# **AI Chatbot Architecture for Incident Resolution**

## **1. Introduction**

This document outlines the architecture of an AI-powered chatbot designed to provide incident resolution metrics. The chatbot leverages knowledge base (KB) articles and incident data architecture documents to generate insights and assist users in resolving incidents efficiently.

## **2. System Overview**

The chatbot will:

* Parse and analyze KB articles and incident data.
* Extract and compute resolution metrics.
* Provide insights and recommendations for incident resolution.
* Allow users to query incident-related information.

## **3. High-Level Architecture**

### **3.1. Components**

1. **User Interface (UI)**
   1. Web-based chat interface.
   2. API for third-party integrations (Slack, Teams, ServiceNow, etc.).
2. **Natural Language Processing (NLP) Engine**
   1. Intent detection and entity extraction.
   2. Pre-trained language model (e.g., OpenAI GPT, BERT, or Llama).
3. **Document Processing Engine**
   1. Parses KB articles and incident data.
   2. Uses embeddings (e.g., OpenAI Embeddings, Sentence-BERT) for retrieval.
4. **Knowledge Base (KB) & Incident Data Storage**
   1. Stores historical incident reports and resolution steps.
   2. Supports structured and unstructured data.
5. **Analytics & Metrics Engine**
   1. Computes key incident metrics:
      1. Mean Time to Resolution (MTTR)
      2. Mean Time Between Failures (MTBF)
      3. Resolution Success Rate
   2. Generates trends and predictive insights.
6. **AI-powered Recommendation Engine**
   1. Suggests resolution steps based on historical data.
   2. Provides alternative solutions if an incident is unresolved.
7. **Security & Authentication**
   1. Role-based access control (RBAC).
   2. Secure data handling and encryption.
8. **Logging & Monitoring**
   1. Tracks chatbot interactions.
   2. Monitors query response accuracy.

## **4. Workflow**

### **4.1 User Query Processing**

1. User submits a query via the chat interface.
2. The NLP engine processes the query to identify intent and extract entities.
3. The chatbot searches relevant KB articles and historical incidents.
4. The recommendation engine suggests a resolution based on past data.
5. The analytics engine provides incident resolution metrics.
6. The chatbot delivers the response and insights to the user.

### **4.2 Data Processing & Learning**

1. Periodic ingestion of new KB articles and incident data.
2. Preprocessing and vectorization of document content.
3. Continuous learning from resolved incidents and user feedback.
4. Model fine-tuning to improve response accuracy.

## **5. Technology Stack**

|  |  |
| --- | --- |
| **Component** | **Technology Options** |
| **Frontend** | React, Angular, Vue.js |
| **Backend** | Python (FastAPI, Flask) |
| **NLP Engine** | OpenAI API, Hugging Face, Rasa NLP |
| **Database** | PostgreSQL, MongoDB |
| **Search & Retrieval** | Elasticsearch |
|  |  |
| **Security** | OAuth |

## **6. Deployment & Scalability**

* **Deployment:** Docker + Kubernetes for microservices.
* **Hosting:** AWS/GCP/Azure with auto-scaling.
* **CI/CD:** GitHub Actions or Jenkins for automated deployment.
* **Monitoring:** Prometheus + Grafana for performance tracking.

## **7. Conclusion**

This chatbot architecture ensures an intelligent, scalable, and efficient system for retrieving incident resolution metrics. It integrates NLP, analytics, and security features to provide an enhanced user experience.

**Next Steps:**

* Fine-tune the NLP model with domain-specific data.
* Deploy the chatbot in a controlled environment for testing.