# Building a self-learning Guidewire interview bot

**claude.ai**/chat/a1252ae6-0ca9-4731-a14d-b2f40aeb215f

The user is asking for a comprehensive design of a Guidewire Interview Bot that:

1. Uses their existing knowledge base (100 topics with PPTs, video demos, PDFs, documentation, 500 Q&As)
2. Should be production-ready with 100% accuracy
3. Self-learning capabilities (like "trikala" - seems to be referencing a self-evolving system)
4. Eventually scale to become a trainer/assistant developer/architect
5. Built using modern tools (Cursor, Codex, Supabase, etc.)
6. They want to use ChatGPT for execution due to cost considerations
7. They want project instructions, knowledge base structure, prompts for ChatGPT

This is a serious technical request. I need to be realistic about what's achievable while providing a solid architectural design. No fluff, just technical substance.

Yes, this is absolutely doable. Here's a no-BS technical architecture for a production-grade Guidewire Interview Bot with self-learning capabilities.

## Core Architecture

### Phase 1: Foundation (2-3 weeks)

#### Knowledge Graph + RAG Pipeline

```
Components:
- Vector DB: Pinecone/Weaviate (not Supabase for vectors - it's suboptimal)
- Document Processing: LlamaIndex for chunking/indexing
- Embeddings: OpenAI ada-002 or Cohere embed-v3
- LLM: GPT-4-turbo for reasoning, Claude for code generation
- Database: Supabase PostgreSQL for structured data
```

#### Data Pipeline:

1. Parse all 100 topics into hierarchical knowledge graph
2. Extract code samples, configurations, best practices separately
3. Create semantic chunks with metadata tags (topic, difficulty, context)
4. Build relationship mappings between concepts
5. Index Q&A pairs with similarity scoring

### Phase 2: Interview Engine (2 weeks)

#### Core Modules:

python

~~~
# Simplified architecture
class GuidewireInterviewBot:
    def __init__(self):
        self.knowledge_retriever = RAGPipeline()
        self.question_generator = AdaptiveQuestionEngine()
        self.response_evaluator = ResponseScorer()
        self.learning_module = ReinforcementLearner()

    def conduct_interview(self, candidate_level):
        # Adaptive difficulty based on responses
        # Real-time concept linking
        # Performance tracking
```

**Key Features:**
- Dynamic question generation based on candidate responses
- Multi-modal understanding (code, diagrams, architecture)
- Real-time code validation against Guidewire standards
- Weakness detection and targeted follow-ups

### Phase 3: Self-Learning System (3-4 weeks)

**Learning Pipeline:**
```
User Response → Evaluation → Gap Detection → Knowledge Update → Model Fine-tuning
~~~

1. **Response Analysis:** Every answer gets vectorized and compared against expert responses
2. **Pattern Recognition:** Identify common misconceptions and knowledge gaps
3. **Knowledge Evolution:** Automatically generate new test cases from edge cases
4. **Feedback Loop:** Record successful interview patterns, update difficulty scoring

## Technical Implementation Stack

**Backend:**

- FastAPI for API endpoints
- Celery for async processing
- Redis for caching and session management
- PostgreSQL for structured data
- Vector DB for semantic search

**Frontend:**

- Next.js with TypeScript
- Real-time WebSocket for live coding
- Monaco Editor for code input
- Speech-to-text for verbal responses

**ML Pipeline:**

- LangChain for orchestration
- Custom fine-tuned models on your 500 Q&As
- RLHF implementation for continuous improvement

## ChatGPT Project Instructions

### Project Configuration

markdown

~~~
Project Name: Guidewire Interview Bot Development
Goal: Build production-ready interview system with 100% accuracy
Timeline: 8 weeks to MVP, 12 weeks to production
```

### Knowledge Base Structure for ChatGPT
```
/knowledge_base
  /guidewire_core
    - products.json (PC, BC, CC, etc.)
    - architecture_patterns.pdf
    - integration_points.json
  /interview_data
    - questions_answers.json (your 500 Q&As)
    - difficulty_mapping.csv
    - topic_relationships.json
  /code_samples
    - best_practices/
    - anti_patterns/
    - real_implementations/
```

### Core Prompts for ChatGPT

**Prompt 1: Code Generation**
```
You are building a Guidewire Interview Bot. Generate production-ready Python code for:
1. RAG pipeline that processes Guidewire documentation
2. Semantic search with 95%+ accuracy
3. Include error handling, logging, monitoring
4. Use LangChain, FastAPI, PostgreSQL
5. Implement caching strategies
Context: [paste specific module requirements]
```

**Prompt 2: Data Processing**
```
Process this Guidewire content for vector embedding:
- Extract key concepts, code samples, configurations
- Create metadata tags for: product, version, difficulty, prerequisites
- Generate semantic chunks of 512 tokens max
- Maintain context relationships
- Output as JSON with embedding-ready format
Content: [paste your PDFs/PPTs content]
```

**Prompt 3: Question Generation**
```
Based on this Guidewire topic, generate interview questions:
- 5 screening level (basic syntax, definitions)
- 5 intermediate (implementation, troubleshooting)
- 3 expert (architecture, optimization, edge cases)
~~~

```
- Include code challenges where applicable
- Provide scoring rubric for each
Topic: [specific Guidewire module]
```

## Execution Roadmap

### Week 1-2: Data Preparation

- Convert all materials to structured format
- Build knowledge graph
- Set up vector database
- Create embedding pipeline

### Week 3-4: Core Bot Logic

- Implement RAG system
- Build question selection algorithm
- Create response evaluation engine
- Set up basic UI

### Week 5-6: Intelligence Layer

- Implement adaptive questioning
- Add code execution sandbox
- Build performance analytics
- Create feedback mechanisms

### Week 7-8: Production Readiness

- Load testing (target: 100 concurrent interviews)
- Edge case handling
- Security audit
- Deployment pipeline

## Critical Success Factors

1. **Data Quality:** Your 500 Q&As need verification against latest Guidewire versions
2. **Chunking Strategy:** Guidewire configs need special parsing (XML, Gosu code)
3. **Evaluation Metrics:** Build custom scorers for Guidewire-specific patterns
4. **Sandbox Environment:** Spin up Docker containers for code validation
5. **Version Management:** Guidewire versions differ significantly - tag everything

## Scaling to Trainer/Architect Assistant

Once interview bot works:

1. Add generative capabilities for creating training materials

2. Implement code review features for Guidewire implementations
3. Build architecture recommendation engine
4. Create integration with actual Guidewire environments

## No-BS Reality Check

- **100% accuracy?** Achievable for factual Q&A, 90-95% for complex scenarios
- **Timeline:** 8 weeks minimum for solid MVP
- **Cost:** ~$500-1000/month for infrastructure at scale
- **Maintenance:** Plan 20% time for continuous updates

This is a $50-100K project if outsourced. With ChatGPT + your effort, doable in 2-3 months.

Want me to detail any specific component or create more targeted prompts for ChatGPT?