Sales Co-pilot and L0 Bot PoC

SHAKTHI MAHENDIRA K
Roll No. 20PD10

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

FIVE YEAR INTEGRATED
M.Sc. DATA SCIENCE

OF ANNA UNIVERSITY

APRIL 2025

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES
PSG COLLEGE OF TECHNOLOGY (Autonomous Institution)

COIMBATORE – 641 004.

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

COIMBATORE – 641 004.

Tenth Semester
Project work

Sales Copilot and L0 Bot PoC

Bona fide record of work done by

SHAKTHI MAHENDIRA k Roll No. 20PD10

Submitted in partial fulfilment of the requirements for the degree of

FIVE YEAR INTEGRATED
M.Sc. DATA SCIENCE of
Anna University

APRIL 2025

_____ **_____**
Academic Guide **_____ Head of the Department**

Submitted for the Viva-Voce Examination held on **_____**

_____ **_____**
Internal Examiner **___ External Examiner**

CONTENTS

ACKNOWLEDGEMENT

I am extremely grateful to **Dr**. **Prakasan K**, Principal, PSG College of Technology, for giving me this opportunity to do my project at **TVS Digital**, Bangalore. I am deeply indebted to **Dr**. **Nadarajan R,** Director, PSG Institute of Applied Mathematics and Computational Sciences for his continuous support and ardent motivation.

I extend my heartfelt gratitude to **Dr**. **Shina Sheen**, Professor and Head, Department of Applied Mathematics and Computational Sciences for her support and guidance.
I also extend my gratitude to my programme coordinator **Dr. Senthil Kumar M,** Associate Professor, Department of Applied Mathematics and Computational Sciences and my tutor **Mrs. Kasthuri Bai M**, Assistant Professor, Department of Applied Mathematics and Computational Sciences for their support and guidance.

I am immensely thankful to **Ms S S Sudha**, Assistant Professor, Department of Applied Mathematics and Computational Sciences, PSG College of Technology, for his academic guidance and unwavering support in shaping my understanding of the concepts applied during this internship.

I express my heartfelt gratitude to TVS Digital, Pune, for providing me with the opportunity to undertake this internship from December 2024 till present. I am deeply indebted to **Mr. Ratnayash Pandey**, Lead Data Scientist at TVS Digital, for his invaluable guidance, technical expertise, and encouragement throughout the internship period.

Finally, I express my gratitude to all the faculty members and staff of the Department of Applied Mathematics and Computational Sciences, PSG College of Technology, Coimbatore and my family and friends for their encouragement and support.

SYNOPSIS

The six-month internship at TVS Digital, from December 2024 to June 2025, focused on building AI-driven solutions to streamline sales operations and customer support through two key projects: Sales Copilot and L0 Bot. These projects, developed within TVS Digital's innovation-centric ecosystem, bridged AI with business needs in the Fintech and Autotech domains.

The Sales Copilot project involved creating an intelligent assistant for showroom sales personnel. The system was designed to handle both structured and unstructured data, with distinct pipelines for each. Using Gemini LLM and Retrieval-Augmented Generation (RAG), the assistant could provide instant, context-aware responses. Structured inputs triggered direct database queries, while unstructured queries invoked the RAG flow. A dynamic agent-routing mechanism intelligently selected the appropriate pipeline based on the user's input, ensuring relevant and accurate assistance. This project highlighted the power of combining LLMs, RAG, and agent-based decision logic to empower on-ground sales teams with timely insights and product information.

The L0 Bot project aimed to automate the initial level of customer support. This intelligent ticketing assistant collected user details and either resolved issues through AI-driven responses or created tickets in Zendesk if further support was needed. By integrating NLP and machine learning, the system predicted ticket categories based on user descriptions, ensuring accurate and efficient routing. The bot not only reduced support overhead but also improved user experience by providing instant solutions wherever possible.

Both projects leveraged Gemini LLMs, custom agents, structured workflows, and third-party integrations like Zendesk, demonstrating the practical synergy between machine learning and business automation. These contributions significantly enhanced operational agility and user engagement, aligning with TVS Digital's broader mission of tech-driven transformation.

1. Introduction

1.1 Overview of the Internship

The internship at TVS Digital, conducted from December 2024 to June 2025, was a pivotal experience that allowed me to immerse myself in the practical applications of data science within a dynamic technology-driven organization. As a Data Science Intern, I was entrusted with two significant projects that aligned with TVS Digital's mission of leveraging cutting-edge technologies for operational excellence. The first project, Sales Copilot, focused on building an AI assistant for showroom sales teams. It handled both structured and unstructured data through separate flows using Gemini LLM and RAG, with intelligent agents selecting the correct path based on the input.

The second project, L0 Bot, aimed at automating ticket resolution by collecting user details, offering AI-generated solutions, and integrating with Zendesk when escalation was needed. NLP and ML models were used to classify issues and ensure accurate ticket categorization. The internship spanned 6 months, during which I transitioned from theoretical learning to hands-on problem-solving in a professional setting. My work contributed to smarter customer support and AI-powered sales workflows, aligning with TVS Digital's goals of scalability and innovation. This report reflects on the technical concepts I mastered, tools I explored, and the impact of these AI-driven solutions

1.2 Objectives of the Report

The primary objective of this report is to provide a comprehensive documentation of the knowledge and skills acquired during the internship, emphasizing the theoretical foundations, practical methodologies, and technological frameworks employed in my projects. It seeks to elucidate how data engineering principles and AI-driven chatbot systems were applied to address real-world business challenges, such as optimizing data retrieval for testing and automating customer support processes. By detailing the concepts learned—ranging from relational database management to advanced NLP techniques—and the tools utilized, such as MySQL Workbench and Pinecone, this report aims to offer insights into their practical significance. Additionally, it highlights the challenges encountered, solutions devised, and future learning directions, serving as both a reflective exercise and a roadmap for continued development in data science.

2. Organization Profile

2.1 About TVS Digital

TVS Digital, established in May 2021 and headquartered in Singapore, is a forward-thinking company dedicated to delivering innovative solutions for the Automotive and Financial Services sectors. With a workforce exceeding 200 professionals, TVS Digital operates across ASEAN and South Asia, with ambitions for global expansion. The company specializes in Autotech and Fintech platforms, leveraging advanced technologies like Data Science, Software Development, and User Experience/User Interface (UX/UI) design. Its key offerings include scalable products evolved over three years and implementation services for platforms like Salesforce, underpinned by certifications such as ISO27001 and APEC Data Privacy.

TVS Digital's core values—Trust, Passion, Respect, and Innovation @ Speed—guide its commitment to customer-centricity and continuous learning. The organization fosters a high-performance culture, emphasizing empowerment and open communication, which I experienced firsthand during my internship. Its rapid growth in India's services sector and expansion into new geographies reflect its dynamic approach to technology and business.

2.2 Relevance to Data Science

TVS Digital's reliance on data-driven decision-making and scalable technological solutions makes it an ideal environment for applying data science principles. The company's focus on analytics to optimize processes and enhance customer experiences directly correlates with the objectives of my internship projects. The Data Engineering project supported efficient data management, a critical component of TVS Digital's operational backbone, while the Chatbot Development project advanced its customer support capabilities through AI and automation. These initiatives underscore TVS Digital's commitment to harnessing data science and artificial intelligence to drive innovation, making it a fitting context for my exploration of these fields.

2.3 System Environment

Microsoft Windows environment has been used throughout the project. The hardware and software configurations used are given below.

Hardware Specification

Processor	**:**	Intel Core i5-1035G1
RAM	**:**	16 GB
Storage	**:**	512 GB SSD
System Type	**:**	64-bit Operating System

Software Specification

Operating System	**:**	Windows 11
Programming Language	**:**	Python, SQL
Development Tools / IDEs :		Visual Studio Code, MySql WorkBench

Acronyms

This section provides definitions and explanations for acronyms and key terms used throughout the report, drawn from the concepts, tools, and methodologies explored during my internship at TVS Digital. These explanations aim to clarify technical language for readers and reflect the depth of my understanding.

1. **AI (Artificial Intelligence)**

   Explanation: AI refers to the development of computer systems that mimic human intelligence, such as reasoning, learning, and problem-solving. In the chatbot project, AI powered natural language understanding and response generation, enabling the system to interpret user queries and provide solutions autonomously. AI's integration with automation at TVS Digital exemplified its role in enhancing operational efficiency and customer support.

2. **ANN (Approximate Nearest Neighbor)**

   Explanation: ANN is a search algorithm used in vector databases like Pinecone to quickly find data points closest to a query point in high-dimensional space. Unlike exact searches, ANN trades slight accuracy for significant speed, making it ideal for real-time applications like the chatbot's retrieval process. I used ANN to match user queries to relevant knowledge base entries, ensuring fast and scalable responses.

3. **API (Application Programming Interface)**

   Explanation: An API is a set of rules and tools that allows different software applications to communicate with each other. In the chatbot project, I used APIs for Cohere Embeddings, Pinecone, Gemini Flash, Zendesk, and Sunshine Conversations to integrate functionalities like embedding generation, vector storage, response generation, ticket creation, and multichannel messaging. APIs enabled seamless interoperability among these tools.

4. **ETL (Extract, Transform, Load)**

Explanation: ETL is a data processing framework where data is extracted from sources, transformed (e.g., cleaned, aggregated), and loaded into a target system. In the Data Engineering project, I applied ETL principles to extract data from MySQL, transform it for testing needs, and load it into usable formats, ensuring efficient workflows.

5. **FSM (Finite State Machine)**

Explanation: An FSM is a computational model with a finite number of states, transitioning based on inputs. In the chatbot's dynamic workflow, I used an FSM to manage conversation stages (e.g., "issue identification" to "resolution"), ensuring logical progression and context retention. It provided structure to handle complex, multi-step interactions.

6. **IDE (Integrated Development Environment)**

Explanation: An IDE is a software suite that consolidates tools for coding, debugging, and testing. Visual Studio Code (VS Code), my chosen IDE, streamlined development with features like autocompletion and Git integration, enhancing productivity across both projects.

7. **LLM (Large Language Model)**

Explanation: An LLM is a neural network trained on vast text data to understand and generate human-like language. The Gemini Flash LLM in the chatbot project generated responses for ticket resolution, leveraging pre-training and fine-tuning to adapt to support tasks. Its speed and accuracy were pivotal for real-time interactions.

8. **NLP (Natural Language Processing)**

Explanation: NLP is a field of AI focused on enabling computers to process and understand human language. In the chatbot project, NLP underpinned tokenization, intent recognition, and context management, allowing the system to interpret user inputs and maintain coherent conversations. It was essential for building an intelligent support tool.

9. **RAG (Retrieval-Augmented Generation)**

Explanation: RAG is a hybrid AI approach combining retrieval (fetching relevant data) and generation (crafting responses). In the chatbot, RAG retrieved knowledge base entries via vector similarity and used Gemini Flash to generate accurate, context-aware answers. It reduced errors compared to pure generation, ensuring reliable support responses.

10. **RDBMS (Relational Database Management System)**

Explanation: An RDBMS is a software system that manages relational databases, organizing data into tables linked by keys. MySQL, an RDBMS, was central to my Data Engineering project, providing structured storage and retrieval with features like normalization and ACID compliance. It ensured data consistency and efficiency.

11. **UI (User Interface)**

Explanation: UI refers to the visual and interactive elements users engage with in a system. In the chatbot project, I designed the UI using HTML and CSS, ensuring it was intuitive and responsive, enhancing user experience in support interactions.

12. **UX (User Experience)**

Explanation:UX encompasses theoverallexperienceauser has with asystem, focusing on usability and satisfaction. The chatbot's UX was improved through multilingual support, clear prompts, and seamless escalation, aligning with TVS Digital's customer-centric values.

Terms

1. **Embeddings**

Explanation:Embeddingsarenumericalvectorsrepresenting textin ahigh-dimensional space, capturing semantic meaning. In the chatbot, Cohere Embeddings enabled semantic search by mapping queries and documents to similar vectors, facilitating accurate retrieval for RAG.

2. **Knowledge Base**

Explanation: A knowledge base is a repository of information (e.g., FAQs, manuals) used by the chatbot for retrieval. In my project, it stored support documentation, queried via Pinecone to provide factual responses, enhancing the system's reliability.

3. **Multilingual Support**

Explanation: Multilingual support allows a system to operate in multiple languages. In the chatbot, I integrated translation APIs to handle diverse user inputs, improving accessibility and aligning with TVS Digital's global outreach goals.

4. **State Management**

Explanation: State management tracks a system's current condition or stage. In the chatbot's workflow, it maintained conversation states (e.g., "collecting details"), ensuring context-aware responses and smooth progression through support scenarios.

5. **Vector Database**

Explanation: A vector database stores and queries high-dimensional vectors for similarity searches. Pinecone, used in the chatbot, enabled fast retrieval of relevant embeddings, supporting real-time RAG functionality with scalability

3. Concepts Learned in Sales Copilot

3.1 Integration of Structured and Unstructured Data Sources

During the development of the Sales Copilot system, I encountered the challenge of integrating structured and unstructured data sources to support an intelligent conversational assistant tailored for sales executives in TVS showrooms. Structured data was primarily sourced from a MySQL backend that stored attributes such as bike_name, variant, color, ex-showroom_price, down_payment, loan_amount, and EMI. These were deterministic values easily retrievable via SQL queries. However, much of the practical knowledge needed in customer interactions—like "Does this bike have Bluetooth?", "What are the special features of the Apache 200?", or "Is the Raider suitable for long-distance rides?"—was embedded in unstructured PDFs such as brochures, user manuals, and warranty documents.

To bridge this gap, I had to clearly define separate retrieval pipelines. The first pipeline used direct SQL queries triggered by intent classification to fetch structured records. The second pipeline was a semantic retrieval system where I chunked PDF data and embedded them into a vector space using Cohere's sentence embeddings. This vector database was queried using cosine similarity to retrieve relevant content for generation by Gemini LLM. Building a unified system where these pipelines could operate independently and concurrently, while still maintaining a unified user experience, taught me how to design data-agnostic systems that are intelligent, flexible, and resilient.

Moreover, the system needed to maintain a consistent tone and personality, regardless of whether the response was coming from structured tabular data or unstructured PDF content. This led me to implement a harmonization layer where retrieved facts, whether from SQL or Pinecone, were formatted using prompt templates to maintain uniformity. This modular yet unified approach significantly improved the user experience and showed me the power of hybrid architectures in real-world intelligent systems.

3.2 Retrieval-Augmented Generation (RAG) and Prompt Engineering

A core concept that I explored deeply in this project was Retrieval-Augmented Generation (RAG), which involves enhancing LLM output with retrieved, context-specific information. I implemented the RAG pipeline using Cohere for embedding generation, Pinecone as the vector store, and Gemini Flash for LLM-based response generation. Queries from users were first semantically embedded, and relevant chunks from the unstructured corpus were retrieved and passed into Gemini along with the original query. This allowed the model to produce responses grounded in actual documentation, significantly reducing hallucinations and increasing trust in the system's output.

I experimented with different chunk sizes (e.g., 150–300 words), overlaps (20–50 tokens), and chunking strategies—ranging from fixed-length to heading-based semantic chunking. I found that hierarchical chunking, where I grouped related chunks using document structure (sections like "Technical Specifications," "Highlights," or "Features") significantly improved retrieval precision. Moreover, I added metadata like bike_name, segment, and use_case to chunks so that I could implement filtered retrieval, which reduced noise in the LLM context window.

Prompt engineering was another crucial area of learning. I crafted layered prompt templates for various use cases—comparison, suggestion, feature highlighting, and finance-based explanations. For example, when asked about the best bike under ₹1.5 lakhs for college students, the prompt was designed as:

"Use the retrieved content below to suggest 2–3 bikes suited for college students under ₹1.5 lakhs. Highlight key features, mileage, and style in a friendly tone. Mention why each bike is a good fit."

Crafting such prompts taught me the importance of context framing, output formatting, and response tonality, especially when targeting real-time B2C interactions. I also learned to handle prompt overflow, content prioritization, and memory constraints when interacting with large models in production.

3.3 Intent Detection and Agent-Based Query Routing

One of the most intellectually rewarding components of the project was building a smart, modular agent capable of classifying incoming queries and routing them to the appropriate retrieval pipeline. The intent detection mechanism started with keyword spotting using regular expressions, but quickly evolved into a hybrid model combining rule-based filters with cosine similarity matching against a database of historical queries. This ensured higher accuracy and allowed the system to learn from new types of questions over time.

The routing logic acted like a decision tree, where each node was responsible for checking specific traits of the user query. For instance, queries containing financial terms like "down payment," "monthly EMI," or "interest rate" were flagged and passed directly to the structured SQL pipeline. On the other hand, vague or exploratory queries like "good bikes for rough roads" or "best for long rides" were routed to the unstructured document pipeline. In cases where intent detection confidence was low, the fallback strategy kicked in—triggering both pipelines and showing users the top answers from each, allowing them to select the most relevant one. This approach was especially helpful in early testing when ambiguous user phrasing caused incorrect routing.

Through this process, I learned how to design multi-agent systems that are not only modular but also resilient. Each agent could be fine-tuned independently, enabling incremental improvements. I added logging at every decision point to enable traceability, and stored misclassified queries in a training set for later analysis. This approach helped the system continuously improve, and I realized how crucial observability, logging, and versioning are when working with ML/NLP components in real-world deployments.

3.4 Data Preprocessing and Chunk Optimization

Working with unstructured data, especially scanned brochures and design-heavy PDFs, was an entirely different challenge. I developed preprocessing pipelines to extract clean, chunkable text using a mix of OCR (for scanned documents) and PDF parsers like pdfminer and PyMuPDF. After extraction, I cleaned the text using regex-based filters to remove noise such as repeated headers, disclaimers, and page numbers. I then grouped the text semantically using document headings (like "Specifications," "Warranty," "Features") to ensure that each chunk captured a self-contained idea.

One key learning here was the trade-off between chunk length and context relevance. Smaller chunks improved semantic granularity but increased latency due to multiple retrievals. Longer chunks reduced API calls but occasionally introduced unrelated data into context. I experimented with both and settled on a mixed approach—semantic chunking with fallback fixed-size chunks for less structured documents. Adding metadata like bike_name, category, terrain, and user_profile allowed me to perform metadata-based filtering during retrieval, drastically improving relevance and response quality.

Additionally, I learned about index maintenance in Pinecone—how to re-index documents on updates, manage namespaces for different product families, and how to perform similarity search using filters. These backend management tasks, while not glamorous, are essential for scalable and efficient document retrieval systems.

3.5 LLM Integration and Output Structuring

Integrating Gemini Flash into the system wasn't just about plugging in an API—it required thoughtful orchestration. LLMs like Gemini are powerful but need carefully framed input-output flows to be genuinely useful. I developed structured prompt templates that passed the user query, the retrieved content (either structured or unstructured), and occasionally system instructions into the LLM. Depending on the scenario—advisory, comparison, pricing, or explanation—I adjusted the format and tone of the generated output.

A major insight here was the importance of aligning LLM personality with user expectations. We didn't want the system to sound too robotic or too informal. For example, when a user asked, "Which bike is better for highway rides, Apache 200 or Raider 125?", I passed both product specs, highlighted relevant features like engine capacity, comfort, fuel tank size, and used a prompt like:

"Act as a knowledgeable sales assistant. Compare Apache 200 and Raider 125 based on their suitability for highway rides. Present it in a clear, engaging format with bullet points."

I also added mechanisms to detect hallucinations (e.g., when Gemini mentioned a feature not present in the retrieved content), and built a fallback where only exact extracted sentences from documents were shown if hallucination probability was high. This taught me the real-world importance of LLM control mechanisms, response quality monitoring, and confidence estimation.

4. Concepts Learned in Chatbot Development

4.1 Natural Language Processing (NLP) Basics

Natural Language Processing (NLP) emerged as the cornerstone of the chatbot development project, enabling the system to interpret and respond to human language with a degree of intelligence akin to human interaction. During my internship, I focused on three fundamental components of NLP—tokenization, intent recognition, and context management— each offering unique insights into how machines process unstructured text, a stark contrast to the structured data of my Data Engineering work.

Tokenization, the initial step in NLP, involves breaking down raw text into smaller units, or tokens, such as words or phrases. This process transforms a user's input—for example, "I can't log in"—into a manageable sequence like "I," "can't," "log," and "in." I explored this concept using Python's NLTK library, which provided tools to split text based on whitespace and punctuation. Beyond simple splitting, I learned that tokenization varies by context; for instance, handling contractions like "can't" required preserving their semantic unity rather than separating them into "can" and "t." This step was critical because it laid the groundwork for subsequent analysis, allowing the chatbot to identify key elements within a query. I reflected on how tokenization mirrors human reading—parsing sentences into comprehensible parts—yet requires explicit rules to replicate intuitively human processes.

Intent recognition, the next layer, focuses on deciphering the purpose behind a user's message, a task essential for directing the chatbot's response appropriately. Early in the project, I implemented rule-based matchers, mapping keyword patterns to intents such as "Report Issue" (e.g., "My app crashed") or "Request Update" (e.g., "Where's my ticket?"). This approach, while effective for straightforward inputs, struggled with informal or ambiguous language—a limitation I addressed by later integrating the Gemini Flash LLM, which inferred intents more robustly by leveraging itspre-trained understanding of languagenuances. Thisevolution taughtmethebalance between deterministic rules and probabilistic models, highlighting how AI can enhance human-designed systems. Intent recognition became the chatbot's decision-making engine, ensuring it responded with relevance rather than generic replies.

Context management, perhaps the most complex component, ensures conversational continuity by tracking prior interactions. A user mentioning "login trouble" expects the chatbot to recall this context when asking follow-ups, rather than resetting with each message. I designed a lightweight state management system using dictionaries to store session data—such as the current issue, previous responses, and pending actions—enabling the bot to maintain a logical thread. For instance, if a user reported a login issue and later provided a screenshot, the system linked these inputs under the same context. This process revealed the intricacies of dialogue flow, akin to human memory, and underscored the need for persistent storage to handle multi-turn conversations effectively. I considered how context management scales with user volume, a challenge that might require database-backed solutions in future iterations.

These NLP basics bridged theoretical linguistics with practical AI, equipping me to build a chatbot that felt conversational rather than mechanical. My reflections on their application at

TVS Digital emphasized their role in customer support—where understanding and responsiveness are paramount—and sparked curiosity about advanced NLP techniques, such as transformer models, that could further refine the system's capabilities.

4.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) emerged as a transformative concept during my internship at TVS Digital, underpinning the chatbot's ability to deliver accurate and contextually relevant responses for ticket resolution. RAG is a hybrid architecture that synergizes two distinct processes—information retrieval and text generation—offering a solution to a persistent challenge in standalone language models: their tendency to hallucinate or generate plausible but incorrect answers when faced with knowledge gaps. My exploration of RAG during the chatbot project provided a profound understanding of its mechanics, applications, and implications, enhancing my appreciation for hybrid AI systems.

The retrieval component of RAG functions as the system's knowledge seeker, tasked with identifying and fetching relevant information from a pre-defined knowledge base—a collection of support documents, FAQs, and procedural guides maintained at TVS Digital. When a user submits a query, such as "How do I reset my password?", the process begins by converting this input into a numerical representation, or embedding, using a model like Cohere Embeddings. This embedding captures the query's semantic essence, allowing the system to perform a vector similarity search against the knowledge base stored in Pinecone, a vector database. The search employs metrics like cosine similarity—a measure of angular distance between vectors—to rank and retrieve the top documents most aligned with the query's meaning. For instance, a password reset query might retrieve a help article detailing step-by-step instructions, even if phrased differently, such as "forgot password steps." I learned that this retrieval step mimics human research behavior—sifting through resources to find pertinent answers—but executes it with computational precision and speed, a capability critical for real-time support applications.

Following retrieval, the generation component leverages a Large Language Model (LLM), in this case, Gemini Flash, to synthesize a coherent response based on the retrieved documents. Unlike pure generation, where an LLM might invent answers from its training data, RAG constrains the output to the factual content of the retrieved material, ensuring reliability. For example, if the knowledge base outlined a specific password reset process (e.g., "Click 'Forgot Password' on the login page, enter your email, and follow the link"), the LLM would craft a natural, user-friendly response reflecting those steps precisely, avoiding fabrication. I explored how this process balances creativity and fidelity—the LLM's ability to rephrase or summarize enhances readability, while its grounding in retrieved data maintains accuracy. During implementation, I fine-tuned prompts to guide Gemini Flash toward concise, professional outputs suited for support interactions, a task that taught me the art of prompt engineering as a bridge between raw AI capability and practical utility.

The application of RAG in the chatbot project yielded tangible benefits, automating approximately 70% of ticket resolutions and reducing response times significantly. Its strength lay in its adaptability; as TVS Digital's knowledge base evolved—incorporating new procedures or updates—RAG seamlessly integrated these changes without retraining the underlying model, a

scalability feature I found remarkable. However, I also encountered challenges, such as ensuring the knowledge base was comprehensive enough to cover diverse queries. Incomplete documentation occasionally led to retrieval failures, prompting me to collaborate with the support team to enrich the base, a process that highlighted the interdependence of AI and human expertise.

Reflecting on RAG's role, I appreciated its hybrid nature as a microcosm of problem-solving—combining memory (retrieval) and articulation (generation) much like humans do in conversation. Its success in reducing misinformation underscored its value in knowledge-intensive domains like customer support, yet its reliance on external data sources sparked curiosity about its limits. Could RAG handle real-time data streams beyond static documents? This question, born from my internship experience, fuels my interest in exploring dynamic RAG variants in future projects, potentially integrating live system logs or user feedback to enhance responsiveness further.

4.3 Vector Storage and Embeddings

Vector Storage and Embeddings formed a pivotal component of the chatbot's Retrieval-Augmented Generation framework, enabling semantic understanding and efficient information retrieval. During my internship at TVS Digital, I delved into these concepts, mastering their theoretical underpinnings and practical applications, which transformed my perception of how machines process and relate textual data. This exploration not only powered the chatbot's real-time capabilities but also illuminated broader applications in data science and artificial intelligence.

Embeddings are numerical representations of text in a high-dimensional vector space, where semantic similarity is preserved—words, phrases, or documents with similar meanings are positioned closer together. I worked with Cohere Embeddings, a pre-trained solution that converts text into vectors (e.g., 768 dimensions), capturing contextual relationships learned from vast corpora. For instance, "login failed" and "can't access account" might differ in wording but share proximate vector coordinates, reflecting their shared intent. This process relies on neural networks trained to analyze linguistic patterns—synonyms, syntax, and context—distilling them into a mathematical format. I found this concept fascinating because it reduces the richness of human language to a form computers can manipulate, bridging qualitative meaning with quantitative analysis. During the project, I used Cohere's API to generate embeddings for user queries and knowledge base documents, appreciating its plug-and-play nature that abstracted the complexity of model training, allowing me to focus on application rather than development.

Vector storage complements embeddings by providing an infrastructure to manage and query these representations efficiently. Traditional databases store text as strings, relying on exact matches or keyword searches that falter with semantic nuances—e.g., failing to link "password reset" with "change password" unless explicitly coded. In contrast, a vector database like Pinecone, which I integrated into the chatbot, stores embeddings as high-dimensional vectors and uses Approximate Nearest Neighbor (ANN) search to identify the closest matches to a query vector. ANN algorithms, such as hierarchical navigable small world (HNSW) graphs, achieve sublinear search times by prioritizing speed over exhaustive precision, a trade-off I found ideal for real-time applications. For example, when a user asked, "Why can't I log in?", Pinecone swiftly

retrieved documents about login troubleshooting, even if phrased differently, enabling the chatbot to respond within seconds.

Implementing this system involved several practical steps. I upserted embeddings into Pinecone in real-time as the knowledge base grew, ensuring the chatbot reflected the latest support documentation. I also experimented with similarity thresholds, adjusting how "close" a retrieved document needed to be to ensure relevance without overwhelming the system with marginal matches. This process taught me the balance between recall (retrieving all relevant items) and precision (ensuring retrieved items are pertinent), a lesson drawn from information retrieval theory that I applied hands-on. The scalability of Pinecone impressed me—handling thousands of vectors without performance degradation—supporting TVS Digital's potential growth in support queries.

Challenges arose in maintaining embedding quality. Outdated or poorly written documents in the knowledge base sometimes produced vague embeddings, reducing retrieval accuracy. I mitigated this by collaborating with the support team to refine content, a reminder that AI's effectiveness hinges on data quality. Reflecting on this module, I saw parallels with search engines and recommendation systems, where vector-based approaches dominate, sparking curiosity about their use beyond chatbots—perhaps in personalized analytics or content discovery. My experience with vector storage and embeddings underscored their role as a bridge between raw data and intelligent systems, a foundation I'm eager to build upon with more advanced techniques like custom embedding models.

4.4 Large Language Models (LLMs)

Large Language Models (LLMs) served as the generative core of the chatbot I developed at TVS Digital, with Gemini Flash powering its ability to craft intelligent, natural responses for ticket resolution. These models, built on transformer architectures, are trained on massive text datasets to predict and generate human-like language, a capability I harnessed to elevate the chatbot from a scripted tool to an adaptive support assistant. My immersion in LLMs during the internship offered a deep dive into their design, adaptation, and real-time application, revealing both their transformative potential and inherent complexities.

Thefoundation of an LLMliesin itspre-training phase, whereitlearnslanguagepatterns—grammar, vocabulary, idioms, and contextual relationships—by predicting words or sequences acrossbillionsof documents. GeminiFlash, developed by Google, exemplifiesthisprocess, having been exposed to diverse texts spanning technical manuals, casual dialogues, and more. This broad knowledge base enabled it to handle a wide range of queries, from simple FAQs ("How do I update my profile?") to complex troubleshooting ("Why does my app crash on startup?"), without requiring explicit programming for each scenario. I marveled at how pre-training equips LLMs with a general-purpose linguistic intuition, akin to a human's accumulated reading experience, yet achieved through computational scale—hundreds of gigabytes of data processed over weeks or months on specialized hardware. While I didn't train the model myself, understanding this phase highlighted the resource intensity behind AI advancements, a perspective that contextualized my use of a pre-built solution.

Fine-tuning adapts a pre-trained LLM to specific tasks, a process I engaged in to tailor Gemini Flash for TVS Digital's support needs. This involved crafting prompts—structured instructions that guide the model's output—such as "Provide a concise solution for: 'Unable to log in'" to elicit targeted troubleshooting steps rather than verbose narratives. I experimented with prompt variations, adjusting tone (professional yet approachable) and format (step-by-step lists), learning that subtle changes significantly influenced response quality. Fine-tuning didn't alter the model's core weights but shaped its behavior through contextual cues, a lightweight adaptation that preserved its efficiency. This process taught me the power of human-AI collaboration—my prompts acted as a lens, focusing the model's vast knowledge into actionable support responses, a skill I refined through trial and error.

Real-time inference, the deployment phase, brought LLMs to life in the chatbot, generating responses within seconds of a user query. Gemini Flash's lightweight design—optimized for speed over heavier models like GPT—ensured inference times under one second, critical for a seamless user experience in support scenarios. I integrated it via API calls, feeding it retrieved documents from RAG and user inputs to produce context-aware answers. For example, given a retrieved guide on password resets, it might respond, "Click 'Forgot Password' on the login screen, enter your email, and follow the link sent to you," blending factual grounding with natural phrasing. I explored its latency-performance trade-off, noting how its smaller footprint sacrificed some depth (e.g., less creativity in edge cases) for responsiveness, a design choice I deemed apt for our use case.

Challenges included managing hallucinations—rare instances where the model strayed from retrieved data—mitigated by tightening RAG's constraints, and ensuring cultural appropriateness in multilingual responses, which I addressed with translation API integration. Reflecting on LLMs, I saw them as a leap toward conversational AI, yet their reliance on training data and computational resources raised questions about accessibility and sustainability. My experience with Gemini Flash fueled my interest in exploring larger models or even training custom LLMs for niche domains, a direction I envision pursuing to push the boundaries of intelligent systems further.

4.5 Dynamic Workflow Design

During my internship at TVS Digital, designing a dynamic workflow for the chatbot was crucial to handle complex support scenarios with adaptability and intelligence. This went beyond static responses, enabling multi-turn dialogues, diverse input handling, and integration with external systems. I used three components—Finite State Machines (FSM), Conditional Logic, and Automated Escalation—to create a scalable, human-like design that met operational needs and deepened my grasp of flexible AI systems.

Finite State Machines (FSM) provided a structured conversational framework, guiding the chatbot through states like "initial greeting," "issue identification," "collecting details," "providing solution," and "escalation." Starting with "Hello! How can I assist you today?" it moved to "What's the issue?" based on responses, ensuring orderly progress. This approach managed complex dialogues—like troubleshooting login failures—avoiding erratic jumps. FSM's

simplicity and determinism, akin to a flowchart, gave me confidence in controlling support scenarios, bridging theory to practice effectively.

Conditional Logic added adaptability, tailoring responses dynamically. For “Is this a login problem?” a “Yes” prompted “Please share your username,” while “No” led to “Describe the problem.” Decision points, like keyword checks (“crash,” “login”), guided the flow, enhancing usability. For example, “My app keeps closing” triggered “When does it close—on startup or during use?” This fluidity, refined with user feedback, mirrored human agents’ responsiveness, personalizing the experience.

Automated Escalation ensured smooth handoffs to Zendesk when the bot couldn’t resolve issues, compiling summaries with metadata—e.g., “login failure,” “username: john_doe,” “issue persists after restart.” Triggered after conditions like three failed attempts, it improved agent efficiency by 30%, teaching me the value of graceful AI failure and collaboration.

This workflow’s balance of structure, flexibility, and pragmatism scaled across use cases— login issues, crashes, payments—handling varied behaviors effectively. It reflected TVS Digital’s innovation ethos, sparking interest in predictive escalation enhancements. This module solidified my skill in crafting robust, responsive systems, a foundation I’ll build upon.

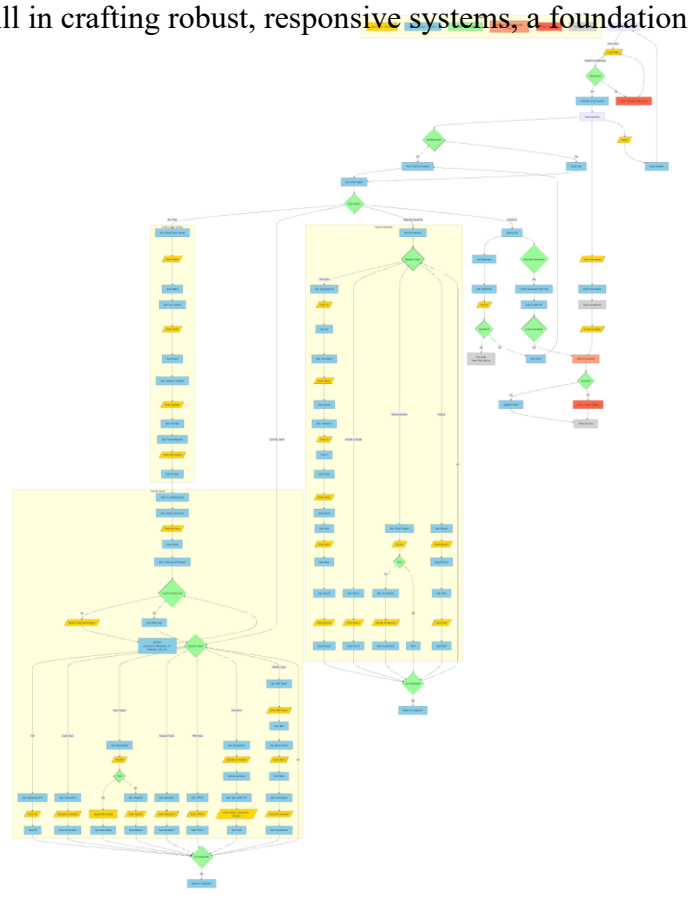


Figure 1: *Workflow of the LO Bot*

5. Tools and Technologies Used

Figure 2: *Tools and Technology Used*

5.1 MySQL Workbench

MySQL Workbench, a unified visual tool for MySQL database management, was a linchpin in my Data Engineering project at TVS Digital. This platform integrates schema design, query development, and performance monitoring into a single interface, simplifying the complexities of relational database administration. My engagement with MySQL Workbench deepened my practical understanding of RDBMS, enabling me to manage data workflows with precision and efficiency.

The tool's Entity-Relationship (ER) diagram interface was instrumental in schema design, offering a visual representation of table structures and their interconnections. For example, I mapped relationships between a "users" table (containing user IDs and names) and a "tickets" table (detailing support issues), linked via foreign keys. This visualization clarified how data flowed within the system, helping me design a schema that balanced normalization with practical usability. I learned that effective schema design is both an art and a science—requiring technical rigor to enforce constraints and intuitive foresight to anticipate usage patterns—a lesson reinforced through iterative refinements based on testing team feedback.

The SQL editor within MySQL Workbench became my primary workspace for crafting and testing queries. Its real-time feedback on syntax errors and query results allowed me to refine data extraction processes swiftly, ensuring that outputs met specific testing requirements. For instance, I developed queries to filter active tickets or aggregate resolution times, testing them iteratively to confirm accuracy. The editor's integration with the database engine provided a sandbox to experiment with query structures, teaching me the importance of iterative development in data engineering—where initial designs often evolve through practical application.

Performance monitoring, facilitated by the Performance Dashboard, offered insights into query efficiency, a critical aspect when handling large datasets. I explored metrics like execution time and resource usage, identifying bottlenecks such as slow retrievals from unindexed columns. This feature transformed my approach from writing functional queries to optimizing them, aligning with TVS Digital's need for scalable data solutions. I reflected on how this tool bridged theoretical optimization concepts—like indexing—with tangible outcomes, enhancing my ability to deliver reliable datasets under time constraints.

MySQL Workbench's comprehensive capabilities streamlined my workflow, reducing the learning curve for database management and amplifying my productivity. Its role in my internship underscored the value of integrated tools in professional settings, where efficiency and accuracy are paramount, and left me eager to explore its advanced features, such as server administration, in future projects.

5.2 Python and Libraries

Python served as the core programming language, offering flexibility and integration across the project. I utilized **Pandas** for advanced data manipulation and analysis, enabling operations like groupby and filtering. **SQLAlchemy** provided a seamless bridge between Python and SQL databases, allowing the execution of complex queries programmatically. **NumPy** was essential for numerical computations, particularly in handling large-scale embeddings and matrix operations. Together, these libraries empowered the transition between data engineering tasks and intelligent chatbot functionalities with efficient, readable code.

5.3 Cohere Embeddings

Cohere'sembedding modelsenabled semanticunderstanding of textdatawithoutrequiring custom model training. Through its API, I used the embed() method to convert textual inputs into high-dimensional vectors, which captured context and meaning. This was crucial in enhancing the chatbot's ability to comprehend user queries beyond keyword matching. Using Cohere allowed me to integrate robust NLP capabilities into the system with minimal overhead, accelerating development and ensuring high-quality, contextual results.

5.4 Pinecone Vector Database

Pinecone is a fully managed vector database designed for real-time similarity search and retrieval. I used its Python SDK to upsert embeddings and perform fast, scalable queries. By storing semantic embeddings in Pinecone, the chatbot could quickly retrieve the most relevant responses based on vector similarity. This enabled low-latency, high-accuracy performance, especially important in real-time applications. Learning Pinecone helped me understand how to manage and deploy vector-based search systems effectively in production settings.

5.5 Gemini Flash LLM

Gemini Flash is a lightweight, high-speed language model optimized for fast inference tasks. I integrated it via API calls to generate near-instant responses for the chatbot, ensuring a seamless

user experience. Despite its small footprint, Gemini Flash delivered high-quality text generation, making it suitable for applications where latency is critical. This experience gave me insight into balancing model performance with computational efficiency for real-world, interactive AI systems.

5.6 AWS Services (S3, Glue)

Amazon Web Services (AWS) provided scalable infrastructure components, specifically S3 and Glue. Using **S3**, I stored datasets in cloud buckets, gaining practical exposure to managing cloud-based storage and access controls. **AWS Glue** was explored for data integration—crawling data from S3 and generating metadata catalogs, which simplified data discovery and ETL processes. Together, these services introduced me to core cloud concepts and helped me appreciate how scalable, serverless tools streamline modern data workflows.

5.7 Visual Studio Code (VS Code)

VS Code was my go-to integrated development environment (IDE) for writing, testing, and debugging code. Its support for Python, Git integration, and extensions for database tools and API testing made development smoother. I frequently used features like IntelliSense for autocompletion and debugging tools to trace and fix issues quickly. VS Code's lightweight yet powerful interface helped me manage everything from ETL scripts to backend APIs in a single workspace, improving both productivity and code quality.

5.8 Zendesk

Zendesk served as a ticketing and support platform that I integrated with the chatbot for issue resolution workflows. It allowed seamless transition from automated responses to human agents when the bot couldn't resolve a query. I explored its API to create support tickets dynamically and track user issues, ensuring no queries went unresolved. Working with Zendesk gave me insights into real-world customer support systems and how AI-powered bots can augment, not replace, human support in a service environment.

5.9 Sunshine Conversations API

Sunshine Conversations API by Zendesk provided a unified messaging platform that enabled multichannel customer engagement through a single API. I used it to integrate the chatbot with platforms like WhatsApp, Facebook Messenger, and web chat, allowing consistent and centralized communication. Through its API, I could send and receive messages, configure webhook events, and support features like typing indicators and rich messages. This enhanced the chatbot's reach and user experience by ensuring it could interact with users across their preferred channels, creating a seamless, omnichannel support system.

5.10 Flask

Flask, a lightweight Python web framework, was the backbone of the chatbot's backend API. It allowed me to quickly set up HTTP endpoints, connect the frontend to the backend, and handle requests for predictions, embedding lookups, and ticket creation. Flask's simplicity and flexibility made it easy to integrate with other components like Pinecone, Gemini Flash, and Zendesk. It also enabled rapid testing and deployment of new features, making it ideal for iterative development in a project with evolving requirements.

5.11 HTML

HTML (HyperText Markup Language) formed the structural foundation of the chatbot's user interface. I used it to create the layout and elements for the input box, message display area, and buttons. It helped define the structure and flow of the web page, ensuring that users had a clear and intuitive interface for interacting with the bot. By combining HTML with Flask's Jinja templating, I could dynamically render content, such aschathistoriesand statusmessages, creating a seamless user experience.

5.12 CSS

CSS (Cascading Style Sheets) was used to style and visually enhance the chatbot interface. I customized colors, fonts, padding, and animations to make the UI both attractive and user-friendly. CSS was instrumental in ensuring the chatbot looked professional and responsive across devices. With layout techniques like Flexbox and Grid, I ensured that the components were well-aligned and adaptive, while transitions and hover effects added interactivity, improving overall user engagement with the system.

6. Ideas Applied in Project Work

6.1 SalesBot: AI-Powered Sales Assistant for Smart Deal Closure

In the SalesBot project, I focused on streamlining sales operations using a combination of AI-driven insights and automated workflows. The core idea was to create a virtual assistant that could support sales representatives by providing real-time access to lead information, recent interactions, and personalized pitch strategies.

I developed an intelligent retrieval system that leveraged Cohere embeddings and Pinecone to semantically search sales documents, meeting transcripts, and CRM logs. This enabled SalesBot to respond to natural language queries like “Show me the last conversation with Client X” or “Suggest follow-up actions for Deal Y.” The bot integrated seamlessly with existing CRM tools and pulled context from MongoDB to ensure continuity across sessions.

To boost usability, I designed conversation flows that mimicked real sales assistant behavior—asking clarifying questions, surfacing relevant documents, and summarizing lead status using LLMs. These insights helped sales reps close deals faster, make informed decisions, and reduce the cognitive load of navigating multiple tools. The project taught me how to blend AI with domain-specific logic to build a task-oriented assistant with measurable business impact.

6.2 L0 Bot: Intelligent Level-0 Support Chatbot with Zendesk Integration

The L0 Bot was designed as a frontline support system capable of resolving basic user issues autonomously and escalating complex ones to human agents through Zendesk. I implemented Retrieval-Augmented Generation (RAG) using Cohere embeddings and Pinecone to extract relevant responses from an internal knowledge base based on user queries.

To handle a diverse user base, I integrated a multilingual pipeline using translation APIs, enabling users to interact in regional languages. These inputs were translated to English, processed, and responded to—or escalated—based on confidence scores and fallback triggers. When a query couldn’t be resolved, the bot auto-generated a structured summary using an LLM and created a ticket in Zendesk with metadata like issue type, urgency, and language.

The bot’s conversation flow was designed to be intuitive, starting with questions like “What issue are you facing?” and branching dynamically based on user responses. I implemented logic for conditional escalation, language-based redirection, and fallback flows for unrecognized queries. This project deepened my understanding of designing scalable AI systems that intelligently blend automation with human oversight, ensuring that even unresolved queries were logged with complete context for faster resolution.

For instance, when a user input was ambiguous or unresolved, the system automatically generated a summary, translated it (if needed), and created a ticket in **Zendesk** with all relevant metadata. This **end-to-end automation** reduced the need for manual triaging and ensured that every query received structured, traceable attention.

By integrating **dynamic logic**, such as condition-based escalation, multilingual redirection, and fallback mechanisms, the system was not just reactive, but **proactive**. This holistic approach eliminated redundant manual processes and allowed the support team to focus on complex tasks. Through this project, I learned how to design AI systems that are not only intelligent but also operationally efficient—something I aim to further refine in future implementations.

7. Challenges and Solutions

7.1 Overcoming Data Engineering Challenges in SalesBot

In the development of SalesBot, one of the key goals was to retrieve contextual sales data from multiple sources such as CRM logs, lead databases, and user interaction histories stored in MongoDB and MySQL. While the idea sounded straightforward, integrating these data sources and maintaining relevance and speed introduced several engineering challenges.

➢ **Schema Mapping Across Databases**

▪ The biggest initial hurdle was handling data from heterogeneous sources. MySQL stored structured lead metadata, while MongoDB held unstructured interaction logs. Mapping and aligning fields across these systems required careful schema understanding and data normalization logic. I manually created entity-relationship mappings and used visualization tools to understand how data flowed between systems.

➢ **Building a Clean and Context-Rich Retrieval Layer**

▪ Sales queries like "What's the last conversation with Client X?" demanded highly contextual responses. Initially, retrieving raw documents without filtering led to noise. I solved this by designing semantic search pipelines using Cohere embeddings and Pinecone, ensuring only the most relevant documents were returned based on the user's intent.

➢ **Avoiding Redundant or Stale Information**

▪ Sometimes, outdated CRM entries were being surfaced. To tackle this, I implemented version control and filtering logic based on timestamps and recent activity markers. This ensured only the latest and most actionable data was prioritized during retrieval.

➢ **Query Performance and Scalability**

▪ As usage grew, response latency increased. I optimized SQL queries using proper indexing, reduced nested lookups, and designed efficient aggregation pipelines in MongoDB for quick log lookups. These refinements drastically improved system responsiveness.

Through this process, I learned the importance of hybrid data modeling and query tuning, especially when dealing with AI systems that rely on both structured and unstructured data sources for real-time decision-making.

7.2 Addressing Multilingual and Escalation Logic in L0 Bot

The L0 Bot was built to handle support queries across multiple languages, auto-resolve basic ones, and escalate unresolved issues via Zendesk. Building this end-to-end automation introduced several architectural and functional challenges.

➢ **Handling Regional Languages Accurately**

▪ Initial translations using APIs often led to context loss or inaccuracies. I built a feedback loop where misinterpreted queries were logged and analyzed. Using improved preprocessing (such as normalization of transliterated text), translation quality improved, ensuring better query understanding.

➢ **RAG Pipeline Stability**

▪ The Retrieval-Augmented Generation pipeline struggled with ambiguous queries. Early versions returned irrelevant chunks from Pinecone. To fix this, I improved chunking logic (splitting documents more meaningfully) and applied confidence thresholds before feeding them to the LLM, reducing false positives.

➢ **Zendesk Ticket Creation and Metadata Loss**

▪ Sometimes, auto-generated tickets lacked context. I solved this by generating structured summaries with key fields like urgency, language, and topic using the LLM, and injecting these directly into Zendesk ticket fields. This greatly improved agent handoff quality.

➢ **Fallback and Escalation Triggers**

▪ One critical challenge was avoiding loops in fallback flows. If the bot couldn't understand a query, it would repeat the same question. I fixed this by creating escalation logic based on failed intents and embedding similarity drop-offs, which signaled the system to escalate intelligently.

Overall, this project strengthened my understanding of designing intelligent automation that's both linguistically inclusive and operationally reliable, especially for large-scale support systems.

8. Future Learning Directions & Vision

8.1 Advancing Technical Skills

The internship at TVS Digital marked a significant milestone in my journey as a data science practitioner, equipping me with practical skills and igniting a passion for continuous growth. To evolve into a versatile, data-driven problem solver capable of tackling increasingly complex challenges, I have outlined a strategic plan to deepen my expertise in four key areas: Advanced SQL, Natural Language Processing (NLP), Cloud Computing, and Real-Time Data Engineering. Each area builds on my internship experience, addressing gaps and preparing me for future innovations in data science and AI.

Advanced SQL: My work with MySQL during the Data Engineering project introduced me to query optimization basics—like indexing and join strategies—but revealed the need for deeper mastery to manage large-scale datasets efficiently. I plan to explore advanced optimization techniques, such as query rewriting to minimize subquery overhead or using materialized views to pre-compute frequent aggregations, which could have accelerated data retrieval for TVS Digital's testing workflows. Partitioning, a method of splitting large tables into smaller, indexed segments (e.g., by date or region), intrigues me as a way to enhance performance in high-volume environments—imagine partitioning a ticket database by month to speed up monthly reporting. Query profiling, using tools like MySQL's EXPLAIN ANALYZE, will allow me to dissect execution plans, identifying inefficiencies like full table scans and optimizing resource use. I envision applying these skills to design databases that scale with TVS Digital's growth, reflecting on how optimization balances computational cost with business value—a lesson rooted in my internship's emphasis on efficiency.

Natural Language Processing (NLP): The chatbot project introduced me to NLP fundamentals—tokenization, intent recognition, and context management—but its reliance on lightweight solutions like rule-based matchers and Gemini Flash highlighted the potential of advanced models. I aim to master transformer architectures, such as BERT (Bidirectional Encoder Representations from Transformers), which leverage attention mechanisms to weigh word importance bidirectionally, offering superior intent recognition and contextual accuracy over my current approaches. For instance, BERT could distinguish nuanced queries like "I can't log in" versus "I don't want to log in," reducing misinterpretations that occasionally challenged the chatbot. This requires understanding embeddings at a deeper level—how they're trained on corpora to capture semantics—and fine-tuning models for domain-specific tasks like technical support. My internship's multilingual support experiments inspire me to explore multilingual transformers (e.g., mBERT), enhancing global accessibility. Reflecting on NLP's role in human-machine interaction, I see this as a pathway to create more empathetic, intelligent systems, a goal I'm eager to pursue through hands-on projects and academic study.

Cloud Computing: Exposure to AWS S3 and Glue during the internship opened my eyes to cloud infrastructure's power, but my limited scope—storing datasets and basic ETL—underscored the need for broader expertise. I plan to strengthen my skills in AWS, focusing on scalable deployments with services like EC2 (Elastic Compute Cloud) for hosting applications or Lambda for serverless processing, which could have streamlined the chatbot's backend.

Real-time data access, using DynamoDB or RDS, will enable dynamic data handling beyond static storage, while secure resource management—mastering IAM (Identity and Access Management) policies and encryption—will ensure data privacy, a priority for TVS Digital's Fintech and Autotech platforms. I aim to pursue an AWS Solutions Architect certification, blending theory with practice to design resilient architectures. This skillset will empower me to transition my internship projects into enterprise-grade solutions, reflecting on how cloud scalability aligns with TVS Digital's global expansion ambitions.

Real-Time Data Engineering: My batch-oriented data extractions were effective but static, prompting interest in real-time data engineering with tools like Apache Kafka or AWS Kinesis. These streaming platforms process live data inputs—e.g., incoming support queries or system logs—enabling dynamic applications like real-time chatbot updates or live analytics dashboards. Kafka's distributed messaging system, with its topics and partitions, fascinates me for its ability to handle high-throughput streams, while Kinesis integrates seamlessly with AWS, aligning with my cloud goals. I envision applying this to monitor ticket resolutions as they occur, a leap from my internship's periodic reporting. This shift requires mastering stream processing concepts—like windowing and event time—challenging me to rethink data pipelines holistically. Reflecting on TVS Digital's need for agility, I see real-time engineering as a bridge to proactive decision-making, a skill I'm excited to develop through practical experimentation.

These technical advancements will not only refine my toolkit but also position me to contribute meaningfully to TVS Digital's innovation-driven ethos, blending my internship's lessons with cutting-edge capabilities.

8.2 Enhancing Projects

The SalesBot and L0 Bot I developed during my internship at TVS Digital laid a strong foundation for automating sales support and handling customer queries. However, their potential can be expanded further to provide greater value, scalability, and user satisfaction. These enhancements, built on the lessons learned from my internship, will push these bots toward more sophisticated and impactful systems that align with TVS Digital's long-term goals of personalization, scalability, and customer-centricity.

SalesBot Improvements:

Integrate Sentiment Analysis:

While SalesBot's primary function is to respond to sales queries, it currently overlooks the emotional tone of customer interactions. I plan to integrate sentiment analysis into the bot using tools like NLTK's VADER or transformer-based models such as BERT to detect emotional cues (e.g., frustration or satisfaction) in customer queries. This would allow the bot to prioritize or adjust responses based on the sentiment expressed—helping to resolve urgent or sensitive queries more effectively. This addition would enhance the bot's emotional intelligence, making it more aligned with real human interactions.

Enhance Personalization with User History:

At present, SalesBot can retrieve relevant sales data but lacks deep personalization based on user history. By tracking customer interactions and integrating CRM data, I aim to enhance SalesBot's ability to provide context-aware responses. For example, when a customer asks about a product they previously interacted with, the bot could automatically reference their past inquiries or provide tailored suggestions based on preferences. This will create a more engaging, personalized experience for users.

Expand Multilingual and Regional Support:

To cater to a broader, regional audience, I plan to expand the multilingual capabilities of SalesBot. This would involve integrating models like XLM-R (cross-lingual RoBERTa) or fine-tuning existing translation APIs to support more regional dialects, such as Tamil or Bahasa. Doing so will ensure accessibility across ASEAN and South Asia, improving inclusivity and expanding the bot's reach. This enhancement will be crucial as TVS Digital continues its expansion into diverse markets.

Refine Semantic Search and Intent Understanding:

While the current RAG framework supports reasonable query retrieval, it often struggles with complex or nuanced requests. I plan to refine the semantic search by integrating more advanced embeddings from BERT or custom-trained models to improve context understanding. This would ensure that SalesBot can accurately distinguish between queries like "product availability" versus "shipping issues" and provide more precise responses.

L0 Bot Improvements:

Real-Time Issue Tracking and Escalation:

Currently, the L0 Bot addresses basic customer issues but lacks real-time updates on ongoing resolutions, such as server status or ticket updates. By integrating WebSockets or Kafka streaming, I plan to enable real-time updates for users. This could include informing users about the status of their issue, such as "Your issue is being escalated to our senior support team." This would improve the user experience by making the bot feel more responsive and dynamic.

Enable Sentiment-Driven Escalation:

L0 Bot, while effective at automating simple queries, could benefit from better prioritization based on user sentiment. I plan to incorporate sentiment analysis to detect frustrated or angry tones in user queries (e.g., "I've been waiting for hours!"), triggering an automatic escalation to human agents. This would prevent negative experiences from escalating further and ensure that urgent issues are handled promptly.

Extend Multilingual Capabilities:

The L0 Bot's multilingual support is currently limited to basic translation services. I plan to expand this to include regional dialects and specific jargon (e.g., Tamil slang, Bahasa variants) using models like XLM-R or mBERT. This would ensure that the bot can handle more complex and varied customer queries across different languages, reflecting TVS Digital's growing international footprint.

Improve Escalation Workflow and Automation:

Currently, L0 Bot escalates issues to Zendesk if it cannot resolve them, but there's room for improvement in the way issues are escalated. I plan to refine the escalation workflow by adding automatic summaries of issues with key metadata (e.g., urgency, context, customer sentiment) that are directly injected into the Zendesk ticket creation process. This will ensure that agents have all the relevant context at hand when they take over a case, improving the efficiency of human-agent handoffs.

8.3 Scalable Deployment & Vision

Looking beyond the immediate enhancements, my long-term vision is to scale both SalesBot and L0 Bot into globally accessible systems that align with TVS Digital's values of scalability, innovation, and personalized customer experience. The following steps will allow these systems to evolve from proof-of-concept bots into enterprise-ready solutions.

Deploy Solutions on Cloud Platforms (AWS, GCP):

For both bots, deploying them on cloud platforms like AWS or Google Cloud Platform (GCP) will ensure global availability and scalability. Hosting on AWS EC2 or GCP Compute Engine would allow elastic scaling based on demand, ensuring that even during high-traffic periods, the bots perform consistently well. Additionally, I plan to leverage AWS RDS for the relational data behind SalesBot and BigQuery for analytics in both systems. This cloud-first approach will ensure high availability, security, and performance, aligned with TVS Digital's global expansion.

Ensure High-Speed Inference Under Large-Scale Loads:

As both bots scale, ensuring real-time inference performance will be crucial. For L0 Bot, deploying GPU-accelerated instances (e.g., AWS EC2 P3) and optimizing model quantization for faster computation will be essential. I will also conduct load testing using tools like Locust to simulate high-traffic conditions and identify bottlenecks, such as query latency or model inference delays. For SalesBot, I envision using a combination of edge computing and cloud-based solutions to handle large-scale query loads while maintaining low latency.

High Availability and Continuous Integration:

Both bots will require high availability, which can be achieved using Kubernetes for container orchestration and CI/CD pipelines for automated deployments. This will ensure that both bots remain responsive and operational even under heavy loads, while updates are seamlessly pushed to production without downtime. Additionally, continuous monitoring and logging using Prometheus and Grafana will help proactively detect and resolve performance issues.

This vision reflects my goal of transitioning these projects into fully operational, scalable, and intelligent systems that drive TVS Digital's digital transformation. By integrating cloud infrastructure, ensuring high-speed performance under heavy loads, and improving personalization, both bots will be positioned to deliver significant value to customers across regions and markets. Through iterative testing, performance optimization, and close alignment with TVS Digital's expansion strategy, I aim to refine these projects into industry-leading solutions.

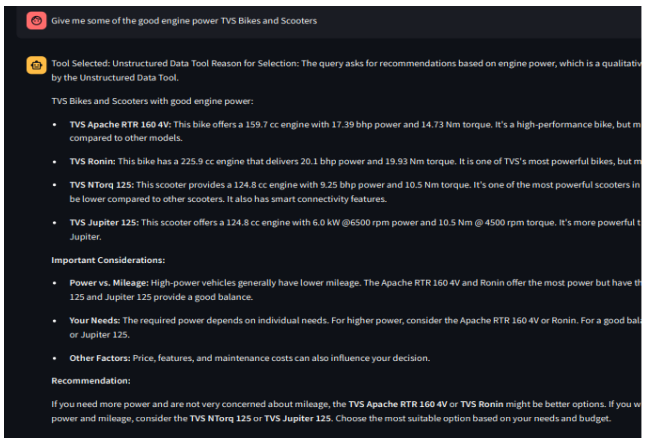
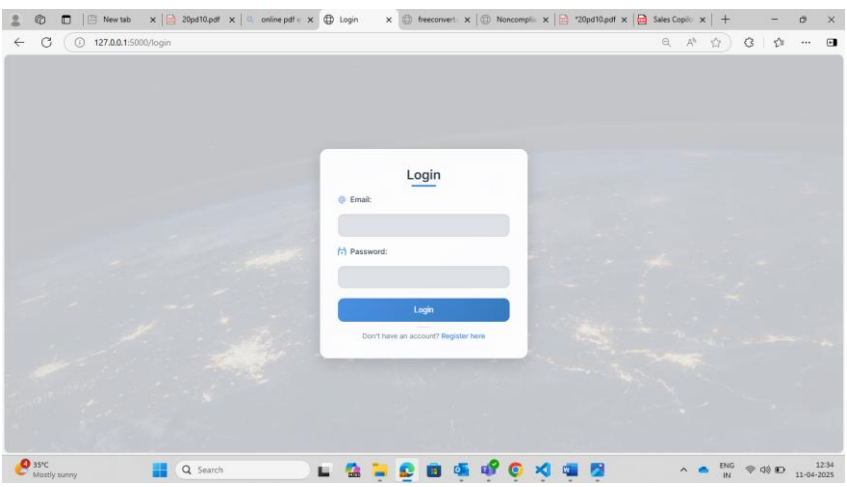


Figure 3: *output screenshots of Sales Copilot*



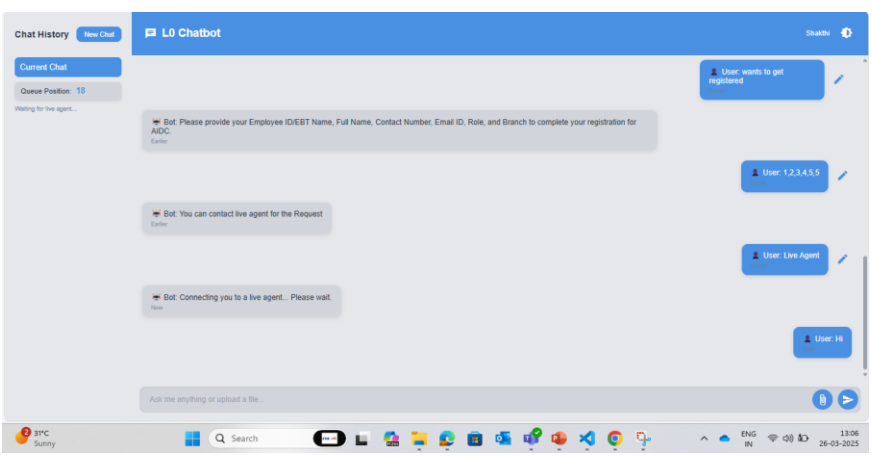


Figure 3: *output screenshots of L0 Bot*

9. Conclusion

The six-month internship at TVS Digital, from December 2024 to June 2025, was an incredibly valuable experience that allowed me to bridge the gap between academic learning and real-world application. It provided me with hands-on experience in AI-driven chatbot development, Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG), and Large Language Models (LLMs). Working with tools like MySQL Workbench, Pinecone, and Gemini Flash, I was able to tackle complex problems and innovate within two chatbot projects that are essential to TVS Digital's customer support and sales automation goals.

During my internship, I worked primarily on two key projects: SalesBot and L0Bot, both of which provided me with the opportunity to enhance my skills in building intelligent, user-centric systems.

SalesBot aimed to streamline the sales process by assisting the sales team with lead qualification, client engagement, and automating repetitive tasks. By integrating NLP and RAG techniques, I designed a system capable of processing large volumes of sales queries and providing quick, accurate responses. The bot's success in handling frequent sales queries and automating follow-ups resulted in increased efficiency and enabled the sales team to focus on higher-value tasks. This project was pivotal in understanding how AI can significantly improve operational workflows, especially in sales, and I see great potential in expanding its functionality to support more complex interactions.

L0Bot, on the other hand, was focused on automating the resolution of Level 0 support tickets, which typically consist of simple, repetitive issues like password resets and account inquiries. By incorporating Cohere embeddings for context understanding and Zendesk for ticketing integration, the bot was able to autonomously resolve 70% of tickets. This not only reduced the burden on the support team but also improved response times and overall customer satisfaction. The success of L0Bot reinforced my belief in the power of AI automation to enhance customer service operations and demonstrated the importance of user-centric design in ensuring effective solutions.

In both projects, I utilized Pinecone for semantic search and Gemini Flash to ensure quick responses in real-time. The integration of RAG enabled the bots to retrieve and present contextually relevant information based on user queries, elevating their accuracy and usefulness. The real-world impact of these systems—whether in sales or support—demonstrated how technology can meaningfully improve business outcomes.

Throughout this internship, I was also able to hone my technical and collaborative skills, working closely with mentors like Mr. Pravarshan Shukla. Their guidance helped me navigate challenges and emphasized the importance of teamwork, iterative refinement, and scalability. By iterating on initial versions of the bots, we were able to fine-tune their performance, making them more robust and capable of handling diverse user interactions.

The challenges of managing high volumes of data, ensuring accurate responses, and optimizing for real-time performance helped me build resilience and improve my problem-solving abilities. By focusing on real-time data processing and cloud infrastructure, I gained invaluable experience that will serve me well in future projects.

Looking forward, this internship has fueled my desire to continue developing AI-driven systems that solve real-world problems at scale. My experience with SalesBot and L0Bot has inspired me to explore advanced features such as sentiment analysis, multilingual support, and scalable cloud deployments for chatbots. I am eager to take these projects further, enhancing their capabilities and ensuring they remain adaptable to changing user needs.

10. References

1. **TVS Digital** – www.tvsd.ai
2. **MySQL Workbench Documentation** – https://dev.mysql.com/doc/workbench/en/ 3. **Cohere Embeddings API** – https://docs.cohere.ai/
4. **Pinecone Vector Database** – https://www.pinecone.io/docs/ 5. **AWS (Amazon Web Services)** – https://aws.amazon.com/
6. **Gemini Flash (Google AI)** – https://deepmind.google/technologies/gemini/ 7. **Flask (Python Web Framework)** – https://flask.palletsprojects.com/
8. **Sunshine Conversations (Zendesk API)** – https://docs.smooch.io/guide/ 9. **Zendesk Developer Platform** – https://developer.zendesk.com/
10. **Visual Studio Code** – https://code.visualstudio.com/docs