre-installation of Python). Download and all necessary files available at: https://github.com/JackJansonius/DT5GL

PS C:\Users\jjans\dt5gl2504> .\dt.exe -source:sinusitis_therapy_v1.txt -nti
'Doctor has determined Acute Sinusitus' (y/n)? *> y
'Patient is allergic to Penicillin' (y/n)? *> y
"What is the years of age of the patient?"
(int)> 58

"What is the creatine level of the patient?"
(real)> 1.85

"What is the weight of the patient?"
(real)> 78

Recommended Medication: Levofloxacin.
250mg every 24 hours for 14 days

Calculated Creatinine Clearance: 48.02

"Is the patient on medication now?"
1. Coumadin
2. Otherwise
> 1

Warning: Coumadin and Levofloxacin can result in reduced effectiveness of Coumadin.

PS C:\Users\jjans\dt5gl2504> .\dt.exe -source:sinusitis_therapy_v1.txt -nti
'Doctor has determined Acute Sinusitus' (y/n)? *> y
'Patient is allergic to Penicillin' (y/n)? *> n
"What is the years of age of the patient?"
(int)> 15

"What is the creatine level of the patient?"
(real)> 1.5

"What is the weight of the patient?"
(real)> 55

Recommended Medication: Cefuroxime.
500mg every 24 hours for 14 days

Calculated Creatinine Clearance: 63.66

"Is the patient on medication now?"
1. Coumadin
2. Otherwise
> 1

Warning: Coumadin and Cefuroxime can result in reduced effectiveness of Coumadin.

Version 2: extended sinusitus diagnose application

```
# Extended sinisitus diagnose application

# Ask if the next patient should be evaluated; No => exit.
# Ask for patient ID; if not in the database, display an error message
# Retrieve name and date of birth from the database; display id, name and age on
the current date to the user.
# Ask other relevant questions regarding medication and dosage.
# Ask the user about other medications to determine interactions with the proposed
medication.


PostgreSQL_database: "sinusitus"
# SQLite_database: "database/sinusitus.db"

Initial_instructions:
>>: Yes = 1          # Done,True
End_Instructions


rTable 0: Determine patient therapy
If:                                     | 0| 1| 2| 3|
'Therapy request for next patient'      | Y| Y| Y| N|
'Patient found in database'             | Y| Y| N| -|
Greeting_text = Yes                     | Y| Y| -| -|
Medication <> ""                        | Y| Y| -| -|
Dosage is Half                          | Y| N| -| -|
Dosage is Full                          | -| Y| -| -|
Ask_active_medications = Yes            | Y| Y| -| -|
Then:
Outcome is Stop_application             |  |  |  | X|
Outcome is Patient_not_found            |  |  | X|  |
Outcome is Half_Dosage                  | X|  |  |  |
Outcome is Full_Dosage                  |  | X|  |  |
Check_Drug_Interaction is Yes           | X| X|  |  |
# .......

Proposition: 'Therapy request for next patient'
Askable_using: "**?"

Proposition: 'Patient found in database'
Obtain_instance_from_database_view: patient

Attribute: Greeting_text            Type: Integer
Equals: print_texts("Patient ", patient_id, ". ", patient.name, "; date of birth:
", patient.date_of_birth, " is now ", str(Age), " years old. ")
# returns a 1 after

Attribute: Ask_active_medications   Type: Integer
Equals: AskActiveMedications()
# returns a 1 after


Table 1: Determine Medication
If:                                     | 0| 1| 2|
'Patient is allergic to Penicillin'     | Y| N| N|
Age >= 18                               | -| Y| N|
Then:
Medication = "Levofloxacin"             | X|  |  |
Medication = "Amoxicillin"              |  | X|  |
Medication = "Cefuroxime"               |  |  | X|
# .......

Proposition: 'Patient is allergic to Penicillin'
Askable_using: "**?"
```

```
Table 2: Determine Dosage
If:                                     | 0| 1| 2| 3| 4|
Age < 15                                | Y| N| N| N| N|
Age > 60                                | -| Y| N| N| N|
Patient_Creatinine_Level > 1.4          | -| -| Y| Y| N|
Patient_Creatinine_Clearance < 50.0     | -| -| Y| N| -|
Then:
Dosage is Half                          | X| X| X|  |  |
Dosage is Full                          |  |  |  | X| X|
# .......

rTable 3: Drug interaction
If:                                     | 0| 1|
Check_Drug_Interaction is Yes           | Y| Y|
Conflicts_detected = Yes                | Y| N|
Active_medications_reported = Yes       | -| Y|
Then:
Drug_Interaction_Warnings is Yes        | X|  |
Drug_Interaction_Warnings is No         |  | X|
# .......

Attribute: Conflicts_detected            Type: Integer
Equals: ConflictsDetected_with(Medication)
Attribute: Active_medications_reported   Type: Integer
Equals: ActiveMedicationsReported()
# returns a 1 after

Attribute: dummy  Type: Integer


GoalAttribute: Outcome
Repeat_until: Stop_application

Case: Stop_application
Print: "---- Ended ------ "

Case: Patient_not_found
Print: "Patient not found! "

Case: Half_Dosage
Print: "-----------------------------------------------------------"
Print: "Recommended Medication: %s." Medication
Print: "250mg every 24 hours for 14 days"
Print: "-----------------------------------------------------------"
Print: "Calculated Creatinine Clearance: %s" Patient_Creatinine_Clearance.getvalue

Case: Full_Dosage
Print: "-----------------------------------------------------------"
Print: "Recommended Medication: %s." Medication
Print: "500mg every 24 hours for 14 days"
Print: "-----------------------------------------------------------"
Print: "Calculated Creatinine Clearance: %s" Patient_Creatinine_Clearance.getvalue


GoalAttribute: Drug_Interaction_Warnings

Case: Yes
>>: dummy = Print_Interaction_Warnings(Medication)

Case: No
Print: "==> No known interactions between %s and the reported medications."
Medication
Print: "         "

Attribute: patient.name           Type: Text
Attribute: patient.date_of_birth  Type: Text
```

```
Attribute: Age
Equals: actual_age(date_today(), patient.date_of_birth)
# date formats: 'yyyy-mm-dd'

Attribute: Weight    Type: Real
Askable_using: "What is the weight of the patient?"

Attribute: Patient_Creatinine_Level
Askable_using: "What is the creatine level of the patient?"

Attribute: Patient_Creatinine_Clearance
Equals: round((140 - Age) * Weight / (Patient_Creatinine_Level*72), 2)

Attribute: patient_id   Type: Integer
Equals: int(input("Enter patient identification number : "))

Database_view: patient
With_attributes: name, date_of_birth
Query:
       select name, date_of_birth
       from patient
       where id = '%s'
With_arguments: patient_id
```

Added to DTFunctions.py:

```
######################################################################
# functions for challenge: Apr-2026 Agentic Medical Services ######
# https://dmcommunity.org/challenge-apr-2026/                ######
######################################################################


from datetime import date, datetime, timedelta
from dateutil.relativedelta import relativedelta

def actual_age(actual_date, date_of_birth):           # date formats: 'yyyy-mm-dd'
    u = datetime.strptime(actual_date, "%Y-%m-%d").date()
    g = datetime.strptime(date_of_birth, "%Y-%m-%d").date()
    return relativedelta(u, g).years


def date_today():                # Return today's date in the format 'yyyy-mm-dd'
    return date.today().strftime("%Y-%m-%d")

def print_texts(*texts):
    print(*texts, sep="")
    return 1



# ------------------------------------------------------------------------
# DRUG INTERACTION CHECKER   FROM CLAUDE (SONNET 4.6)
# ------------------------------------------------------------------------

import csv
import io
import urllib.request


CSV_URL = (
    "https://raw.githubusercontent.com/DMCommunity/dmcommunity_shared/"
    "master/ConflictingMedications.csv"
)
```

```
FALLBACK_CSV = """Medication 1,Medication 2,Interaction,Risk
Levofloxacin,Coumadin,Increases anticoagulant effect,Elevated bleeding risk
(restricted fallback csv)
Levofloxacin,"Antacids (aluminum, magnesium)",Reduces levofloxacin
absorption,Reduced antibiotic effectiveness
Amoxicillin,Methotrexate,Reduces methotrexate clearance,Methotrexate toxicity risk
Amoxicillin,Lithium,Raises lithium levels,Lithium toxicity risk
Amoxicillin,"NSAIDs (ibuprofen, naproxen)",Increases amoxicillin levels
slightly,Mild toxicity risk
Cefuroxime,Coumadin,Increases anticoagulant effect,Elevated bleeding risk
Cefuroxime,"NSAIDs (ibuprofen, naproxen)",Increases cefuroxime levels slightly,Mild
toxicity risk
Cefuroxime,Lithium,Raises lithium levels,Lithium toxicity risk
"""

from typing import List, Dict

def load_conflicts() -> List[Dict]:
    """Fetch conflict CSV from GitHub; fall back to embedded copy on error."""
    try:
        with urllib.request.urlopen(CSV_URL, timeout=5) as resp:
            content = resp.read().decode("utf-8")
        print("  [Drug DB] Conflict database loaded from GitHub.")
    except Exception:
        print("  [Drug DB] Could not reach GitHub – using built-in database.")
        content = FALLBACK_CSV

    reader = csv.DictReader(io.StringIO(content))
    return list(reader)


def find_conflicts(drug: str, patient_meds: List[str], conflict_db: List[Dict]) ->
List[Dict]:
    """
    Return rows from conflict_db where 'Medication 1' matches *drug* and
    'Medication 2' fuzzy-matches any medication the patient is taking.
    Matching is case-insensitive and substring-based so that generic
    names match branded equivalents (e.g. "Coumadin" matches "Coumadin").
    """
    hits = []
    drug_lower = drug.lower()
    for row in conflict_db:
        if row["Medication 1"].strip().lower() != drug_lower:
            continue
        db_med2 = row["Medication 2"].strip().lower()
        for pm in patient_meds:
            pm_lower = pm.lower()
            # Match if either is a substring of the other
            if pm_lower in db_med2 or db_med2 in pm_lower:
                hits.append(row)
                break
    return hits

def separator(char="─", width=60):
    print(char * width)

def section(title: str):
    print()
    separator()
    print(f"  {title}")
    separator()
```

```
def ask_list(prompt: str) -> List[str]:
    """Ask for a comma-separated list; return stripped, non-empty tokens."""
    raw = input(f"{prompt} ").strip()
    if not raw:
        return []
    return [item.strip() for item in raw.split(",") if item.strip()]


patient_meds: List[str] = []
conflicts: List[Dict] = []

def AskActiveMedications():
    global patient_meds
    patient_meds = ask_list("List all active medications (comma-separated; enter =
none):")
    return 1


def ConflictsDetected_with(drug):
    global patient_meds, conflicts
    if patient_meds:
        print()
        conflict_db = load_conflicts()
        conflicts = find_conflicts(drug, patient_meds, conflict_db)
    return 1 if conflicts else 0


def ActiveMedicationsReported():
    global patient_meds
    return 1 if patient_meds else 0


def Print_Interaction_Warnings(drug):
    global patient_meds, conflicts

    if patient_meds:
        print()
        print(f"  Active medications : {', '.join(patient_meds)}")

    if conflicts:
        section("⚠  Drug Interaction Warnings")
        for c in conflicts:
            print(f"  • {drug} ↔ {c['Medication 2'].strip()}")
            print(f"      Interaction : {c['Interaction'].strip()}")
            print(f"      Risk        : {c['Risk'].strip()}")
            print()
        print(
            "  Please consult a clinical pharmacist before prescribing "
            f"{drug} alongside the above medications.\n" )

    patient_meds = []
    conflicts = []

    return 1
```

Testrun version 2

	id [PK] integer	name text	date_of_birth date
1	1	patientis14 (postgres)	2012-02-12
2	2	patientis58 (postgres)	1968-02-12
3	3	patientis15 (postgres)	2011-02-12
4	4	patientis18 (postgres)	2008-02-12

PS C:\Users\jjans\dt5gl2504> .\dt.exe -source:sinusitis_therapy_v2.txt -nti

'Therapy request for next patient' (y/n)? *> y
Enter patient identification number : 7
Patient not found!

'Therapy request for next patient' (y/n)? *> y
Enter patient identification number : 1
Patient 1. patientis14 (postgres); date of birth: 2012-02-12 is now 14 years old.
'Patient is allergic to Penicillin' (y/n)? *> y
List all active medications (comma-separated; enter = none): Coumadin
---
Recommended Medication: Levofloxacin.
250mg every 24 hours for 14 days

Calculated Creatinine Clearance: None

[Drug DB] Conflict database loaded from GitHub.

Active medications : Coumadin

⚠ Drug Interaction Warnings

- Levofloxacin ↔ Coumadin
  Interaction : Increases anticoagulant effect
  Risk : Elevated bleeding risk

Please consult a clinical pharmacist before prescribing Levofloxacin alongside the above medications.

'Therapy request for next patient' (y/n)? *> y
Enter patient identification number : 2
Patient 2. patientis58 (postgres); date of birth: 1968-02-12 is now 58 years old.
'Patient is allergic to Penicillin' (y/n)? *> n
"What is the creatine level of the patient?"
(real)> 1.4

List all active medications (comma-separated; enter = none):

Recommended Medication: Amoxicillin.
500mg every 24 hours for 14 days

Calculated Creatinine Clearance: None

'Therapy request for next patient' (y/n)? *> y
Enter patient identification number : 2
Patient 2. patientis58 (postgres); date of birth: 1968-02-12 is now 58 years old.
'Patient is allergic to Penicillin' (y/n)? *> y
"What is the creatine level of the patient?"
(real)> 1.85

"What is the weight of the patient?"
(real)> 78

List all active medications (comma-separated; enter = none): Coumadin
---
Recommended Medication: Levofloxacin.
250mg every 24 hours for 14 days

Calculated Creatinine Clearance: 48.02

[Drug DB] Conflict database loaded from GitHub.

Active medications : Coumadin

⚠ Drug Interaction Warnings

- Levofloxacin ↔ Coumadin
  Interaction : Increases anticoagulant effect
  Risk : Elevated bleeding risk

Please consult a clinical pharmacist before prescribing Levofloxacin alongside the above medications.

'Therapy request for next patient' (y/n)? *> y
Enter patient identification number : 3
Patient 3. patientis15 (postgres); date of birth: 2011-02-12 is now 15 years old.
'Patient is allergic to Penicillin' (y/n)? *> n
"What is the creatine level of the patient?"
(real)> 1.5

"What is the weight of the patient?"
(real)> 55

List all active medications (comma-separated; enter = none): Coumadin, Aspirin, Lithium
---
Recommended Medication: Cefuroxime.
500mg every 24 hours for 14 days

Calculated Creatinine Clearance: 63.66

[Drug DB] Conflict database loaded from GitHub.

Active medications : Coumadin, Aspirin, Lithium

⚠ Drug Interaction Warnings

- Cefuroxime ↔ Coumadin
  Interaction : Increases anticoagulant effect
  Risk : Elevated bleeding risk

- Cefuroxime ↔ Lithium
  Interaction : Raises lithium levels
  Risk : Lithium toxicity risk

Please consult a clinical pharmacist before prescribing Cefuroxime alongside the above medications.

'Therapy request for next patient' (y/n)? *> n
---- Ended ------

Asking the LLM's

It's interesting to use various LLMs to generate an expert-system-style solution in Python for this challenge.

To do this, I prefixed the challenge text with the following instruction:

> Please provide a Python solution that solves the following problem to help a doctor determine a treatment for a patient with acute sinusitis.
>
> Design it in the style of an expert system, where a solution is reached through a series of questions. Ask exactly enough questions (in order of relevance) to arrive at a suitable therapy or to conclude that no suitable therapy is known.
>
> Use the following formula:
> Patient_Creatine_Clearance = (140 - Age) * Weight / (Patient_Creatine_Level*72)

I also removed all references to service and orchestration.

Neither the free version of ChatGPT nor Claude performed well when it came to asking a *sequence of relevant* questions.

Here are the results for a 14-year-old with a penicillin allergy who is taking Coumadin. In this case, therefore, no questions need to be asked about creatinine level or weight. Both LLMs thus request more input data than necessary.

ChatGPT:

```
=== Acute Sinusitis Therapy Expert System ===
Patient age: 14
Patient weight (kg): 55
Creatinine level (PCr): 1.3
Penicillin allergic? (yes/no): yes
Active medications (comma separated, or none): Coumadin

=== Recommendation ===
Medication: Levofloxacin
Dose: 200 mg every 24 hours for 14 days
Creatinine Clearance (CCr): 74.04 ml/min

=== Warnings ===
⚠ Interaction detected between Levofloxacin and Coumadin
```

Claude Sonnet 4.6:

Acute Sinusitis Therapy Expert System

This system will ask a series of clinical questions to
recommend an appropriate antibiotic therapy.

Patient Demographics

Patient age (years): 14
Is the patient penicillin-allergic? [y/n]: y

Renal Function Assessment

Patient weight (kg): 55
Patient serum creatinine level – PCr (mg/dL): 1.3

Current Medications

Is the patient currently taking any medications? [y/n]: y
List all active medications (comma-separated): Coumadin[3]

[Drug DB] Conflict database loaded from GitHub.

Therapy Recommendation

Diagnosis : Acute Sinusitis
Patient Age : 14 years
Patient Weight : 55.0 kg
PCr : 1.30 mg/dL
CCr (calc.) : 74.0 ml/min

▶ Recommended Drug : Levofloxacin
▶ Dose : 200 mg every 24 hours for 14 days
▶ Dose rationale : Age 14 is outside the 15–60 range

Active medications : Coumadin

⚠ Drug Interaction Warnings

• Levofloxacin ↔ Coumadin
Interaction : Increases anticoagulant effect
Risk : Elevated bleeding risk

Please consult a clinical pharmacist before prescribing Levofloxacin alongside the above medications.

---

[3] That I have incorporated parts of Claude's Python code into my own solution is clearly evident from this point onwards.