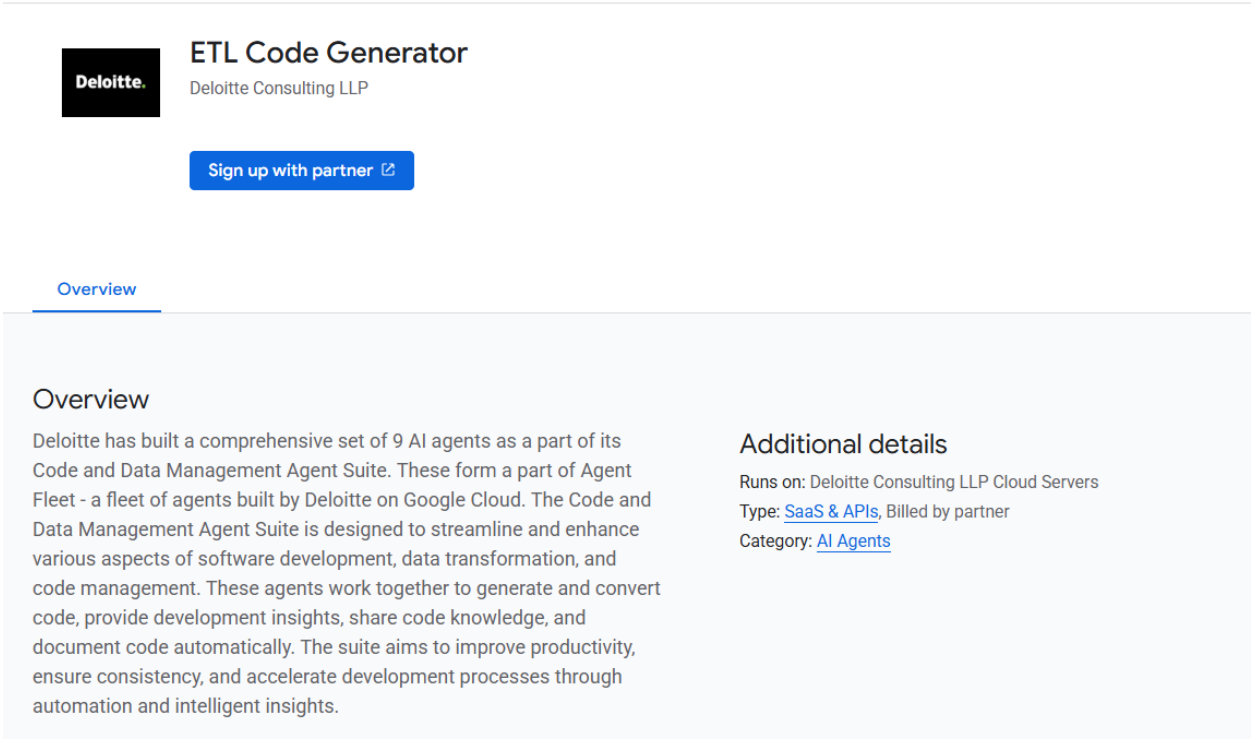
Any to Any – ETL

Dhanush D - 81382



Abstract

In the era of heterogeneous data systems, data interoperability and seamless migration have become critical challenges for organizations. This paper presents "AnytoAny-ETL," a smart Extract, Transform, Load (ETL) framework powered by Generative AI that enables the transfer of data across various database systems. Leveraging the capabilities of Google Gemini and LangGraph, the proposed system simplifies ETL development using natural language instructions and provides an interactive user interface for customizable data transformation. The architecture supports relational (PostgreSQL, MySQL, MSSQL, SQLite) and NoSQL (MongoDB) databases, addressing schema alignment, transformation automation, and script generation in a streamlined workflow. This solution empowers data professionals to build and execute ETL pipelines without writing manual code, reducing time and increasing reliability.

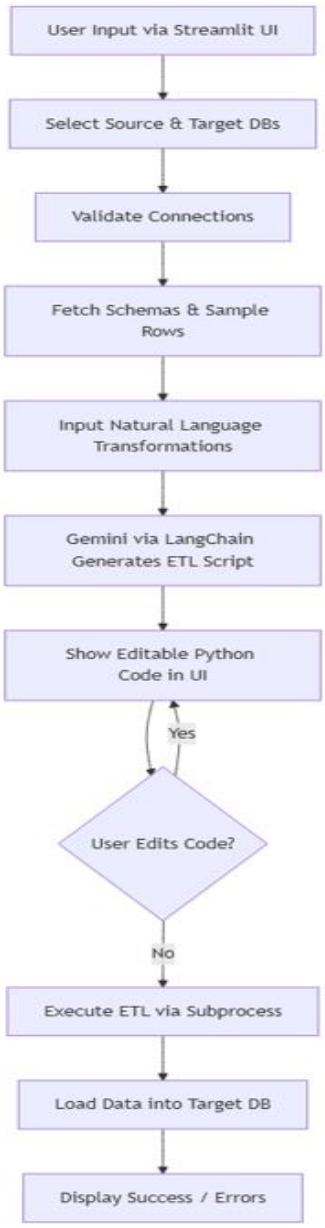
1. Introduction

ETL processes are foundational in data warehousing, integration, and analysis. Traditional ETL tools demand substantial coding effort and domain expertise, especially when handling diverse database ecosystems. With advancements in Large Language Models (LLMs), there exists an opportunity to revolutionize ETL automation through natural language interaction. AnytoAny-ETL utilizes generative AI and agentic workflows to simplify the data pipeline lifecycle from validation to execution.

This project is inspired by the "ETL Code Generator" solution published by Deloitte Consulting LLP on the Google Cloud Marketplace. Deloitte's solution is a part of its broader Agent Fleet suite — a collection of 9 AI agents designed to optimize software development and data workflows. These agents provide intelligent automation for generating, converting, and managing code across enterprise systems. The managed service model and SaaS integration of Deloitte's solution influenced the architecture and modular agentic design used in AnytoAny-ETL.

2. System Overview

The system is structured into modular components integrated through an intuitive web interface built with Streamlit. Users select source and target databases, input credentials, describe transformations in natural language, and receive a full executable Python script tailored for their requirements.



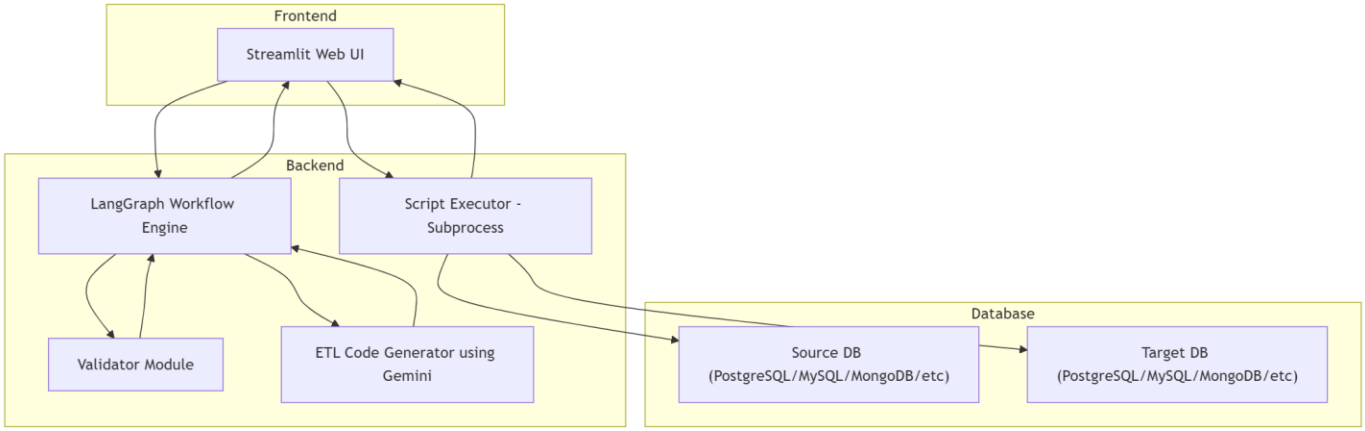
3. Key Features

- **Cross-Database ETL:** Supports PostgreSQL, MySQL, MSSQL, MongoDB, and SQLite.
- **AI-Powered Script Generation:** Utilizes Google Gemini through LangChain to interpret user-described transformations and generate ETL scripts.
- **Schema Preview:** Automatically fetches and displays source/target schemas with sample rows.
- **Editable Code Interface:** Users can review and edit the generated Python code before execution.
- **Execution Engine:** Scripts are executed via Python subprocess using temporary files, ensuring isolation.
- **LangGraph Workflow:** Incorporates agent-based state management to modularize the ETL steps.

4. Technical Architecture

The solution architecture comprises the following layers:

- **Frontend:** Streamlit-based UI for user interactions.
- **AI Layer:** Gemini (via LangChain) for prompt-based ETL script generation.
- **Workflow Layer:** LangGraph handles state transitions such as validation, code generation, editing, and execution.
- **Execution Layer:** Python subprocess module runs generated scripts in isolated environments.



5. Technology Stack

Component	Technology
UI	Streamlit
Code Generation	Google Gemini (LangChain)
Workflow Engine	LangGraph
Databases	PostgreSQL, MySQL, MSSQL, MongoDB, SQLite
Execution	Python subprocess, tempfile
Additional Tools	dotenv, pandas, sqlalchemy

6. Dependencies

Extracted from the `requirements.txt` file:

```
streamlit==1.22.0
langchain==0.0.175
```

```
python-dotenv==1.0.0
psycopg2-binary==2.9.5
pymongo==4.5.0
pyodbc==4.0.35
sqlalchemy==2.0.14
pymysql==1.0.3
sqlite3==3.36.0
```

System Package:

```
sudo apt-get install unixodbc-dev
```

7. Module Breakdown

- **app.py / etl_app.py:** Streamlit UIs.
- **graphETL.py:** LangGraph agent-based ETL workflow.
- **spare.py:** Prompt-driven standalone generator.
- **modules/generator.py:** Generates ETL script via Gemini.
- **modules/executor.py:** Executes generated Python scripts.
- **modules/ui.py:** Renders input fields, schema, and code editors.
- **modules/validator.py:** Validates DB connections and extracts schema.

8. Conclusion

AnytoAny-ETL bridges the complexity gap in cross-database ETL processes by leveraging generative AI and modular design. The framework is extensible, user-friendly, and significantly reduces the manual effort in constructing ETL workflows. It presents a novel

direction for low-code/no-code data engineering pipelines and can be further expanded to include scheduling, logging, and monitoring features in future iterations.

Keywords: ETL, LangChain, Streamlit, Google Gemini, LangGraph, Data Transformation, Database Migration, Low-Code, AI Workflow, Deloitte Agent Fleet