Hack-Nation's

4th Global AI Hackathon & Venture Incubation Program

in collaboration with MIT Sloan AI Club & MIT Club of Northern California

Empowering the next generation of top AI builders & entrepreneurs.

Designing a Self-Learning AI System for Support that Builds Trust

Sponsored Track by RealPage

1. Goals and Motivation

Customer support is one of the most consequential trust interfaces between organizations and the world. Every interaction is a moment where trust is either reinforced or eroded - through accuracy, consistency, tone, compliance, and accountability.

Yet most enterprises still rely on:

- static knowledge bases
- manual quality assurance
- fragmented documentation
- and slow feedback loops.

As organizations scale globally and products evolve rapidly, these approaches break down. Not because of bad intent, but because knowledge, policy, and judgment cannot move at the speed required to sustain trust at scale.

This challenge asks you to build something fundamentally different:

A self-learning AI layer that continuously reads customer interactions, extracts operational knowledge, updates institutional memory, and guides both humans and AI agents in real time.

The ambition is not to automate tickets.

It is to build the intelligence layer that allows organizations to learn from customer interactions and systematically embed trust into decision-making. One that learns from every interaction, enforces traceability through feedback loops, and improves the accuracy, consistency, and safety of decisions over time.

Problem Statement

Support involves high-volume, compliance-sensitive workflows. Support teams must resolve issues using a mix of historical Salesforce cases, Tier 3 scripts/runbooks, and an evolving knowledge base—and then document outcomes cleanly so the organization can learn from every interaction.

- Knowledge is fragmented across tickets, scripts/runbooks, and knowledge articles.
- Complex Tier 3 issues require the right script/runbook and the correct inputs, but agents often cannot find them quickly.
- When a fix is discovered, it frequently stays trapped in case notes instead of becoming reusable, searchable knowledge.
- QA and coaching are manual and inconsistent, making it hard to scale quality while maintaining compliance and data privacy.

Your goal is to build a self-learning support intelligence layer that triages new cases, recommends the best resource (KB vs. script), updates knowledge with traceability, and enables consistent QA scoring.

---

2. Hero Features – What You Could Build

Choose one direction and go deep.

Feature	**Description**
Self-Updating Knowledge Engine	Automatically generate and update knowledge articles from resolved cases and transcripts, with versioning and traceability.
AI Support Coaching Agent	Provide structured feedback on tone, accuracy, policy adherence, and empathy.
Compliance & Safety Guardrail System	Detect unsafe, non-compliant, or inconsistent responses.

Live Agent Copilot	Suggest best-practice answers in real time from the evolving knowledge base.
Root Cause Intelligence Mining	Identify recurring product failures and unclear documentation.
Knowledge Quality Dashboard	Score articles by accuracy, freshness, and real-world usage.

---

3. Hints and Resources (Refer to Section 6 of the document for details)

This challenge ships with a single Excel workbook containing multiple tabs. Dataset workbook (XLSX): SupportMind__Final_Data.xlsx

Use this dataset as an example. Feel free to add more synthetic data and make it yours as part of the hackathon.

For any questions regarding the data set please contact Arun (972) 310 9556 via WhatsApp

Data Sources

Data	Description
Support Transcripts	Call center audio → text, chat logs, ticket histories
Public Helpdesk Datasets	Zendesk / Intercom / Kaggle datasets
Product Documentation	Manuals, SOPs, internal policies
Synthetic Case Generator	Simulated support scenarios via LLMs

---

AI & Cloud Tools (Optional but Encouraged)

Tool	Purpose
LLMs (Gemini, GPT, Claude)	Knowledge extraction and article generation
Speech-to-Text	Transcribe phone calls

Vector Databases	Knowledge retrieval
RAG Pipelines	Power AI agents
Multi-Agent Orchestration	Review → extract → validate → publish → coach
Evaluation Models	Score correctness and compliance

4. Evaluation Criteria

Criterion	**What Success Looks Like**
Learning Capability	System improves knowledge automatically from conversations
Compliance & Safety Value	Detects policy violations or risky guidance
Accuracy & Consistency	Responses align with updated knowledge
Automation & Scalability	Handles thousands of conversations
Clarity of Demo	Input → AI analysis → knowledge update + coaching
Enterprise Readiness	Fits real support workflows

5. Why It Matters

Customer support is a trillion-dollar global workforce, and one of the most operationally risky functions in modern enterprises. In regulated and high-stakes environments, a single incorrect or inconsistent response can trigger compliance failures, customer harm, and significant financial loss.

Trust at this scale cannot depend on individual judgment, static documentation, or periodic audits. It must be operationalized and embedded into how knowledge is created, validated, deployed, and improved in real time.

A self-learning support intelligence system transforms customer support from a reactive cost center into a living operating system for trust. Instead of static documentation and delayed reviews, organizations gain a system that learns continuously, enforces safety and compliance as part of execution, and scales quality without scaling headcount.

The economic impact is material: faster resolution times, fewer errors, reduced training costs, lower regulatory risk, and more resilient operations. But the strategic impact is even greater - organizations gain the ability to scale automation and intelligence without sacrificing accountability or confidence.

This is not just automation.

It is foundational infrastructure for how organizations learn, govern, and act at scale.

6. Dataset Appendix

This challenge ships with a single Excel workbook containing multiple tabs. Dataset workbook (XLSX): SupportMind__Final_Data.xlsx The data is intentionally synthetic: no real customer names, property names, unit numbers, IDs, phone numbers, emails, or live URLs are included. Scripts and articles use placeholders so participants can understand what inputs would be required in a real environment. **Use this dataset as an example. Feel free to add more synthetic data and make it yours as part of the hackathon.**

6.1 Workbook Tabs and How They Connect

Tab	What it contains	How to use it (keys / joins)
README	One-page orientation for participants (joins, placeholder meaning, suggested build tracks).	Start here.
Conversations	Synthetic call/chat transcripts with agent and customer turns.	Join to Tickets on Ticket_Number (and Conversation_ID).
Tickets	Synthetic Salesforce-style case records (Description, Resolution, Tier, Priority, Script_ID, KB references).	Join to Conversations on Ticket_Number. Script_ID joins to Scripts_Master. KB_Article_ID joins to Knowledge_Articles.
Questions	1,000 synthetic customer questions with ground truth. Fields include Answer_Type and Target_ID.	Use Answer_Type to decide where Target_ID points (Scripts_Master, Knowledge_Articles, or Tickets).

Scripts_Master	Canonical Tier 3 scripts with placeholders (Script_Text_Sanitized).	Script_ID is referenced by Tickets.Script_ID and Questions.Target_ID when Answer_Type=SCRIPT.
Placeholder_Dictionary	Definitions for placeholders used in scripts and articles (e.g., <LEASE_ID>, <SUPPORT_EMAIL>).	Use to explain required inputs in demos.
Knowledge_Articles	Combined RAG corpus: seed KB plus synthetic KB articles generated from Tier 3 cases (Source_Type indicates which).	KB_Article_ID is referenced by Tickets.KB_Article_ID and Questions.Target_ID when Answer_Type=KB.
Existing_Knowledge_Articles	Seed-only KB corpus from the provided RAG node export (redacted for external sharing).	Subset of Knowledge_Articles for baseline RAG experiments.
KB_Lineage	Explicit provenance mapping from KB articles back to Tickets, Conversations, and Scripts.	Use to answer provenance questions without relying on semantic guessing.
Learning_Events	Simulated learning workflow events (gap detected -> draft KB -> review decision).	Use to demo self-learning loops and governance.
QA_Evaluation_Prompt	Standard QA rubric prompt for scoring interaction quality and case documentation (with weighted scoring + autozero red flags).	Use as human QA guidance or as an LLM evaluator prompt to output structured JSON.

6.2 Key Join Fields

Primary keys used across the workbook:
- Ticket_Number: joins Conversations <-> Tickets; also used as a Target_ID for ticket-resolution questions.

- Conversation_ID: secondary join between Conversations and Tickets when needed for disambiguation.
- Script_ID: joins Tickets / KB_Lineage / Questions to Scripts_Master.
- KB_Article_ID: joins Tickets / Questions / KB_Lineage to Knowledge_Articles (and Existing_Knowledge_Articles for seed-only).

6.3 Ground Truth and Suggested Evaluation

The dataset is designed to support measurable demos and objective evaluation without requiring proprietary systems:
- Retrieval accuracy: Use Questions.Answer_Type + Target_ID as ground truth (e.g., hit@k / exact-match ID).
- Provenance / traceability: Use KB_Lineage to validate whether a knowledge article was created from a specific ticket/transcript/script.
- Agent QA: Use QA_Evaluation_Prompt to score transcript and ticket quality (weighted scoring; autozero red flags).
- Self-learning loop: Use Learning_Events + Generated_KB_Article_ID to demonstrate how your system proposes, reviews, and publishes knowledge updates.