AI-Powered Support Chatbot

1. Introduction

1.1 Purpose of This Document

This document explains the design, architecture, and working of an AI-powered support chatbot system. It is written for beginners with no prior SysOps, DevOps, or enterprise system experience.

The goal is to clearly explain:

- What the chatbot is supposed to do
- How it works internally
- How different components interact
- How the system behaves in different situations (cases)

This document can be used by:

- Interns and new engineers
- SysOps and support teams
- Developers building or maintaining the system

1.2 Problem Statement

In large organizations, support teams often face the following problems:

- Many repetitive questions from users
- Slow first response times
- Manual effort required to create support tickets
- No structured learning from past issues

Without automation, support teams spend time answering the same basic questions again and again.

1.3 Vision and Goals

The vision of this chatbot system is to act as a Level-1 Support Agent that can:

- Instantly answer known and frequently asked questions
- Escalate unknown questions in a structured way
- Never leave the user without a response
- Reduce manual workload on support teams

The key goals are:

- Fast and accurate responses
- Controlled escalation to human support
- Closed support loop (no unanswered queries)
- Easy maintenance through an admin panel

1.4 Assumptions & Constraints

The following assumptions and constraints are made while designing and documenting the chatbot system. These help define clear boundaries and avoid over-engineering during Phase-1.

1.4.1 Assumptions

- The chatbot is intended to function as a Level-1 support assistant, not a replacement for human support.
- Users interacting with the chatbot are publishers or customers, not anonymous internet users.
- User authentication (if required) is handled outside the chatbot system.
- The chatbot does not process or store personally identifiable information (PII).
- Salesforce is available and accessible through secure APIs.
- The Knowledge Base is curated and maintained manually by admins.
- Phase-1 chatbot is not conversational (no multi-turn context memory).

1.4.2 Constraints

- The chatbot must provide **deterministic and auditable responses**.
- AI responses must come **only from approved knowledge**, not free-form generation.
- Escalation must always succeed even if AI services fail.
- System design prioritizes **safety, simplicity, and observability** over intelligence.

---

2. High-Level System Overview

2.1 What the Chatbot Does (Simple Explanation)

The chatbot is a web-based support assistant.

From a user's point of view:

- The user types a question into a chat box
- The chatbot tries to find the best possible answer
- If the chatbot knows the answer, it replies instantly
- If it does not know the answer, it creates a support ticket and shares the reference number

From a system point of view:

- The chatbot checks an internal knowledge database
- It uses AI to understand the meaning of the question
- It decides whether the answer is reliable enough to show

2.2 Two Execution Paths (Key Concept)

The chatbot system has one architecture but two execution paths.

These are not two different chatbots. They are two different outcomes of the same system.

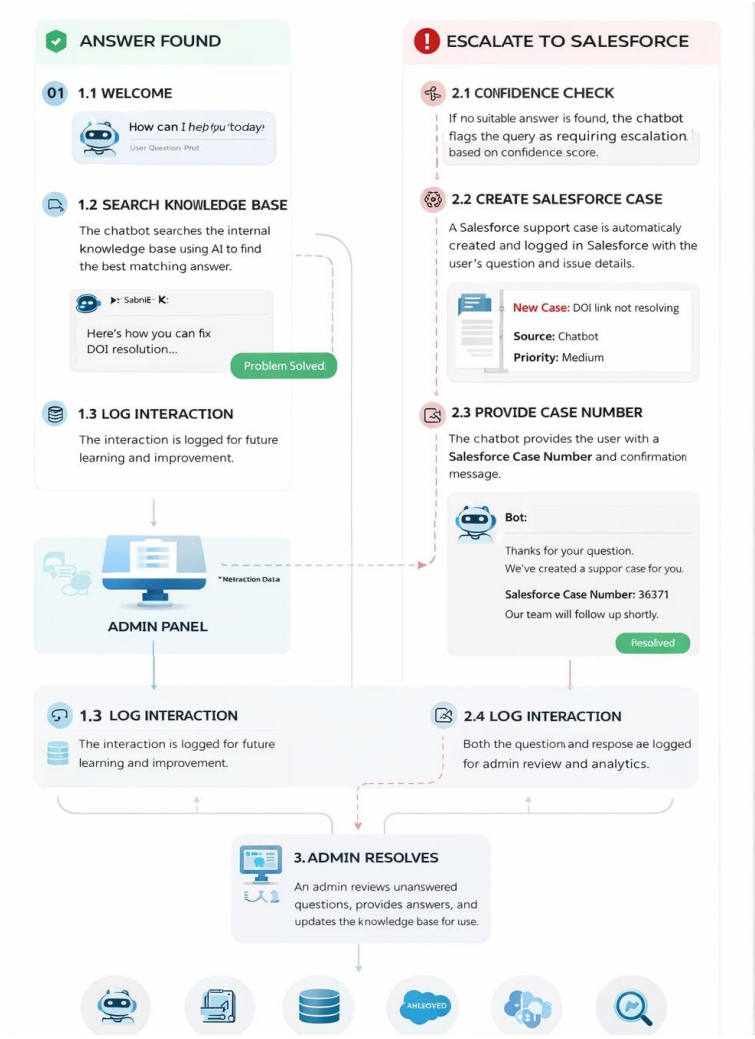
Case 1: Answer Found

- The system finds a confident answer in the knowledge base
- The chatbot replies immediately
- The interaction is logged

Case 2: Answer Not Found

- The system does not find a confident answer
- A support case is created automatically
- The case number is returned to the user
- The question is stored for future improvement

The figure below shows the Conversation flow when a user enters a query.



2.3 Scope Definition & Guardrails

This section defines what the chatbot will and will not do in Phase-1.

2.3.1 In Scope (Phase-1)

- Text-based web chatbot
- Knowledge-base-driven answers
- Confidence-based escalation
- Salesforce ticket creation
- Admin panel for managing Q&A
- Logging and monitoring

2.3.2 Out of Scope (Phase-1)

- Conversational multi-turn dialogue
- Voice or messaging platform integrations
- Automatic AI learning without human approval
- Personalized responses based on user identity
- Advanced analytics or recommendations

Guardrail principle:
If the chatbot is unsure, it escalates — it never guesses.

---

3. System Components Overview

This section lists all major components of the chatbot system and explains them at a beginner level.

3.1 Chat UI

The Chat UI is the user-facing web interface where users type questions and read responses.

Purpose:

- Accept user input
- Display chatbot responses

3.2 Backend API Layer

The Backend API acts as the central controller of the system.

Purpose:

- Receive requests from the Chat UI
- Coordinate between AI, databases, and external systems
- Return responses back to the UI

3.3 Query Processing / NLP Layer

This layer prepares the user’s question for AI processing.

Purpose:

- Clean and normalize text
- Understand user intent
- Handle variations in phrasing

3.4 AI Matching Engine

This component finds the best possible answer from the knowledge base.

Purpose:

- Match user questions with stored answers
- Use semantic understanding instead of keywords

3.5 Confidence Scoring Logic

This logic decides whether the AI is confident enough to answer.

Purpose:

- Prevent incorrect or low-quality answers
- Decide between Case 1 and Case 2

3.6 Knowledge Base Database

This database stores approved questions and answers.

Purpose:

- Act as the single source of truth
- Prevent AI hallucination

3.7 Chat Logs Database

This database stores conversation history.

Purpose:

- Auditing and debugging
- Analytics and improvement

3.8 Admin Panel

The Admin Panel is a web interface for internal teams.

Purpose:

- Add and edit questions and answers
- Review escalated issues
- Improve chatbot knowledge over time

3.9 Authentication and Roles

This component controls access to admin features.

Purpose:

- Secure the system
- Restrict sensitive actions

3.10 Salesforce Integration Service

This service communicates with Salesforce to create support cases.

Purpose:

- Ensure no user query is left unresolved
- Provide a reference number to users

3.11 Logging and Monitoring

This component tracks system behavior and errors.

Purpose:

- Detect failures
- Monitor performance
- Support SysOps troubleshooting

3.12 Data Model Overview

This component tracks system behavior and errors.

Purpose:

- Detect failures
- Monitor performance
- Support SysOps troubleshooting

3A: UX & PRODUCT DESIGN DOCUMENTATION

This section describes how different users interact with the chatbot system.

3A.1 User Personas

Persona 1: Publisher / Customer User

- Goal: Get quick answers to common issues
- Pain Points:
  - Waiting for support replies
  - Repeating the same questions
- Expectations:
  - Fast responses
  - Clear next steps
  - Case reference when escalated

Persona 2: Admin / Support Team

- Goal: Reduce repetitive tickets
- Pain Points:
  - Manual triaging
  - Knowledge scattered across teams
- Expectations:
  - Central control
  - Clear audit trail
  - Easy KB updates

3A.2 User Scenarios

Scenario 1: Known Issue

- User asks a common question
- Chatbot answers instantly
- Interaction logged

Scenario 2: Unknown Issue

- User asks a new or unclear question
- Chatbot escalates to Salesforce
- User receives case number

Scenario 3: Admin Improvement

- Admin reviews escalated case
- Converts resolution into KB entry
- Future users get automated answers

3A.3 User Stories (Agile Format)

- As a **publisher**, I want instant answers so that I don't raise unnecessary tickets.
- As a **publisher**, I want a case number so that I can track unresolved issues.
- As an **admin**, I want to review unanswered questions so that the chatbot improves.
- As a **SysOps engineer**, I want logs and metrics so that I can monitor system health.

3B: MINIMUM VIABLE PRODUCT DEFINITION (MVP)

The MVP focuses on delivering **core value with minimal complexity**.

3B.1 MVP Features

- Text-based chat interface
- Knowledge base lookup
- Confidence-based decision
- Salesforce escalation
- Admin panel for KB management
- Logging and basic monitoring

3B.2 Excluded from MVP

- Conversational context memory
- Voice interfaces
- Automated KB generation
- Advanced analytics dashboards

---

4. CASE 1 - Answer Found

4.1 Purpose, Vision, Goals (Case 1)

Purpose
Handle user questions that already have a known, approved answer.

Vision
Act like a fast, reliable Level-1 support agent that resolves common issues instantly.

Goals

- Zero human intervention
- Sub-second to few-second response time
- No incorrect or hallucinated answers
- Every interaction logged for auditing

4.2 Components Involved (Case 1 Only)

Only the components that participate when an answer **is found**:

1. Chat UI
2. Backend API Layer
3. Query Processing / NLP Layer
4. AI Matching Engine
5. Confidence Scoring Logic
6. Knowledge Base Database
7. Chat Logs Database

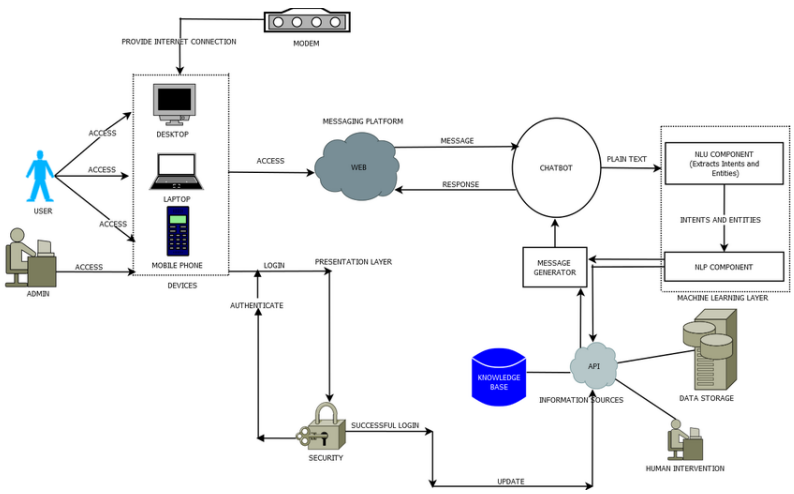
❗ Salesforce and Admin Panel are **NOT involved** in Case 1 runtime.

4.3 End-to-End Workflow Architecture (Case 1)

Step-by-step flow (very low level)

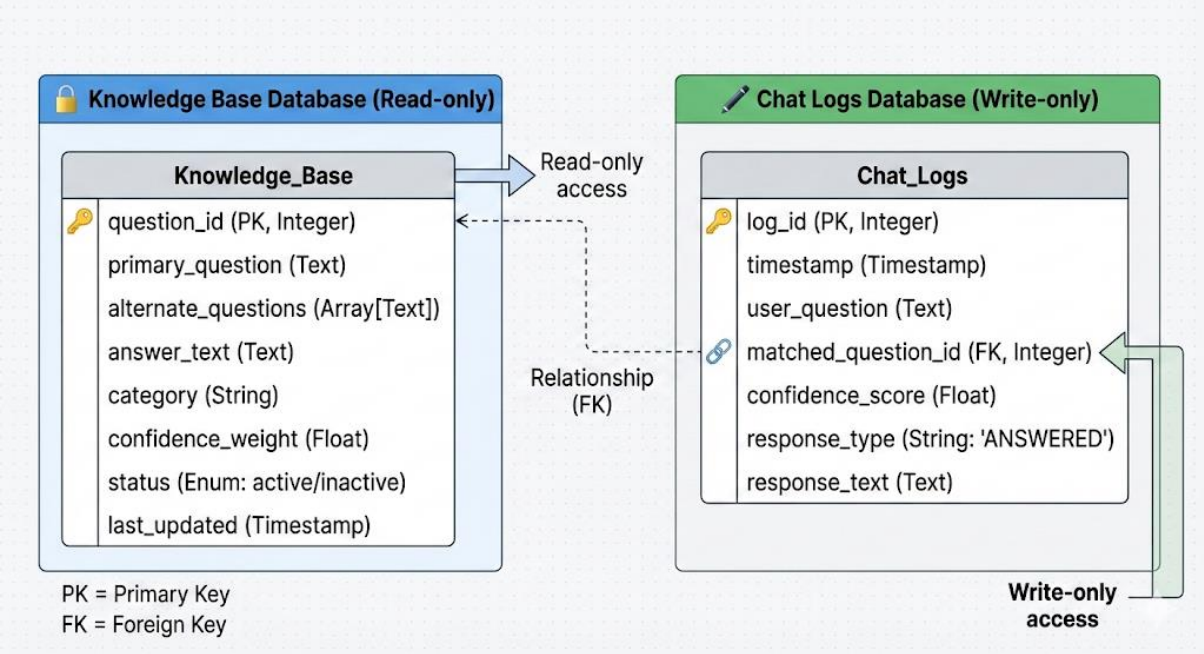
1. User types a question in the Chat UI
2. Chat UI sends the text to the Backend API
3. Backend sends text to Query Processing
4. Cleaned query is passed to AI Matching Engine
5. AI searches the Knowledge Base
6. Confidence Score is calculated
7. Score ≥ threshold → answer approved
8. Answer returned to user
9. Interaction stored in Chat Logs DB

Workflow Architecture Diagram — Case 1



4.4 Database Design (Case 1)

Databases involved



4.5 Component-wise Breakdown (Case 1)

→ *Purpose* → *Workflow* → *Database* → *Tech Stack*
for **every single component**.

4.5.1 Chat UI

Purpose
Provide a simple, conversational interface for users.

Workflow

- Collect text input
- Send request to backend
- Display response

Database Interaction
❌ None (never talks directly to DB)

Tech Stack

- React / Vue

- WebSocket or REST
- HTML/CSS

4.5.2 Backend API Layer

Purpose
Central brain of the system.

Workflow

- Accept request from UI
- Orchestrate AI + DB calls
- Return final response

Database Interaction

- Reads Knowledge Base
- Writes Chat Logs

Tech Stack

- Node.js (Express) or Python (FastAPI)
- REST APIs
- JSON payloads

4.5.3 Query Processing / NLP Layer

Purpose
Normalize messy human input.

Workflow

- Lowercasing
- Removing noise
- Tokenization
- Intent normalization

Database Interaction
✕ None

Tech Stack

- Python
- spaCy / NLTK
- Lightweight preprocessing (not LLM)

4.5.4 AI Matching Engine

Purpose
Find the *best semantic match*.

Workflow

- Convert query to embedding
- Compare against stored embeddings
- Rank results

Database Interaction

- Reads embeddings from Knowledge Base

Tech Stack

- OpenAI embeddings / Azure OpenAI
- Vector DB (FAISS / Pinecone)

4.5.5 Confidence Scoring Logic

Purpose
Decide whether the system is “sure enough”.

Workflow

- Combine similarity score
- Apply threshold rule
- Output decision (Answer / Escalate)

Database Interaction

- Reads confidence weights
- Writes final decision to logs

Tech Stack

- Backend logic (Python/JS)
- Simple rules (no ML needed)

4.5.6 Knowledge Base Database

Purpose
Source of truth.

Workflow

- Queried only
- Never modified in Case 1

Tech Stack

- PostgreSQL / MongoDB
- Optional vector DB extension

4.5.7 Chat Logs Database

Purpose
Observability + audit.

Workflow

- Store every interaction
- Used later for analytics

Tech Stack

- PostgreSQL / MongoDB
- Indexed for time + category

4.6 Tech Stack Summary (Case 1)

Layer	Technology
Frontend	React / Vue
Backend	Node.js / FastAPI
AI	Embeddings + Vector Search
DB	PostgreSQL / MongoDB
Infra	Cloud VM / Containers

5 CASE 2 — Answer Not Found (Escalation)

5.1 Purpose, Vision, Goals (Case 2)

Purpose

Ensure that **no user query is left unresolved**, even when the chatbot does not know the answer.

Vision

The chatbot should behave like a responsible support agent:

"If I don't know the answer, I will immediately involve the right human team and give the user a reference."

Goals

- Never respond with "I don't know"
- Automatically create a support ticket
- Provide a **case number** to the user
- Capture data for future learning
- Close the loop via admin intervention

This is what your manager meant by:

"The loop should be closed."

5.2 Components Involved (Case 2)

Case 2 uses **all Case 1 components**, plus escalation-specific ones.

Components Active in Case 2

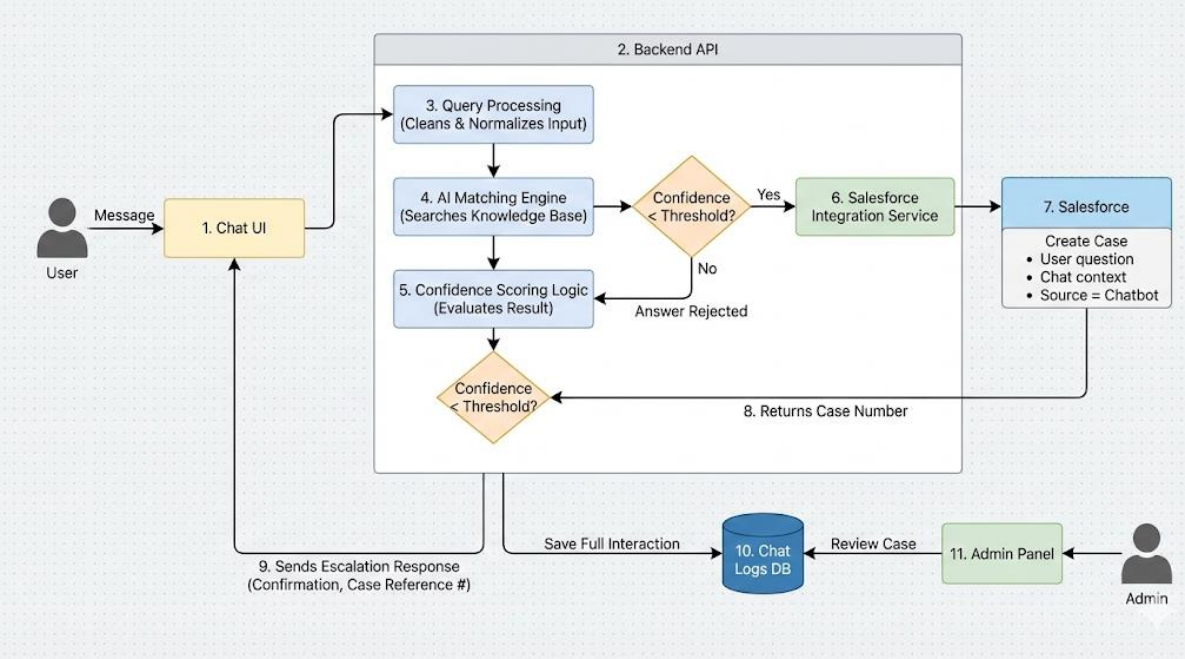
1. Chat UI
2. Backend API Layer
3. Query Processing / NLP
4. AI Matching Engine
5. Confidence Scoring Logic
6. Knowledge Base Database
7. Chat Logs Database
8. **Salesforce Integration Service**
9. **Admin Panel**
10. Authentication & Roles
11. Logging & Monitoring

Key difference from Case 1:
The flow **leaves the chatbot system** and enters **Salesforce**, then later re-enters via Admin Panel.

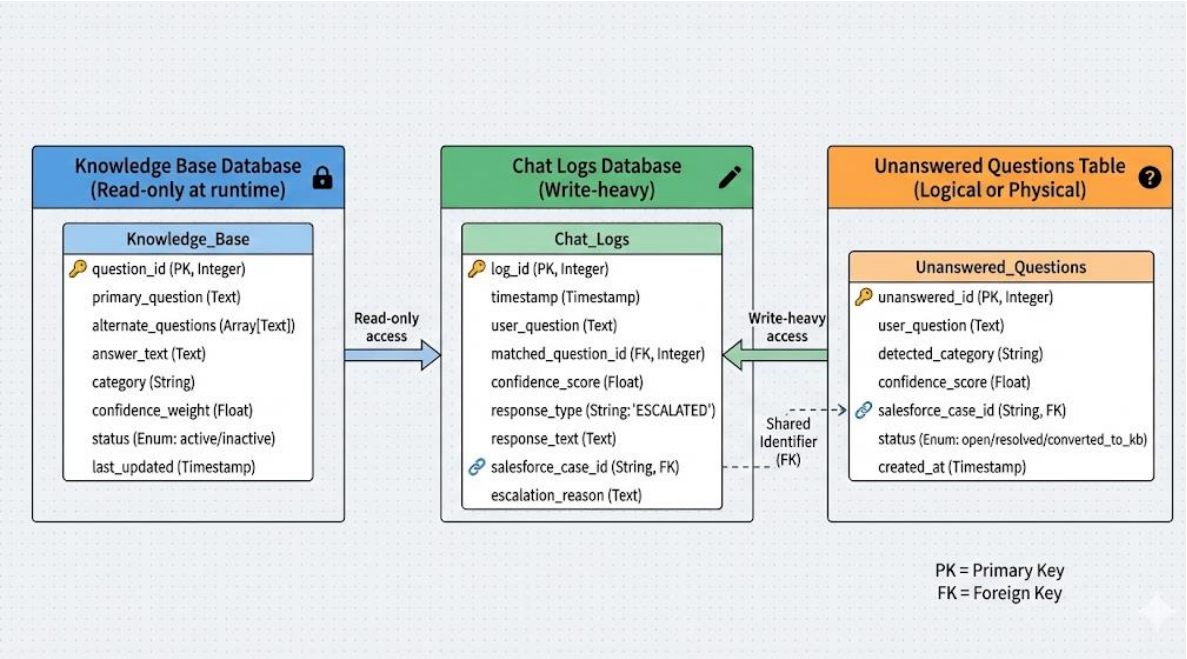
5.3 End-to-End Workflow Architecture (Case 2)

Step-by-step flow (very low-level)

1. User submits a question in Chat UI
2. Chat UI sends question to Backend API
3. Query Processing cleans and normalizes input
4. AI Matching Engine searches Knowledge Base
5. Confidence Scoring Logic evaluates result
6. **Confidence < threshold → Answer rejected**
7. Backend triggers Salesforce Integration Service
8. Salesforce Case is created with:
    - User question
    - Chat context
    - Source = Chatbot
9. Salesforce returns **Case Number**
10. Backend sends escalation response to user:
    - Confirmation message
    - Case reference number
11. Full interaction is saved in Chat Logs DB
12. Admin later reviews the case via Admin Panel



5.4 Database Design (Case 2)



5.5 Component-wise Breakdown (Case 2)

We again follow your required structure:

Purpose → Workflow → Database → Tech Stack

5.5.1 Confidence Scoring Logic (Case 2 Behavior)

Purpose
Prevent low-quality or risky answers.

Workflow

- Compare similarity score to threshold
- If below threshold → trigger escalation flag
- Pass escalation reason to backend

Database Interaction

- Writes decision + score to Chat Logs

Tech Stack

- Backend logic (Node.js / Python)
- Configurable threshold value

5.5.2 Salesforce Integration Service

Purpose
Automatically raise a support ticket when chatbot fails.

Workflow

1. Receive escalation request
2. Format Salesforce payload
3. Call Salesforce Case API
4. Receive Case ID
5. Return Case ID to backend

Database Interaction

- Stores Salesforce Case ID in Chat Logs
- Links to unanswered questions

Tech Stack

- Salesforce REST API
- OAuth authentication
- Backend service (Node.js / Python)

5.5.3 Chat UI (Case 2 Response)

Purpose
Reassure the user and provide traceability.

Workflow

- Display confirmation message
- Show Salesforce Case Number
- End conversation cleanly

Database Interaction
❌ None

Tech Stack

- Same as Case 1 (React / Vue)

5.5.4 Admin Panel (Critical in Case 2)

Purpose
Human-in-the-loop system to improve chatbot intelligence.

Workflow

1. Admin logs in
2. Views escalated questions
3. Reviews Salesforce case status
4. Writes an approved answer
5. Converts unresolved question → Knowledge Base entry

Database Interaction

- Reads unanswered questions
- Writes new Q&A to Knowledge Base
- Updates escalation status

Tech Stack

- Web UI (React)
- Secure backend API
- Role-based access

5.5.5 Authentication & Roles

Purpose
Protect sensitive operations.

Workflow

- Admin login
- Role validation
- Permission enforcement

Database Interaction

- Reads user roles
- Logs admin actions

Tech Stack

- JWT / Session-based auth
- Role-based access control

5.6 Tech Stack Summary (Case 2)

Layer	Technology
Frontend	React / Vue
Backend	Node.js / FastAPI
AI	Embeddings + Vector Search
DB	PostgreSQL / MongoDB
Ticketing	Salesforce API
Auth	JWT / RBAC
Infra	Cloud VM / Containers

6 ADMIN PANEL — DESIGN & WORKFLOWS

The Admin Panel is what turns this chatbot from a "demo bot" into a real, maintainable enterprise system**.**

Without this, the chatbot would never improve.

6.1 Purpose of the Admin Panel (Plain English)

Why the Admin Panel exists

The chatbot does NOT learn automatically.

Instead:

- Humans review failures
- Humans approve answers
- Humans control what the chatbot is allowed to say

The Admin Panel is the human control layer.

What problems it solves

- Chatbot gives wrong answers → ✕ avoided
- Unanswered questions pile up → ✕ avoided
- Knowledge lives only in people's heads → ✕ avoided

6.2 Who Uses the Admin Panel (User Roles)

This matters in SysOps.

Role 1: Admin (SysOps / Platform Team)

Can:

- Add / edit / delete Q&A
- Convert Salesforce cases into answers
- Enable / disable chatbot responses
- View analytics

Role 2: Support Editor (Optional / Phase-2)

Can:

- Review unanswered questions
- Suggest answers
- Cannot publish without approval

This is why authentication & roles exist.

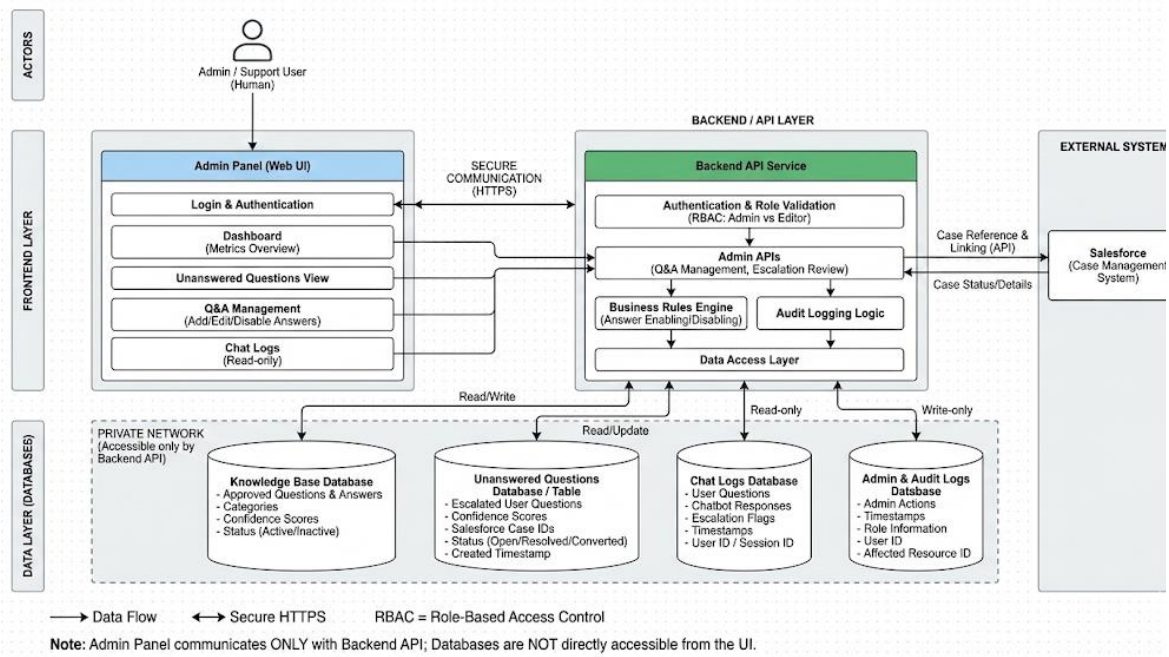
6.3 Admin Panel – High-Level Architecture

The Admin Panel is not standalone.
It connects to the same backend as the chatbot.

Key rule (SysOps best practice):

Admin UI never talks directly to databases

Admin Panel Architecture Diagram



Flow summary:

1. Admin opens Admin UI
2. Logs in
3. UI calls Backend APIs
4. Backend validates role
5. Backend reads/writes databases
6. Changes affect chatbot behavior immediately

6.4 Core Admin Workflows (Very Important)

We'll now break down each admin workflow, one by one.

6.4.1 Workflow 1 - View Unanswered / Escalated Questions

Purpose

Identify where the chatbot failed.

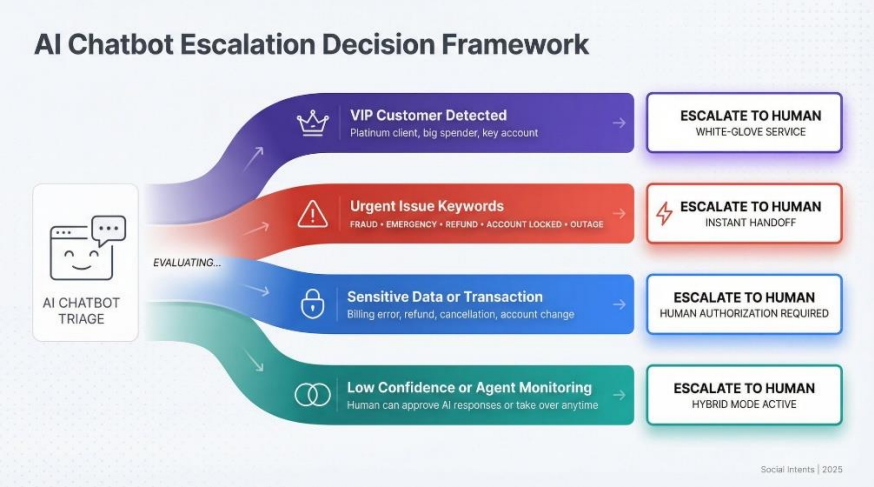
Trigger

A Case 2 escalation happened.

Step-by-step workflow

1. Admin logs in
2. Opens “Unanswered Questions” dashboard
3. Sees list of:
    - User questions
    - Confidence scores
    - Salesforce Case IDs
    - Status (Open / Resolved)

Workflow Diagram — View Escalations



Database interaction

Reads from:

- Chat Logs DB
- Unanswered Questions table
- Salesforce Case reference (read-only)

6.4.2 Workflow 2 — Review Salesforce Case & Resolution

Purpose

Understand how the issue was solved by humans.

Step-by-step

1. Admin clicks an escalated question
2. Views:
    - Original user question
    - Full chat context
    - Salesforce Case ID
3. Checks case status (Open / Closed)
4. Reads final human resolution

Why this matters

This step ensures:

- Only correct answers enter the Knowledge Base
- No assumptions or hallucinations

6.4.3 Workflow 3 — Convert Resolved Case → Knowledge Base Entry

This is the most important admin workflow.

This is how the chatbot gets smarter over time.

Step-by-step conversion process

1. Admin clicks “Convert to Knowledge Base”
2. Admin fills:
    - Primary Question
    - Alternate Questions
    - Approved Answer
    - Category
    - Initial Confidence Weight
3. Admin saves entry
4. Entry becomes active
5. Future users now get Case 1 answers

Database interaction

Writes to:

- Knowledge Base DB

Updates:

- Unanswered Questions status → converted_to_kb

6.4.4 Workflow 4 — Edit / Disable Existing Answers

Purpose

Prevent outdated or risky answers.

Step-by-step

1. Admin searches existing Q&A
2. Selects an entry
3. Can:
    - Edit answer
    - Update confidence score
    - Disable entry

Why disabling matters

If:

- A process changes
- A service is deprecated

Admins can **instantly stop** the chatbot from answering incorrectly.

6.5 Admin Panel Screens (Logical View)

You don’t need UI perfection, but **clarity matters**.

Core Screens

1. Login Screen
2. Dashboard Overview
3. Unanswered Questions
4. Q&A Management
5. Add / Edit Q&A
6. Chat Logs (Read-only)

6.6 Admin Panel – Database Design

Tables involved

1. Admin Users

- admin_id
- email
- role
- last_login

2. Knowledge Base

(same as earlier, but writable here)

3. Unanswered Questions

- escalation tracking

4. Audit Logs

- admin_id
- action_type
- timestamp
- affected_record

6.7 Admin Panel – Tech Stack

Layer	Technology
Frontend	React / Vue
Backend	Node.js / FastAPI
Auth	JWT / RBAC
DB	PostgreSQL / MongoDB
API	REST
Security	HTTPS, role checks

6.8 Why SysOps Cares About the Admin Panel

From a SysOps point of view:

- No uncontrolled AI answers
- Clear audit trail
- Human approvals
- Fast rollback (disable answers)
- Continuous improvement

This turns the chatbot into a **safe system**, not a risky one.

7 Security, Logging & SysOps Considerations

This section explains **how the system is kept safe, observable, and operable** in a real corporate environment.

SysOps is not just about "making things work"
It's about making sure things don't break silently, don't leak data, and can be fixed quickly

---

7.1 Security Considerations

Security applies at every layer of the chatbot system.

We'll break this down clearly.

7.1.1 User-Facing Security (Chat UI)

What needs to be protected

- User input
- Chat responses
- Session data

Security Measures

1. HTTPS only

- All communication between browser and backend must be encrypted
- Prevents data sniffing and man-in-the-middle attacks

2. Input sanitization

- Prevents malicious input (scripts, injections)
- Especially important for free-text chat

3. Rate limiting

- Prevents abuse or spam
- Stops automated bots from flooding the system

Beginner takeaway:
Never trust user input. Ever.

7.1.2 Backend API Security

The Backend API is the **most critical attack surface**.

Security Measures

1. API authentication

- Only trusted UIs can call backend APIs
- Prevents unauthorized access

2. Authorization checks

- Admin-only APIs must reject normal users
- Role validation happens on **every request**

3. Secure configuration

- No secrets in code
- Environment variables for keys and tokens

7.1.3 Admin Panel Security

Admin Panel access is **high risk** because it can change chatbot behavior.

Security Measures

1. Authentication

- Login required
- Strong password policy

- Session expiration

2. Role-Based Access Control (RBAC)

- Admin vs Editor roles
- Principle of least privilege

3. Audit logging

- Every admin action is logged
- Who changed what and when

SysOps rule:
If it's not logged, it didn't happen.

7.1.4 Data Security (Databases)

What data is sensitive

- Chat logs
- User questions
- Admin actions
- Salesforce case references

Security Measures

1. Database access restrictions

- Databases accessible only from backend
- No public exposure

2. Encryption

- Encryption at rest
- Encryption in transit

3. Limited data retention

- Store only what is necessary
- Avoid personal or sensitive data in chat logs

7.1.5 External Integration Security (Salesforce & AI APIs)

Risks

- API key leaks
- Unauthorized case creation
- Data exposure

Security Measures

1. Secure API key storage

- Environment variables
- Secret managers (recommended)

2. Scoped permissions

- Salesforce API access limited to case creation only

3. Timeout & retry controls

- Prevent backend from hanging if Salesforce is slow

7.2 Logging Strategy (Very Important in SysOps)

Logging is how SysOps understands **what happened after it happens**.

7.2.1 What Should Be Logged

Application Logs

- Incoming requests
- API responses
- Errors and exceptions

Chat Logs

- User question
- Response type (Answered / Escalated)
- Confidence score
- Timestamp

Admin Action Logs

- Login attempts
- Q&A edits
- Enable/disable actions

7.2.2 What Should NOT Be Logged

- Passwords
- API keys
- Sensitive personal data

Beginner rule:
If you wouldn't want it leaked, don't log it.

7.3 Monitoring & Observability

Monitoring tells you something is wrong before users complain.

7.3.1 Key Metrics to Monitor

System Metrics

- API response time
- Error rates
- Request volume

Chatbot Metrics

- % questions answered
- % escalated
- Confidence score trends

Integration Metrics

- Salesforce API success/failure
- AI API latency

Monitoring & Alerting Architecture Diagram

7.3.2 Alerts (When SysOps Should Be Notified)

Examples:

- Salesforce API failures
- Spike in escalations
- Backend error rate > threshold
- Slow response times

Alerts ensure:

- Faster incident response
- Reduced downtime

7.4 Failure Handling & Resilience

Failures will happen. Good systems plan for them.

7.4.1 Graceful Degradation

If:

- AI service fails
- Knowledge DB is unavailable

Then:

- Skip AI
- Directly escalate to Salesforce
- Inform user politely

Better to escalate than give a wrong answer.

7.4.2 Retry & Timeout Strategy

- Salesforce calls have retry limits
- AI calls have timeouts
- Prevents cascading failures

7.5 SysOps Best Practices Applied

This system follows key SysOps principles:

- Separation of concerns
- Least privilege access
- Observability by default
- Human-in-the-loop controls
- Fail-safe escalation

This is why the design is **s**afe to operate in production.

---

8. Non-Functional Requirements

Non-functional requirements describe how well the system should work, not what it does. These are critical in SysOps because they affect reliability, performance, and maintainability.

8.1 Performance

- Chatbot response time should be within a few seconds for normal queries.
- Knowledge Base lookup should be fast and optimized.
- Salesforce case creation should complete quickly and not block the user interface.

8.2 Scalability

- The system should support multiple users chatting at the same time.
- Backend APIs should be stateless so they can be scaled horizontally.
- Databases should be indexed for search and time-based queries.

8.3 Reliability & Availability

- The chatbot should be available most of the time.
- If one component fails (AI, database, Salesforce), the system should degrade gracefully instead of crashing.

- Escalation should always work even if AI fails.

8.4 Maintainability

- Clear separation between UI, backend, AI logic, and data storage.
- Easy to add or modify Q&A entries via Admin Panel.
- Configurable confidence thresholds without code changes.

8.5 Security

- Secure communication using HTTPS.
- Proper authentication and authorization for admin users.
- Sensitive credentials stored securely.

8A: TESTING STRATEGY & QUALITY ASSURANCE

This section defines how the chatbot will be validated.

8A.1 Test Plan

Testing will ensure:

- Correct responses for known questions
- Proper escalation for unknown questions
- Admin actions affect chatbot behavior
- Failures are handled gracefully

8A.2 Types of Testing

- **Functional Testing**
  - Case 1 flow
  - Case 2 escalation
- **API Testing**
  - Backend endpoints
  - Salesforce integration
- **Admin Panel Testing**
  - Add/edit/disable Q&A
- **Failure Testing**
  - AI unavailable
  - Salesforce unavailable

8A.3 Sample Test Cases

Test Case	Expected Result
Confidence ≥ threshold	Answer returned
Confidence < threshold	Salesforce case created
Admin disables answer	Chatbot stops responding
Salesforce down	Graceful error + retry

---

9. Project Roadmap & SDLC Execution

This project follows an Agile SDLC model.

9.1 Agile Model Overview

- Short iterations
- Continuous feedback
- Incremental improvements

9.2 Sprint Plan (Phase-1)

Sprint 1

- Chat UI
- Backend API
- KB structure

Sprint 2

- AI matching
- Confidence logic
- Logging

Sprint 3

- Salesforce integration
- Admin panel
- Testing & hardening

9.3 Roadmaps

Strategy Roadmap

- Reduce L1 support load
- Improve response time

Technology Roadmap

- Phase-1: Web chatbot
- Phase-2: Conversational memory
- Phase-3: Analytics & AI suggestions

Release Roadmap

- MVP
- Admin controls
- Observability
- Enhancements

9.A Risks, Scalability Concerns & Mitigation Plan

Risk	Impact	Mitigation
Traffic spikes	Slow responses	Auto-scaling backend
AI latency	Poor UX	Timeouts + escalation
Salesforce downtime	Escalation failure	Retry + alert
KB growth	Slow search	Indexed + vector DB
Cost overruns	Budget issues	Rate limits & quotas

9.B Metrics & Success Criteria:

- ≥ 70% auto-resolution rate
- < 3s average response time
- < 30% escalation rate
- Reduced manual ticket volume
- Faster admin resolution turnaround

---

10. Future Enhancements (Phase-2)

The following improvements are out of scope for Phase-1 but can be added later.

10.1 Multi-Language Support

- Allow users to ask questions in different languages.
- Translate responses automatically.

10.2 Smarter AI Suggestions

- Auto-suggest draft answers to admins based on past cases.
- Reduce manual effort for knowledge base updates.

10.3 Publisher-Specific Responses

- Customize answers based on publisher or customer type.
- Improve relevance and accuracy.

10.4 Advanced Analytics

- Track trending issues.
- Identify gaps in documentation.
- Measure chatbot impact on support workload.

10.5 Conversational Chatbot

- Remember context across turns.
- Understand follow-up questions
- Change behaviour based on previous answers.

---

11. Conclusion

This document described the complete design and architecture of an AI-powered support chatbot system from a beginner-friendly perspective.

Key highlights:

- The system uses a single architecture with two execution paths.
- Known questions are answered instantly (Case 1).
- Unknown questions are escalated safely to Salesforce (Case 2).
- An Admin Panel ensures human control and continuous improvement.
- Security, logging, and monitoring make the system safe to operate.

This design follows core SysOps principles:

- Reliability over guesswork
- Controlled automation
- Observability and accountability
- Human-in-the-loop decision making

The result is a chatbot that is not only intelligent, but also safe, maintainable, and production-ready.