# RAG-Powered Intelligent Chatbot for Enhanced Internal and Customer Support in Online Business Platforms

*Hassan Ali (25100037); Tariq Mehmood (23030009); Ubaidullah Azeem (23030027)*

## Introduction

This is an application-based project that aims to develop an intelligent chatbot powered by Retrieval-Augmented Generation (RAG) to streamline both internal operations and customer-facing interactions for an online business platform, such as an online food delivery service (e.g. FoodPanda). The chatbot will leverage large language models (LLMs) to handle a variety of complex queries, ranging from internal employee inquiries about company policies to customer support for real-time order information. By integrating data sources like SQL databases, PDFs, and web pages, coupled with a few approaches found from literature review, the chatbot will serve as a highly efficient and scalable solution for both employee productivity and customer satisfaction.

1. Internal Support:
    - Enable employees to query internal documents (HR, IT, finance, procurement policies, etc.) using an intuitive chatbot interface.
    - Ensure secure, real-time access to documents in various formats (PDFs, PPTs, Confluence pages, etc.), improving operational efficiency.
2. Customer Support:
    - Provide accurate, real-time responses to common customer queries, such as FAQs, delivery statuses, terms and conditions, and company policies.
    - Integrate a SQL database to allow the chatbot to retrieve personalised customer information and resolve specific order-related inquiries seamlessly.
3. Data Privacy and Guardrails:
    - Implement strict data segregation and role-based access control to ensure that sensitive internal data is not exposed to customers.

## Related Work

Since this is an application-based project, there were few papers which demonstrated how and what to do to make a chatbot for internal organisational use, some mentioned about benchmarks and performance evaluation while some presented the idea of integrating RAG for text-to-SQL.

*FACTS About Building Retrieval Augmented Generation-based Chatbots* [1] talks about a FACTS framework (freshness (F), architectures (A), cost economics of LLMs (C), testing (T), and security (S)) for developing enterprise level chatbots, it highlights the concerns with developing RAG chatbots like meticulous engineering of rag pipelines, model finetuning, engineering prompts etc and lists down the 15 control points of RAG pipelines to achieve user acceptable levels of quality. This paper also makes use of multiple formats of data like serviceNow, confluence etc

*From RAG to QA-RAG: Integrating Generative AI for Pharmaceutical Regulatory Compliance Process* [2] talks about a more specific approach “QA RAG” as compared to simple RAG and how it is an improvement over general RAG as it may fall short in the regulatory compliance domain, where domain-specific and highly specialised information is required. Their approach is to retrieve half documents using the actual query and other half using a prompt from a fine-tuned LLM which optimises query to get highly relevant chunks and then using a BGE re-ranker to get relevant docs at the top. We can utilise this approach for our customer support RAG pipeline. Or we can employ a similar multi-query approach to fetch relevant chunks as mentioned in [6].

*LLM4PM: A case study on using Large Language Models for Process Modeling in Enterprise Organizations* [3] presents the first case study showcasing how LLM can be used for process modelling in large enterprises, specifically, Hilti Group. They mostly list down their methodology and approach, but we found their human evaluation part to be interesting. They tested it with 10 human evaluators from within the organisation, and asked questions like: "would you use this in your daily routine?" which is something we plan to do as well.

*Retrieval augmented text-to-SQL generation for epidemiological question answering using electronic health records* [4] combines text-to-SQL generation with retrieval augmented generation (RAG). Utilising a dataset of manually annotated question-SQL pairs, the study demonstrates that RAG significantly improves SQL query generation performance compared to traditional methods. We will be using a similar but more general approach to converting natural language queries to SQL to provide real-time customer support based on the relevant data. However, the approach mentioned in the paper will be particularly relevant to us if we use our RAG pipeline for a customer that deals with specialised terminologies (such as medical domain) where RAG could be used to create effective SQL queries.

*Better Customer Support Using Retrieval-Augmented Generation (RAG) at Thomson Reuters* [5] employed RAG to deliver better, faster customer service at Thomson Reuters. The interesting thing is that they were able to reduce resolution using a GPT 4 powered solution, coordinating the company's research and customer success functions to deliver a solution.

## Proposed Approach

We will adopt a step-by-step approach to develop the RAG pipeline for the chatbot application:

1. **Data Collection/Preparation:** Collect and organise internal and external documents (PDFs, PPTs, Confluence pages, Web pages, FAQs etc) for both employee and customer queries. This might involve creating synthetic data as finding real world data of a company would involve privacy concerns. We will prefer to use the publicly available data as-is.
2. **Document Chunking and Embedding:** Develop a strategy to chunk documents based on document type (PDFs, Web pages etc) and create embeddings for efficient retrieval.
3. **Database Initialization:** Set up a vector database to store document embeddings and relevant metadata for easy querying.
4. **Retrieval Step:** Implement retrieval mechanisms (utilising prompt engineering and reranking methods) to fetch the most relevant document chunks based on user queries. For SQL queries, employ Function Calling to fetch data from MySQL.
5. **Prompt Engineering:** Design and refine prompts that leverage retrieved information to generate accurate, contextually aware responses. Incorporate separate LLM chains and agents to ensure security and privacy of data and incoming queries.
6. **Streamlit Frontend:** Develop a user-friendly interface using Streamlit for both internal employees and customers to interact with the chatbot.
7. **Evaluation:** Benchmark chatbot performance to detect any hallucination or retrieval failures or incorrect responses.
8. **Deployment:** Finalise and deploy the chatbot solution in production using FastAPI, ensuring scalability, security, and ongoing monitoring for optimization.
9. **Optimizations:** If time permits, compare various LLMs focusing on performance factors such as cost-efficiency, accuracy, response time, and overall user satisfaction.

## Timeline and division of work

| Milestone | Timeline | Division of Work |
|---|---|---|
| Data Preparation | 19 Oct 2024 | Hassan, Tariq, Ubaid |
| Prepare Document Embeddings & Setup Database | 25 Oct 2024 | Hassan, Tariq, Ubaid |
| Prepare Retrieval Pipeline | 01 Nov 2024 | Hassan, Tariq |
| Prompt Engineering | 08 Nov 2024 | Hassan, Ubaid |
| Frontend Development | 15 Nov 2024 | Tariq, Ubaid, Hassan |
| Evaluation | 29 Nov 2024 | Hassan, Tariq, Ubaid |
| Deployment | 06 Dec 2024 | Hassan, Ubaid |
| Optimization (if time permits) | 11 Dec 2024 | Hassan, Tariq |

## Github Repository link

github.com/HassanAli101/RAG-Enterprise-Chatbot

## References

[1] Rama Akkiraju et. al, FACTS About Building Retrieval Augmented Generation-based Chatbots arXiv: 2407.07858v1, 2024
[2] Jaewoong Kim et. al, From RAG to QA-RAG: Integrating Generative AI for Pharmaceutical Regulatory Compliance Process, arXiv: 2402.01717, 2024
[3] Clara Ziche et. al, LLM4PM: A case study on using Large Language Models for Process Modeling in Enterprise Organizations, arXiv: 2407.17478, 2024
[4] Angelo Ziletti et. al, Retrieval augmented text-to-SQL generation for epidemiological question answering using electronic health records, arXiv: 2403.09226, 2024
[5] Unni, K. (2023) *Better customer support using retrieval-augmented generation (RAG) at Thomson Reuters*, *Medium*. Available at: https://medium.com/tr-labs-ml-engineering-blog/better-customer-support-using-retrieval-augmented-generation-rag-at-thomson-reuters-4d140a6044c3 (Accessed: 04 October 2024).
[6] Langchain-Ai (no date) *Langchain-ai/RAG-from-scratch*, *GitHub*. Available at: https://github.com/langchain-ai/rag-from-scratch (Accessed: 04 October 2024).