Project: AI Concierge for Small-Business AI Adoption

The concierge must answer questions about AI technologies, schedule demo calls, and keep a running to-do list. It should **judge the quality of its retrieved context (self-grading)** and **adapt to explicit user feedback (self-reflection)**.

Core Requirements – Technical Detail

1. Framework Choice (Pick One)

- **Option A**: LangGraph (LangChain ecosystem)
- **Option B**: OpenAI Agents SDK

2. Required Tools/Functions (Minimum 2)

Tool 1: `retrieve_docs` - Vector Search RAG

- Performs vector search over supplied JSON/CSV knowledge base
- **Must include source citations** in responses
- Returns: `{docs: [{content: str, source: str, metadata: dict}], query: str}`

Tool 2: `manage_tasks` - To-Do List Manager

- Add tasks: `manage_tasks(action="add", task={title: str, when: str, description: str})`
- List tasks: `manage_tasks(action="list")` returns all current tasks
- Used for demo scheduling, follow-ups, etc.
- In-memory storage for this assessment

3. Self-Grading (RAG Reliability)

After `retrieve_docs` executes:

1. Call LLM with rubric prompt:

```
Rate the following on a scale of 0-1:
```

```
Question: <question>

Retrieved Documents: <docs>

Factual Relevance: [0-1] - How relevant are the documents to
the question?

Answer Coverage: [0-1] - How completely can the question be
answered with these documents?
```

2. If **relevance < 0.6** OR **coverage < 0.6**:
   - Re-query vector DB with refined search term (once)
   - If still below threshold: respond "My knowledge base does not cover that topic."
3. Log all scores for debugging

4. Self-Reflection (Adaptive Learning)

- Accept commands: `/good_answer` or `/bad_answer`
- Store `feedback_score` in session memory:
  - `/good_answer`: +1
  - `/bad_answer`: -1
  - Apply decay factor 0.5 per new turn
- Modify next system prompt based on cumulative score:
  - If cumulative < 0: "Be more concise and cite sources explicitly."
  - If cumulative > 0: "Maintain current style, user is satisfied."

5. Memory Requirements

- Message history for current session
- Feedback score stored alongside conversation
- Session-scoped (no cross-session memory pollution)

6. FastAPI Requirements (Async)

- `/register` - User registration
- `/login` - JWT authentication
- `/concierge/chat` - Main chat endpoint
  - JWT protected
  - Async implementation
  - Rate limit: 30 requests/minute per user
  - Return streamed responses or full JSON

7. Bonus Features

- Voice Interface: `/concierge/voice`
    - Accept `audio/wav`
    - Use open-source STT (Vosk/Whisper-cpp)
    - Use open-source TTS (pyttsx3/edge-tts)

Examples & Expected Behavior

Example 1: Self-Grading (RAG Reliability)

Successful Query:

```
User: "What is few-shot learning?"

Agent → retrieve_docs("few-shot learning")

Agent → self_grade() → relevance: 0.82, coverage: 0.77

Agent → ✅ Thresholds met → Provides answer with citations

Response: "Few-shot learning is... [Source: AI Fundamentals, page
45]"
```

Out-of-Scope Query:

```
User: "How do I set up payroll software?"

Agent → retrieve_docs("payroll software")

Agent → self_grade() → relevance: 0.09, coverage: 0.05

Agent → ❌ Below threshold → Refine query

Agent → retrieve_docs("software setup payroll")

Agent → self_grade() → relevance: 0.11, coverage: 0.08

Agent → Still below threshold

Response: "I'm sorry, my knowledge base doesn't cover payroll
software."
```

Example 2: Self-Reflection & Decay

```
Turn 5: User: "/bad_answer"

        System: feedback_score = -1 (cumulative: -1)

        Next prompt prepended: "Be more concise and cite sources
explicitly."


Turn 6: User: "What is RAG?"

        Agent provides concise answer with citations

        System: Apply decay → feedback_score = -1 * 0.5 = -0.5


Turn 7: User: "/good_answer"

        System: feedback_score = -0.5 + 1 = 0.5 (cumulative: 0.5)

        System: Apply decay → feedback_score = 0.5 * 0.5 = 0.25


Turn 8: User: "Explain transformers"

        Normal prompt (positive cumulative score)

        System: Apply decay → feedback_score = 0.25 * 0.5 = 0.125
```

Example 3: Tool Usage

```
User: "Schedule an AI demo for Thursday 3 pm"

Agent → manage_tasks(action="add", task={

    title: "AI demo",

    when: "Thursday 3:00 PM",

    description: "AI technology demonstration"

})

Response: "✔️ Demo scheduled for Thursday at 3 PM."
```

```
User: "What do I have scheduled?"

Agent → manage_tasks(action="list")

Response: "Your scheduled tasks:

1. AI demo - Thursday 3:00 PM"


User: "What is vector search?"

Agent → retrieve_docs("vector search")

Agent → self_grade() → relevance: 0.87, coverage: 0.91

Response: "Vector search is a method for finding similar items...

[Source: ML Basics, Chapter 7]

[Source: Search Technologies, Section 3.2]"
```

Deliverables

1. Source Code

- Clean, modular architecture
- Type hints and docstrings
- Error handling

2. README.md containing:

- Environment setup instructions
- Self-grading rubric explanation & threshold justification
- Self-reflection algorithm details (including decay mechanism)
- API testing guide with cURL/HTTPie examples:
    - Registration and login flow
    - Happy-path Q&A with citations
    - Out-of-scope question handling
    - Feedback command demonstration

3. Docker Configuration (Mandatory):

- `Dockerfile`:
  ```
  FROM python:3.11-slim

  WORKDIR /app

  ……………rest of the file
  ```

- `docker-compose.yml`:
  - API service
  - Vector database (e.g., ChromaDB)
  - Optional STT/TTS services for voice bonus

4. Testing Suite

- Unit tests for core functions
- Integration tests for API endpoints
- Examples of test queries and expected outputs

5. Voice Bonus (if implemented):

- Script names and usage
- Sample cURL commands for voice endpoints
- Required OS packages/dependencies

6. Submission Deadline

- All deliverables must be uploaded to a **public or private GitHub repository** before **Sunday, 12:00 AM (midnight)**.
- Ensure the README is in the root directory and the repository includes all required code, tests, and Docker configuration.
- The repo should be kept **public.**
- Late submissions will not be considered for evaluation.