1. Goal of Thesis

The primary objective of this thesis is to systematically evaluate multiple combinations of LLMs and Retrieval-Augmented Generation (RAG) methodologies specifically tailored to natural language-based generation tasks encompassing:

- SQL Queries
- REST API Calls

Through structured empirical investigation, the research explicitly addresses the following critical research questions:

1. How effective are AI-powered chatbots in generating code from natural language input to generate
   a. SQL queries
   b. REST API calls
2. What impact on performance does combining SQL and API generation capabilities into one framework have?
3. Which combination of LLM and RAG technique yields the highest accuracy in structured data retrieval and process automation?

By exploring these questions, the thesis aims to provide actionable insights that can inform practical chatbot deployments and significantly advance the theoretical understanding of integrated NLP-based automation systems.

2. Way to go

To address the research questions presented in Chapter 1, this study examines combinations across three key dimensions:

- **LLMs**: Two selected models, each chosen for distinct strengths—OpenAI o4-mini, recognized for superior coding proficiency, and Google Gemini 2.5 Pro, known for its exceptionally large context window.
- **RAG frameworks**: Three specific frameworks—Standard RAG, Self-RAG, and CoRAG—each providing distinct capabilities in external knowledge retrieval and integration.
- **Documentation scenarios**: Three documentation contexts—SQL-only, API-only, and hybrid setups—to assess how different levels of contextual documentation impact performance.

The combination of these dimensions creates a structured experimental matrix with 18 unique configurations, enabling thorough comparative analysis.

Test environment: SAP Transactional Banking (SAP TRBK) System

The experiments are conducted within the SAP TRBK transactional banking system, chosen due to its inherent complexity, extensive public documentation, and broad relevance across various industries. The publicly accessible SAP TRBK OData APIs (SAP SE 2025) are fundamental to this integration, allowing realistic simulation of combined SQL and REST API interaction scenarios.

Selection of LLMs

Two LLMs were selected based on rigorous criteria detailed in Chapter 2:

- OpenAI o4-mini: Chosen for exceptional coding proficiency, achieving the highest Artificial Analysis Coding Index score of 63, making it ideal for precise coding tasks.

- Google Gemini 2.5 Pro: Selected due to its substantial context window capability (up to 1,000,000 tokens), enabling extensive integration of external documentation and context.

RAG Frameworks

The research evaluates three advanced retrieval-augmented generation frameworks:

- Standard RAG: Uses external retrieval mechanisms to enhance LLM-generated outputs by ensuring accurate grounding in relevant documentation.
- Self-RAG: Builds upon standard RAG by adding iterative retrieval coupled with internal reflection and self-assessment, dynamically adapting retrieval actions based on output quality and progress.
- CoRAG: Employs an iterative, multi-step retrieval process, facilitating complex reasoning tasks that require sequential retrieval steps and deeper integration of external knowledge.

Documentation Scenarios

The influence of contextual documentation is critically assessed through three distinct setups:

- SQL-only: Exclusive provision of database schema documentation, omitting API details.
- API-only: Inclusion of only API documentation, without database schemas.
- Hybrid: Combines SQL schemas and API documentation to test whether additional context positively, negatively, or neutrally influences performance.

This differentiation allows a practical and theoretical evaluation of documentation completeness on code generation quality.

Test Data Generation

Due to a lack of suitable public datasets for combined SQL and REST API evaluation, this research developed a custom test data generation pipeline leveraging the APIGen framework (Liu, et al. 2024). This pipeline comprises several key stages:

- System Documentation Collection: Documentation was extensively compiled from the SAP API Business Hub (SAP SE 2025). Custom scripts were developed to extract and structure data from OData API specifications, creating pseudo-database schemas that maintain semantic consistency between SQL and API contexts.
- Automated Test Case Generation: Automated scripts generated realistic test cases. These generated test cases include natural language queries paired with corresponding SQL or API commands, systematically categorized into four complexity levels (easy, medium, hard, very hard). These range from simple parameter queries to complex scenarios involving nested conditions, joins, and conditional logic.
- Humanization: Automated humanization scripts refined natural language inputs, translating technical phrasing into conversational language for realism. API test cases were validated by mock server execution, confirming correct HTTP responses. SQL queries underwent syntactic validation via the sqlparse library, ensuring syntactic accuracy.
- Validation: All generated test cases underwent additional manual review by domain experts to confirm practical relevance and accuracy.