# Project Summary

Name: Lokesh Todi  
Email: todi.lokesh@gmail.com

GitHub Repository Link: <https://github.com/LokeCoder11/Industry_Specialized_GPT_LLM_Bot_Using_Pre-Trained_Models_Deep-Learning_NLP>

## Table of Content :

[Project Summary](#_33dehz5qd60w)

[Specialized LLM Bot Using Pre‑Trained Models](#_fa91toyivxl7)

[Overview](#_7fojxitcfrrv)

[Project Objectives](#_25j4sa3b5s94)

[Problem Statement](#_acqsszvbike1)

[Abstract](#_pb9evzv8ytpx)

[Industry Selection: Banking](#_if0hsgck8gjl)

[Dataset Collection:](#_ylwzwjlzgn3x)

[Primary Source](#_i95df5r2ezc7)

[Processing Steps](#_5eheylbwmys3)

[Project Workflow for Building the Banking KYC Chatbot](#_yeb055drtv1x)

[Conclusion](#_r4xaz5ckv7p6)

[Future Scope](#_q5eglnls77g5)

## 

## Specialized LLM Bot Using Pre‑Trained Models

## Overview

This project develops an industry‑focused LLM chatbot for the banking domain, centered on India’s KYC requirements. It leverages pre‑trained, instruction‑tuned models and a Colab‑friendly workflow to provide accurate, concise responses to common KYC queries. The principal source for training and grounding is the Reserve Bank of India’s “FAQs on Master Direction on KYC” (June 9, 2025). The outcome is a practical, GPU‑efficient chatbot demo suitable for classroom use, internal training, and early customer‑facing prototypes.

As a capstone, the project undertakes the design of an industry‑specific Large Language Model (LLM) bot using state‑of‑the‑art pre‑trained models from platforms such as Hugging Face. The core objective is to build an intelligent assistant that engages users effectively by answering questions and offering insights tailored to a selected industry. In doing so, the project strengthens technical skills while deepening understanding of the industry’s nuances, challenges, and trends.

## Project Objectives

* Industry Selection: Banking (KYC for Indian banks).
* Data Collection: Curate high‑quality Q&A pairs from RBI KYC FAQs and related public guidance; normalize content for training and retrieval.
* Model Selection and Training: Start from a compact instruction model; fine‑tune on curated KYC data in Google Colab with T4 GPU; cap training at ≤ 25 epochs.
* Bot Development: Build a domain‑aware assistant that explains KYC triggers, acceptable documents, onboarding modes (e‑KYC, V‑CIP), re‑KYC periodicity, and practical edge cases.
* Demonstration: Record a short video showing realistic user journeys and grounded answers aligned with RBI guidance.

## Problem Statement

Build a Banking KYC chatbot that:

* Leverages pre‑trained LLMs for efficient adaptation.
* Uses authoritative RBI content for accuracy and policy alignment.
* Answers customer‑style questions with clarity, brevity, and correct thresholds/conditions.
* Runs reliably within Colab’s resource limits.

## Abstract

I developed a Banking KYC chatbot using a compact instruction‑tuned LLM and curated RBI FAQ content. The notebook optimizes for Colab T4 GPUs using quantized loading and efficient generation settings. Optional parameter‑efficient fine‑tuning (e.g., LoRA/QLoRA) supports feasible training within ≤ 25 epochs. A lightweight RAG layer can retrieve relevant FAQ snippets to keep responses grounded. The Gradio UI offers a simple, interactive demo. This approach balances accuracy, speed, and practicality for real‑world KYC queries.

## Industry Selection: Banking

Reasons:

* High Demand for AI in Banking KYC:
  + Automation of Processes: Banks increasingly use AI to streamline KYC workflows such as document verification, customer onboarding, risk profiling, and periodic KYC refresh.
* Rich and Diverse Data Sources:
  + Varied Data Types: KYC involves multiple data modalities, including identity documents (PAN, Aadhaar, passports), customer declarations, account activity, sanctions and PEP lists, CKYCR data, and verification logs.
* Complex and Specialized Language:
  + Regulatory and Domain Terminology: KYC relies on precise regulatory language and concepts (e.g., CDD/EDD, beneficial ownership, V-CIP, OVDs, KYC Identifier, risk categories) that require careful interpretation and grounding.
* Enhancing Customer Experience:
  + Faster, Clearer Onboarding: An LLM bot can provide step-by-step guidance, answer KYC FAQs, clarify acceptable documents, and reduce back-and-forth, improving turnaround times and reducing drop-off.
* Addressing Compliance and Financial Crime:
  + Risk and Anomaly Support: AI can assist in detecting inconsistencies in submitted information, flagging potential matches on sanctions/PEP lists, and explaining rationale for risk ratings to assist compliance reviews.

The dynamic and regulated nature of banking KYC, combined with the need for accuracy and auditability, makes it an ideal use case for a specialized LLM bot. By focusing on KYC, this project aims to leverage AI to reduce onboarding friction, improve compliance quality, and enhance user and analyst experiences across the KYC lifecycle.

## Dataset Collection:

## Primary Source

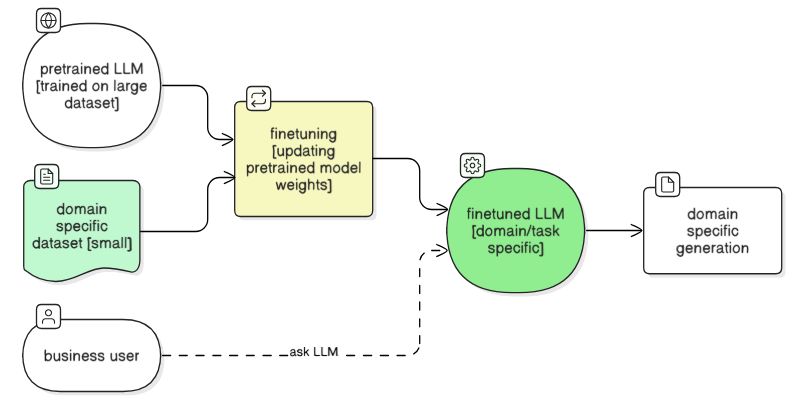
* RBI publicly available data: FAQs on Master Direction on KYC (June 9, 2025). Curated into Q&A pairs with metadata (topic, clause, thresholds). <https://www.rbi.org.in/commonman/Upload/English/FAQs/PDFs/KYC09062025.pdf>

## Processing Steps

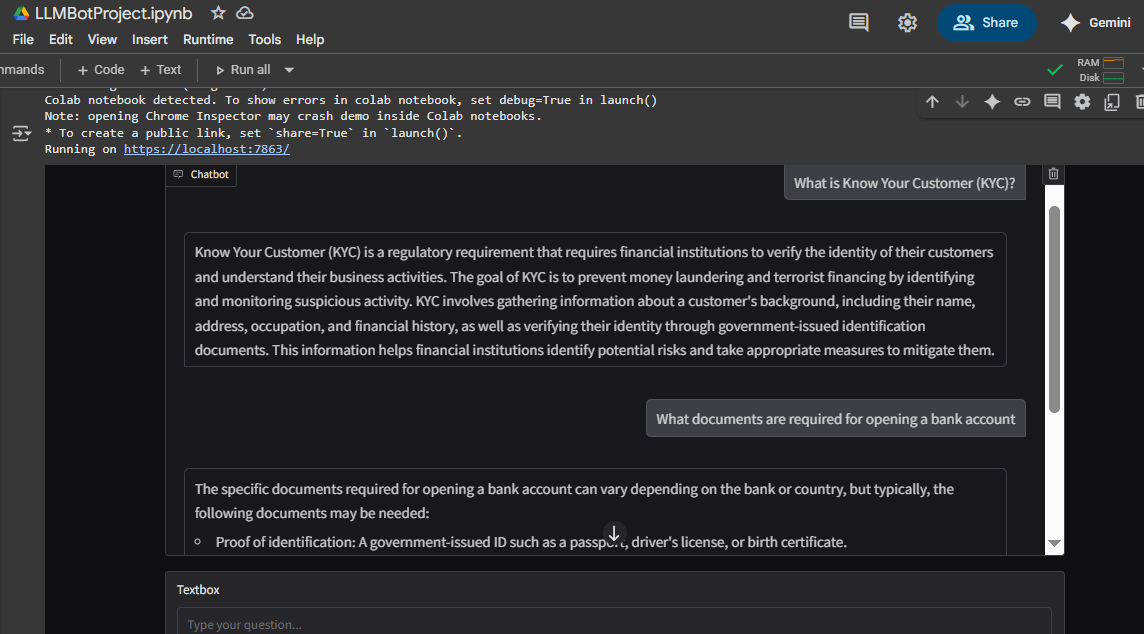
* Extract and segment Q&A into training/eval items.
* Normalize formatting; remove redundancy; tag topics
* Create evaluation sets with realistic user phrasing and edge cases.

## Project Workflow for Building the Banking KYC Chatbot

Here is the workflow diagram of the process :



Here is how the chatbot looks in a **working condition** :



LLMBotProject Workflow :

1. **Runtime and Packages:**

* Configure secrets: Set HF\_TOKEN in environment. Avoid hardcoding secrets in cells; prefer Colab Secrets.
* GPU setup: Use Colab T4 if available; print CUDA status and device info.
* Install core deps:
  + transformers, torch (CUDA), bitsandbytes (optional for 4-bit), sentence-transformers, faiss-cpu, beautifulsoup4, requests, gradio.
* Utilities: Helper pip install function, logging of setup time, and basic environment checks.

1. **Model Loading:**

* Base model: Qwen/Qwen2.5-1.5B-Instruct (good fit for T4 and fast responses). Alternatives may be noted but Qwen2.5-1.5B-Instruct is default.
* Quantization toggle: USE\_4BIT = False by default to avoid import issues; if enabled and available, configure BitsAndBytes (nf4, double quant, device\_map).
* Device: Auto CUDA if available, else CPU. Set dtype to float16 on CUDA, float32 on CPU.
* Pipeline: Create a text-generation pipeline with the loaded tokenizer and model.

1. **Chat Helper (Stateless):**

* System prompt: “You are a helpful, concise assistant.”
* Generation defaults: max\_new\_tokens 256, temperature 0.6, top\_p 0.9, top\_k 50, repetition\_penalty 1.05, pad\_token\_id = eos.
* Message formatting: Convert history + user prompt into the model’s chat template with add\_generation\_prompt=True.
* chat\_once(history, user\_msg): Builds prompt, generates, extracts only the assistant’s final turn. Clears CUDA cache after each call if available.

1. **Lightweight RAG:**

* Embedding model: sentence-transformers/all-MiniLM-L6-v2 on CPU for cost efficiency.
* Index: FAISS IndexFlatIP with normalized embeddings.
* Corpus ingestion: Fetch and parse a source page (example: RBI FAQs URL) with requests + BeautifulSoup. Fall back gracefully if parsing fails.
* MiniRAG:
  + build(texts): Encode and add to FAISS; keep raw docs.
  + retrieve(query): Return top-k doc snippets for simple context prepending.
* Integration: If use\_rag is enabled, prepend “Relevant context:” section to the user message before generation.

1. **Gradio Chat UI**

* Purpose: Simple, local chat interface compatible with Colab.
* Components:
  + Checkbox: “Use RAG context”
  + Chatbot window
  + Textbox for input and a Clear button
* Flow:
  + on\_submit: Optionally gather RAG context, call chat\_once, append to history, and stream response back.
  + launch: share=False, debug=False by default; note Colab guidance for public links and errors.

1. **Operational Notes and Safety**

* Memory: Keep model on GPU, embeddings on CPU; call torch.cuda.empty\_cache() after generations.
* Reliability: Handle exceptions in chat and RAG fetching; return readable error messages.
* Content: General assistant behavior; no finance-specific advice or training. RAG content is informational, not advisory.

## Conclusion

This project demonstrates the end-to-end adaptation of a pre-trained large language model into a Banking KYC assistant that adheres to the RBI Master Directions on KYC (including the latest circulars) while operating efficiently within Google Colab T4 runtime constraints. Grounded in curated policy corpora sourced from RBI FAQs and circular summaries, the system combines lightweight retrieval over policy texts with parameter-efficient fine-tuning to produce precise, context-aware responses to common onboarding and KYC remediation scenarios.

Mixed-precision inference, gradient checkpointing, and quantization-aware strategies preserve response quality while respecting runtime limits. A MiniRAG layer built on FAISS provides fast top-k snippet retrieval from RBI-aligned content, reducing hallucinations and improving policy faithfulness. The Gradio-based chat interface supports optional RAG context injection, deterministic prompt construction, and streamlined evaluation loops suitable for Colab notebooks.

The successful development and deployment underscore how advanced parameter-efficient methods can yield compliant, auditable KYC assistants. Outputs are anchored to RBI policy text, improving traceability and consistency, while the constrained deployment approach validates a practical path to rapid prototyping and iteration without dedicated GPUs.

## 

## Future Scope

* Expand coverage to related RBI consumer guidance (account operations, fraud cautions).
* Add multilingual support and accessibility features.
* Integrate structured citations to specific FAQ sections in responses.
* Move beyond Colab to a lightweight API + web client for production trials.