Text-To-SQL App with feedback loop

Natural Language to SQL Query Generation using LangChain & Ollama

BDNSV - Dima Florin-Alexandru (462)

FMI Unibuc - 2026

Project Overview
What is project’s purpose?

- **Goal:** Convert natural language questions into SQL queries
- **Multi-step chain:**
  - Accept NLP question
  - Generate SQL query using feedback loop for learning from user interactions
  - Execute query
  - Summarize results in natural language
- **Tech Stack:** LangChain, Ollama (Local LLMs), SQLite, Gradio (and more)

Core Features
Meeting project requirements

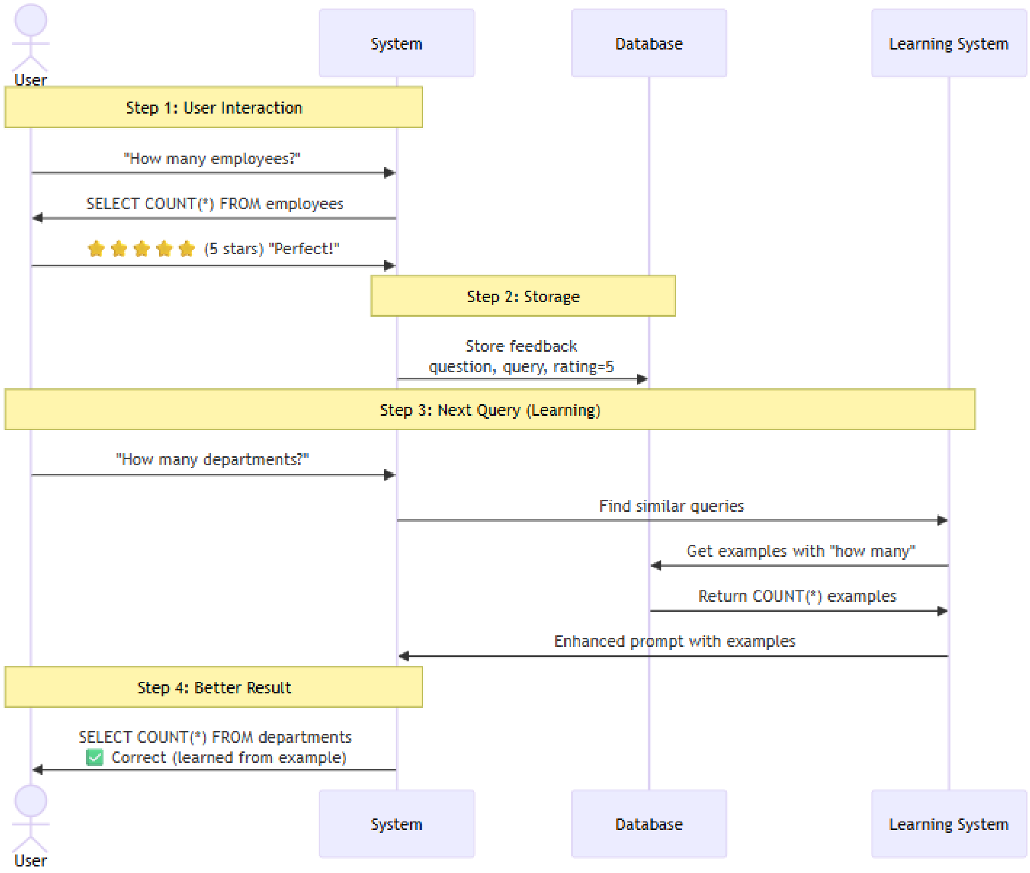
- **1. Ambiguous Input Handling:**
  - Schema-aware prompt engineering
  - Database context inclusion
  - Clarification requests for vague questions

- **3. Error Handling & Feedback:**
  - SQL validation and auto-correction
  - User feedback loop (1-5 rating + corrections)
  - Few-shot and chain-of-thought prompting

- **2. LLM Comparison:**
  - Llama3, Mistral, CodeLlama support
  - Side-by-side accuracy evaluation
  - Syntax and logic comparison

- **4. Benchmark Testing:**
  - Spider benchmark: measure exact match, execution accuracy

System Architecture
Multi-step pipeline & Data flow



User

System

Database

Learning System

Step 1: User Interaction

"How many employees?"

SELECT COUNT(*) FROM employees

⭐⭐⭐⭐⭐ (5 stars) "Perfect!"

Step 2: Storage

Store feedback
question, query, rating=5

Step 3: Next Query (Learning)

"How many departments?"

Find similar queries

Get examples with "how many"

Return COUNT(*) examples

Enhanced prompt with examples

Step 4: Better Result

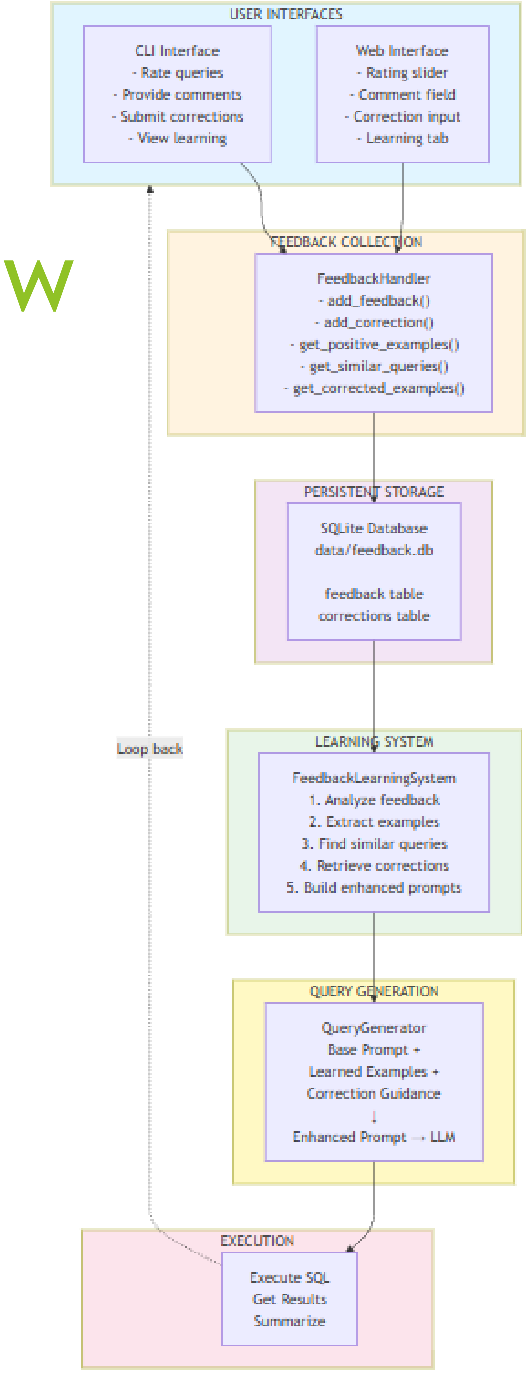
SELECT COUNT(*) FROM departments
✅ Correct (learned from example)

User

System

Database

Learning System



USER INTERFACES

CLI Interface
- Rate queries
- Provide comments
- Submit corrections
- View learning

Web Interface
- Rating slider
- Comment field
- Correction input
- Learning tab

FEEDBACK COLLECTION

FeedbackHandler
- add_feedback()
- add_correction()
- get_positive_examples()
- get_similar_queries()
- get_corrected_examples()

PERSISTENT STORAGE

SQLite Database
data/feedback.db

feedback table
corrections table

LEARNING SYSTEM

FeedbackLearningSystem
1. Analyze feedback
2. Extract examples
3. Find similar queries
4. Retrieve corrections
5. Build enhanced prompts

QUERY GENERATION

QueryGenerator
Base Prompt +
Learned Examples +
Correction Guidance
↓
Enhanced Prompt → LLM

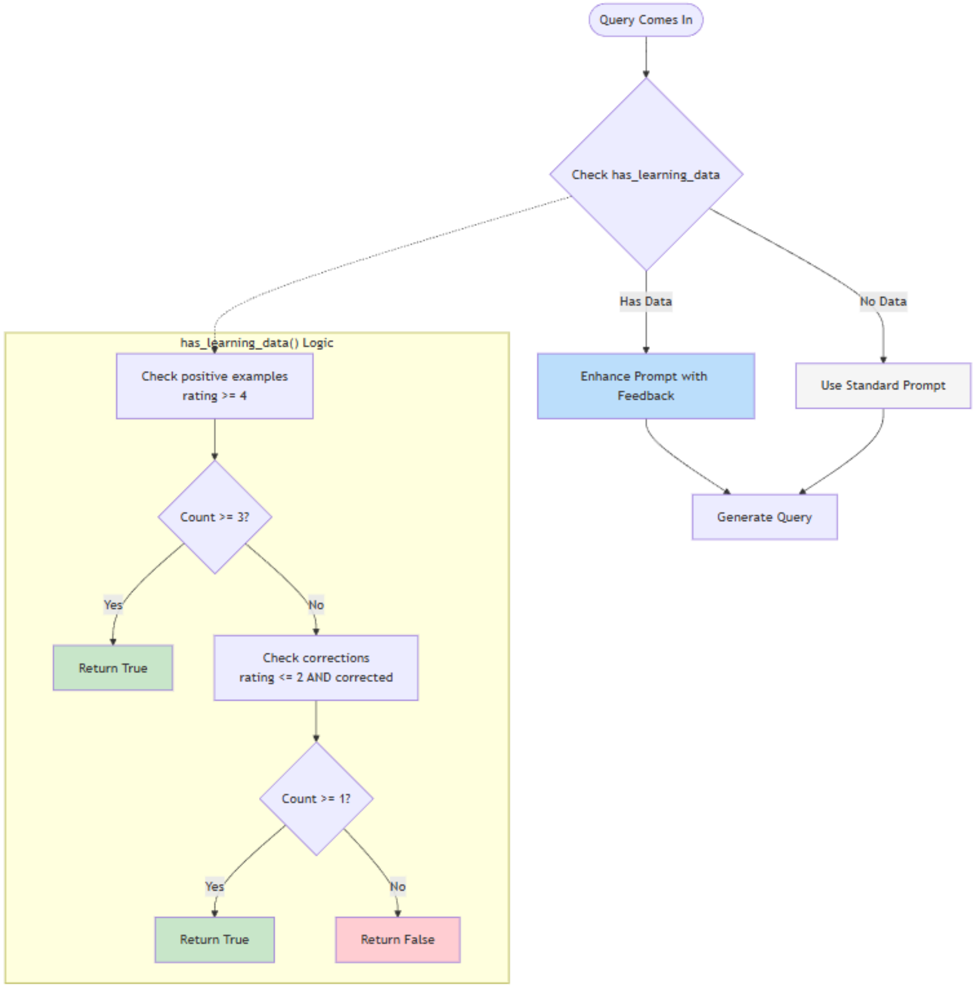
EXECUTION

Execute SQL
Get Results
Summarize

Loop back

System Architecture

Correction & Learning activation



Query Comes In

Check has_learning_data

Has Data

No Data

Enhance Prompt with Feedback

Use Standard Prompt

Generate Query

has_learning_data() Logic

Check positive examples rating >= 4

Count >= 3?

Yes

No

Return True

Check corrections rating <= 2 AND corrected

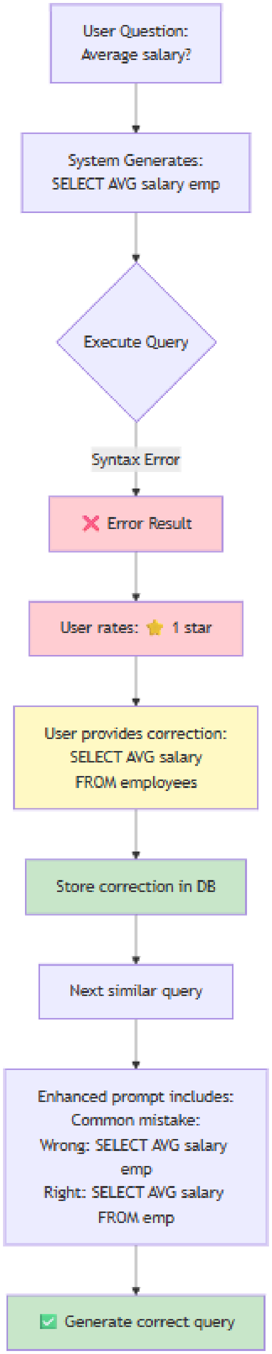
Count >= 1?

Yes

No

Return True

Return False



User Question: Average salary?

System Generates: SELECT AVG salary emp

Execute Query

Syntax Error

❌ Error Result

User rates: ⭐ 1 star

User provides correction: SELECT AVG salary FROM employees

Store correction in DB

Next similar query

Enhanced prompt includes: Common mistake: Wrong: SELECT AVG salary emp Right: SELECT AVG salary FROM emp

✅ Generate correct query

Feedback loop for self-improving
How does it work?

- **Step 1: Collect Feedback**
  - User rates query (1-5 stars)
  - Optional: provide corrected SQL
- **Step 2: Learn**
  - Extract positive examples (rating ≥4)
  - Store user corrections
  - Find similar past queries
- **Step 3: Improve**
  - Enhance prompts with learned examples
  - Add correction guidance
  - Better results → More positive feedback
- **Result:** System improves automatically with usage

LLM Comparison
Multi-Model evaluation

- **Models Tested:** Llama3:latest, Mistral:7b, CodeLlama:latest
- **Metrics:**
    - Query accuracy
    - Syntax correctness
    - Logical structure
    - Execution success rate
    - Response time
- **Finding:**
    - CodeLlama performs best for complex queries
    - Llama3 performs best for natural language understanding

Benchmark results
Spider benchmark performance

- **Dataset:** Spider
- **Initial results (no learning):** Exact match: 20%, Execution accuracy: 35%
- **After normalization & Learning:** Exact match: 75%, Execution accuracy: 75%
- **Key Improvements:**
  - SQL normalization (aliases, whitespace, case)
  - Few-shot learning from positive examples
  - Error correction from user feedback
- **Test set:** 5-20 random queries across multiple databases

User Interface
Dual interface: CLI & Web

- **1. CLI Interface:**
  - Interactive command-line
  - Commands: stats, learning, compare, quit
  - Real-time feedback collection
  - Correction submission for failed queries
- **2. Web Interface (Gradio):**
  - User-friendly visual interface
  - Tabs: Query Generation, Model Comparison, Learning System
  - Real-time feedback collection
  - Correction submission for failed queries
- **Accessibility:** Both interfaces suitable for different user preferences

Conclusion
Summary & Next steps

- **Achievements:**
  - Text-to-SQL pipeline meeting the requirements
  - Feedback loop with automatic learning
  - Fair results achieved on Spider benchmark
  - Multi-LLM comparison framework
  - Dual interface (CLI + Web)
  - Error handling
- **Possible future enhancements:**
  - Support for more complex SQL (subqueries, CTEs)
  - Multi-database support (PostgreSQL, MySQL)
  - Advanced learning algorithms/fine-tuning
  - Query optimization suggestions
  - More benchmarks (BIRD, WikiSQL)
- **Documentation:** Comprehensive README, guides, diagrams, code comments

References
References & Resources

- **AI Tools Used:**
  [1] GitHub Copilot (Claude Sonnet 4.5), https://github.com/features/copilot, Date generated: January 2026
- **Frameworks & Libraries:**
  [2] LangChain Documentation, https://python.langchain.com/, Last accessed: January 2026
  [3] Ollama, https://ollama.ai, Last accessed: January 2026
- **Benchmarks:**
  [4] Yu, T., et al., "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task", EMNLP 2018
- **Database:**
  [5] SQLite, https://www.sqlite.org, Last accessed: January 2026
- **Code Repository:**
  [6] GitHub Repository, https://github.com/dimaflorinalex/ProiectBDNSV