

Medical Appointment Scheduling Agent - Assessment

Overview

Build an intelligent conversational agent that helps patients schedule medical appointments through Calendly integration. The agent should understand available time slots, ask relevant questions, answer FAQs using RAG, and handle scheduling intelligently.

Problem Statement

Create a conversational scheduling system that:

1. Integrates with Calendly API to fetch available and booked time slots
2. Engages in natural conversation to understand patient needs
3. Intelligently suggests available time slots based on preferences
- N. Answers frequently asked questions using RAG
- U. Books appointments through Calendly API
6. Handles rescheduling and cancellation requests
7. Seamlessly switches between scheduling and FAQ answering

Technical Requirements

Tech Stack

- **Backend:** FastAPI (Python 3.10+)
- **LLM:** Any (OpenAI, Anthropic, Llama, Mistral, etc.) - Document your choice
- **Vector Database:** Your choice (Pinecone, ChromaDB, Weaviate, Qdrant, etc.)
- **Calendar API:** Calendly API (or mock implementation)
- **Frontend** (if fullstack): React with chat interface

Core Features

1. Calendly Integration

- Fetch doctor's schedule (working hours, existing appointments)
- Get available time slots dynamically
- Create new appointments
- Handle different appointment types with durations:
 - General Consultation: 30 minutes
 - Follow-up: 15 minutes
 - Physical Exam: 45 minutes
 - Specialist Consultation: 60 minutes

2. Intelligent Conversation Flow

Phase 1: Understanding Needs

- Greet patient warmly
- Understand reason for visit
- Determine appointment type
- Ask about preferred date/time

Phase 2: Slot Recommendation

- Show 3-5 available slots based on preferences
- Explain why slots are suggested
- Handle "none of these work" gracefully
- Offer alternative dates/times

Phase 3: Booking Confirmation

- Collect necessary information:
 - Patient name
 - Phone number
 - Email
 - Reason for visit
- Confirm all details before booking
- Create appointment via Calendly
- Provide confirmation details

3. FAQ Knowledge Base (RAG)

Create RAG system for clinic information:

- ♦ **Clinic Details:**
 - Location and directions
 - Parking information
 - Hours of operation

- ♦ **Insurance & Billing:**

- Accepted insurance providers
- Payment methods
- Billing policies

- ♦ **Visit Preparation:**

- Required documents
- First visit procedures
- What to bring

- ♦ **Policies:**

- Cancellation policy
- Late arrival policy
- COVID-19 protocols

Seamless Context Switching:

- User asks FAQ in middle of booking → Answer, then return to booking
- User wants to schedule after FAQ → Smoothly transition to scheduling
- User asks multiple FAQs → Answer all, maintain context

4. Smart Scheduling Logic

- **Time Preferences:** Morning person? Evening availability?
- **Date Flexibility:** ASAP vs specific date
- **Appointment Duration:** Match appointment type with time slot
- **Buffer Time:** Account for travel, preparation
- **Conflict Handling:** No double-booking
- **Timezone Awareness:** Handle if applicable

5. Edge Cases & Error Handling

- ♦ **No Available Slots:**

- Clearly explain situation
- Offer waitlist or alternative dates
- Suggest calling office

- ♦ **User Changes Mind:**

- Handle gracefully mid-booking
- Allow restart without confusion

- ♦ **Ambiguous Time References:**

- "Tomorrow morning" → Clarify specific time
- "Next week" → Confirm which day
- "Around 3" → Confirm AM/PM

- ♦ **Invalid Input:**

- Non-existent dates
- Past dates
- Outside business hours

- ♦ **API Failures:**

- Calendly unavailable
- Network timeout
- Graceful degradation

Calendly API Integration

Option 1: Real Calendly API

If using real Calendly:

- Sign up for free Calendly account
- Create API token
- Set up event types
- Use these endpoints:
 - `GET /event_types` - Get appointment types
 - `GET /event_types/{uuid}/available_times` - Get availability
 - `POST /scheduled_events` - Create booking

Option 2: Mock Calendly API

If mocking, implement these endpoints:

```
# Mock Calendly Endpoints to Implement
```

```
GET /api/calendly/availability
```

```
Query params:
```

- `date: "2024-01-15"`
- `appointment_type: "consultation" | "followup" | "physical" | "special"`

Response:

```
{
  "date": "2024-01-15",
  "available_slots": [
    {"start_time": "09:00", "end_time": "09:30", "available": true},
    {"start_time": "09:30", "end_time": "10:00", "available": false},
    {"start_time": "10:00", "end_time": "10:30", "available": true},
    ...
  ]
}
```

POST /api/calendly/book

Body:

```
{
  "appointment_type": "consultation",
  "date": "2024-01-15",
  "start_time": "10:00",
  "patient": {
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "+1-555-0100"
  },
  "reason": "Annual checkup"
}
```

Response:

```
{
  "booking_id": "APPT-2024-001",
  "status": "confirmed",
  "confirmation_code": "ABC123",
  "details": {...}
}
```

Agent Capabilities Required

Conversation Example 1: Successful Booking

User: I need to see the doctor

Agent: I'd be happy to help you schedule an appointment! What's the main reason for your visit today?

User: I've been having headaches

Agent: I understand. For persistent headaches, I'd recommend a general consultation (30 minutes) where the doctor can assess your symptoms. Does that sound appropriate, or would you prefer a longer specialist consultation?

User: General consultation is fine

Agent: Perfect. When would you like to come in? Do you have a preference for morning or afternoon appointments?

User: Afternoon if possible, sometime this week

Agent: Let me check our afternoon availability this week. I have these options:

- Tomorrow (Jan 16) at 2:00 PM
- Wednesday (Jan 17) at 3:30 PM
- Thursday (Jan 18) at 1:00 PM
- Friday (Jan 19) at 2:30 PM

Which works best for you?

User: Wednesday at 3:30

Agent: Excellent! Wednesday, January 17th at 3:30 PM for a 30-minute consultation. Before I confirm, I'll need a few details:

- Your full name?
- Best phone number to reach you?
- Email address for confirmation?

[Collects information, confirms booking]

Agent: All set! Your appointment is confirmed for Wednesday, January 17th at 3:30 PM. You'll receive a confirmation email at john@example.com with all the details. The confirmation code is ABC123.

Is there anything else you'd like to know about your visit?

Conversation Example 2: FAQ During Booking

User: I want to book an appointment

Agent: I'd be happy to help! What brings you in today?

User: Actually, first - what insurance do you accept?

Agent: Great question! We accept most major insurance providers including Blue Cross Blue Shield, Aetna, Cigna, UnitedHealthcare, and Medicare. We also accept self-pay patients. Do you have insurance with one of these providers?

User: Yes, I have Blue Cross. Okay, I'd like to schedule a checkup

Agent: Perfect, we accept Blue Cross Blue Shield. Now for your checkup - would this be your first visit with us, or are you an existing patient?

[Continues with scheduling...]

Conversation Example 3: No Available Slots

User: Can I see the doctor today?

Agent: I understand you'd like to be seen today. Unfortunately, we don't have any available appointments for today. However, I have several options for tomorrow:

- 9:00 AM
- 11:30 AM
- 2:00 PM

Would any of these work? If you need urgent care, I can also provide information about our walk-in clinic hours.

User: Is there a waitlist for today?

Agent: We don't have a formal waitlist, but I'd recommend calling our office at (555) 123-4567 as they sometimes have last-minute cancellations. For planned appointments, would you like me to book you for tomorrow?

Submission Requirements

Required Files & Structure

```
appointment-scheduling-agent/  
├─ README.md  
├─ .env.example  
├─ requirements.txt  
├─ architecture_diagram.png/pdf  
├─ backend/  
│   ├── main.py  
│   ├── agent/  
│   │   ├── scheduling_agent.py  
│   │   └─ prompts.py  
│   ├── rag/  
│   │   ├── faq_rag.py  
│   │   ├── embeddings.py  
│   │   └─ vector_store.py  
│   ├── api/  
│   │   ├── chat.py  
│   │   └─ calendly_integration.py  
│   ├── tools/  
│   │   ├── availability_tool.py  
│   │   └─ booking_tool.py  
│   └─ models/  
│       └─ schemas.py  
├─ frontend/                                # If fullstack  
│   ├── package.json  
│   └─ src/  
│       ├── App.jsx  
│       └─ components/  
│           ├── ChatInterface.jsx  
│           └─ AppointmentConfirmation.jsx  
├─ data/  
│   ├── clinic_info.json                    # Provided FAQ data  
│   └─ doctor_schedule.json                # Provided schedule data  
└─ tests/  
    └─ test_agent.py
```

README.md Must Include

1. Setup Instructions

- Calendly API setup (if using real API)
- Environment variables
- Running the application

2. System Design

- Agent conversation flow
- Calendly integration approach
- RAG pipeline for FAQs
- Tool calling strategy

3. Scheduling Logic

- How available slots are determined
- Appointment type handling
- Conflict prevention

N. Testing

- Example conversations
- Edge cases covered

.env.example

```
# LLM Configuration
LLM_PROVIDER=openai
LLM_MODEL=gpt-4-turbo
OPENAI_API_KEY=your_key_here

# Calendly (if using real API)
CALENDLY_API_KEY=your_calendly_key
CALENDLY_USER_URL=https://calendly.com/your-username

# Vector Database
VECTOR_DB=chromadb
VECTOR_DB_PATH=./data/vectordb

# Clinic Configuration
CLINIC_NAME=HealthCare Plus Clinic
CLINIC_PHONE=+1-555-123-4567
TIMEZONE=America/New_York

# Application
```

BACKEND_PORT=8000

FRONTEND_PORT=3000

Architecture Diagram Must Include

- Conversation agent flow
- Calendly API integration
- RAG pipeline for FAQs
- Tool calling (availability check, booking)
- Context switching mechanism (scheduling ↔ FAQ)
- Error handling paths

Evaluation Focus Areas

1. **Conversational Quality** (Highest Priority - 30%)

- Natural, empathetic conversation
- Appropriate questions at right time
- Smooth transitions between topics
- Context awareness

2. **RAG Quality** (30%)

- Accurate FAQ retrieval
- Relevant answers
- No hallucinated information
- Seamless context switching

3. **Scheduling Intelligence** (25%)

- Understands preferences
- Smart slot recommendations
- Handles appointment types correctly
- Validates bookings

N. **Edge Case Handling** (15%)

- No slots available
- API failures
- Ambiguous inputs
- User changes mind

Time Limit

24 Hours from receipt of assessment

Submission Method

GitHub repository with all required files

Notes

- Focus on natural conversation flow
- Don't be rigid or robotic
- Handle context switches gracefully
- Be empathetic (healthcare context)
- Clear confirmations before booking