DataForge

Automated Data Warehouse Schema
Generatorwith AI-Driven Enhancements

Faculty of Computer and Information Sciences - Ain Shams University
(Information Systems Department)

Our Team

- Abdelrahman Abdelnasser Gamal Mohamed
- Abdelrahman Adel Atta Mohamed
- Ahmed Reda
- Ahmed Mahmoud
- Arwa Elsharawy
- Alaa Emad

Under Supervision of:

Dr. Yasmine Afify

TA. Yasmine Shabaan

Table of Contents

Introduction

In today's data-driven landscape, organizations increasingly rely on data warehouses to consolidate and analyse vast amounts of information from disparate sources. A well-designed data warehouse schema is crucial for efficient data storage, retrieval, and analysis, enabling businesses to make informed decisions. However, designing such schemas manually can be time-consuming and error-prone, especially as data complexity grows. This project presents DataForge, a tool designed to automate the creation of data warehouse schemas. Leveraging both backend processing and an interactive frontend interface, DataForge simplifies schema design, enhances accuracy, and integrates AI-driven suggestions to optimize data structures. Additionally, DataForge empowers users to refine and customize generated schemas, ensuring they align perfectly with specific business requirements.

Problem Definition

Designing data warehouse schemas involves identifying and organizing fact and dimension tables, defining primary and foreign keys, and ensuring consistency across various data sources. Manual schema generation poses several challenges:

- Time-Consuming Process
- Human Error
- Scalability Issues
- Lack of Optimization
- Limited Flexibility

Time-Consuming Process: Manually parsing SQL files and designing schemas can be labour-intensive, delaying data integration projects.

Human Error: Mistakes in identifying relationships or defining keys can lead to inefficient queries and unreliable data insights.

Scalability Issues: As data volumes and complexity increase, maintaining and updating schemas manually becomes impractical.

Lack of Optimization: Without automated tools, optimizing schemas for performance and scalability is challenging, potentially leading to suboptimal data structures.

Limited Flexibility: Manual processes may not easily accommodate dynamic changes, or specific customization needs from different business domains.

Objective

The primary objective of the **DataForge** project is to develop an automated tool that streamlines the creation of data warehouse schemas. Specifically, the project aims to:

1. Automate Schema Parsing

Efficiently parse SQL files to extract table definitions, columns, and relationships.

3. Generate Fact & Dimension Tables

Automatically identify and create fact tables.

5. Enable User Customization

Allow users to edit and modify the generated schemas post-generation to accommodate specific business needs and preferences.

2. Ensure Scalability and Accuracy

Handle large and complex databases while maintaining high accuracy in schema generation and optimization.

4. Enhance Schemas with AI

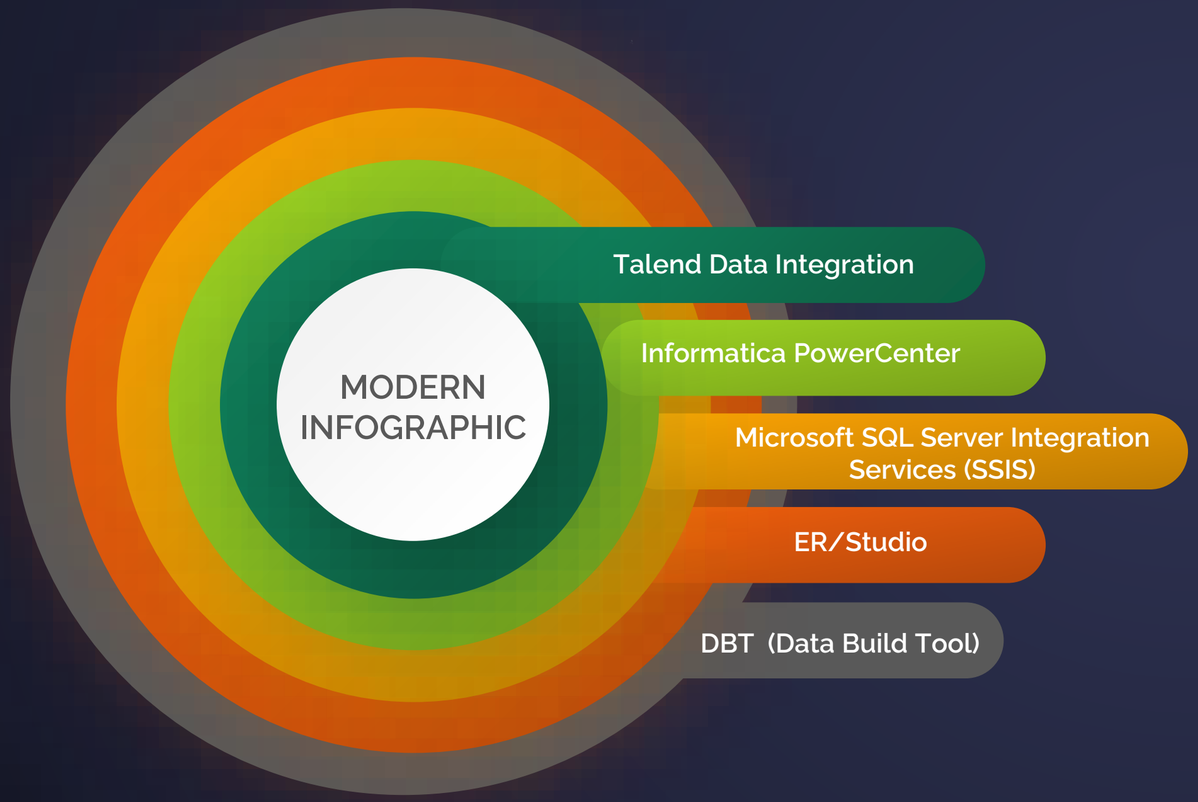
Utilize AI services to detect domains, missing tables or columns, and generate enhanced schemas.

6. Provide Interactive Visualization

Offer a user-friendly interface that visualizes the schemas, allowing users to interact with the schema structure.

Related Works

Several tools and frameworks exist for data warehouse schema design and automation



01 A comprehensive data integration platform that offers tools for ETL (Extract, Transform, Load) processes, including schema mapping and data transformation.

02 An enterprise data integration solution that provides features for data profiling, mapping, and transformation, aiding in schema design.

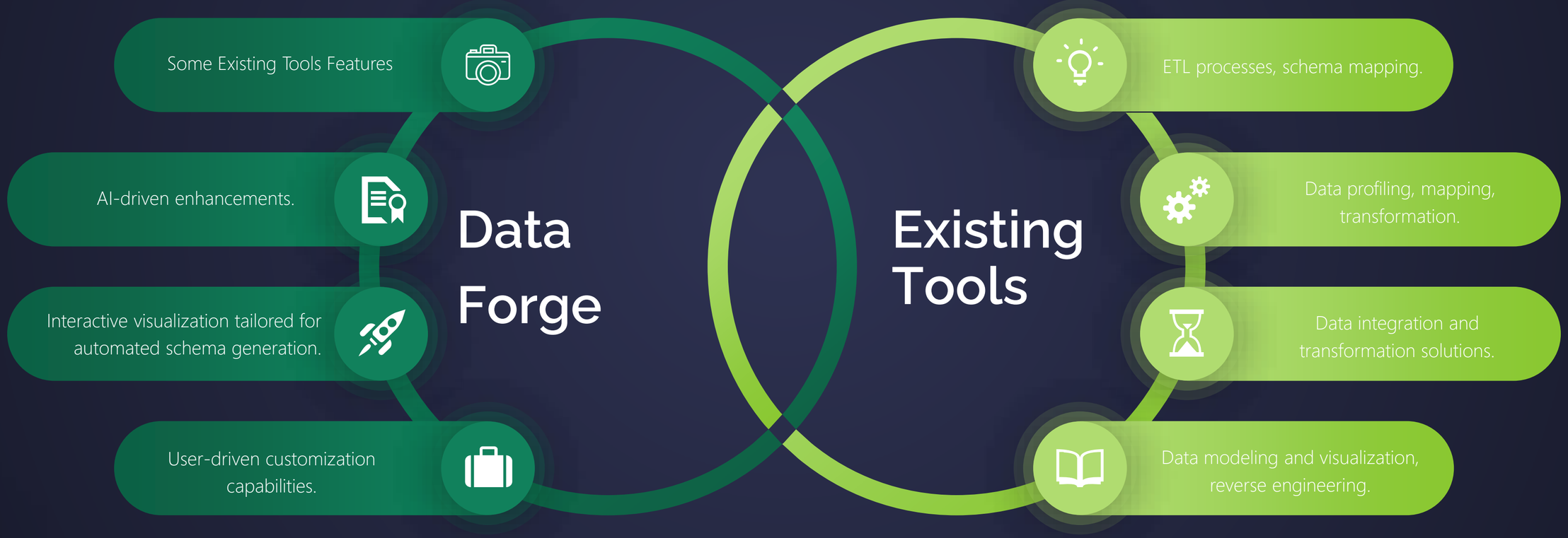
03 A platform for building enterprise-level data integration and transformation solutions, including schema generation tools.

04 A data modelling tool that supports the design and visualization of complex data warehouse schemas, offering features like reverse engineering from existing databases.

05 A command-line tool that enables data analysts and engineers to transform data in their warehouse more effectively by managing data models and transformations.

Related Works

Fast Comparison



System Architecture

Modular Design

Frontend:
React-based interactive user interface with components for schema visualization and editing.

Backend:
Django powered by Django REST Framework (DRF) for handling requests, parsing, and schema generation.

Components

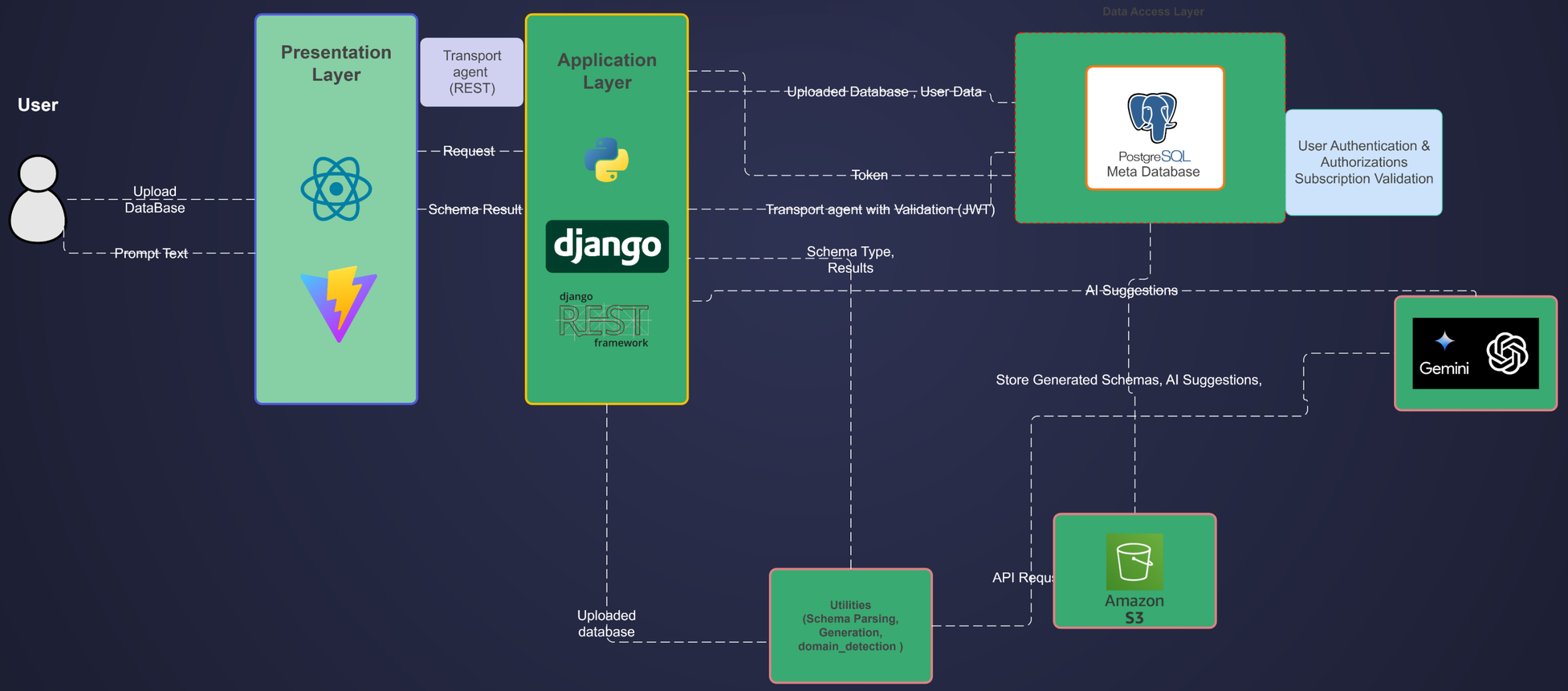
Frontend:
SchemaGraph.jsx: interactive schema graphs.
SchemaResult.jsx: results after schema generation.
SchemaEditor.jsx: Allows users to edit generated schemas.

Backend:
Models: Represent schema data and user inputs.
Views: Handle API requests for schema uploads, processing, and returning results.
Serializers: Convert data between JSON and database.
Utilities: Handle parsing, schema generation, and domain detection.
AI Services: Utilize AI for schema enhancements and suggestions (Gemini API).

Data Flow

Upload: Users upload SQL database files through the frontend interface.
Parsing & Generation: Backend processes uploaded SQL files, parses the schemas, and generates the results.
AI Enhancements: AI services analyze and optimize the generated schemas, providing suggestions for improvements or missing elements.
Storage: Generated schemas and AI suggestions are stored in a database or external storage (Amazon S3) for future retrieval.
Visualization & Editing: The frontend visualizes schemas through interactive graphs and allows users to edit or customize the generated schemas.

System Architecture Diagram



Functional Requirements

01 User Authentication

Account creation, login, secure session management.

02 Schema Upload & Parsing

Extract tables, columns, data types, keys, and Extract tables, columns, data types, keys.

03 Schema Generation & Visualization

Categorize tables, define primary and foreign keys and Interactive graphs distinguishing fact and dimension tables.

04 AI-Driven Enhance& Prompt

Request additional AI suggestions or optimizations & Domain detection, suggest missing elements, optimize schemas.

05 Schema Editing & Error Handling

Modify generated schemas: add/remove/alter tables and columns, Meaningful error messages and feedback.

06 Responsive Design & Performance

Accessible across devices and screen sizes and Efficient handling of large and complex schemas.

07 Accuracy & Efficiency

Continuously track schema generation accuracy and optimize performance through real-time feedback and AI corrections.

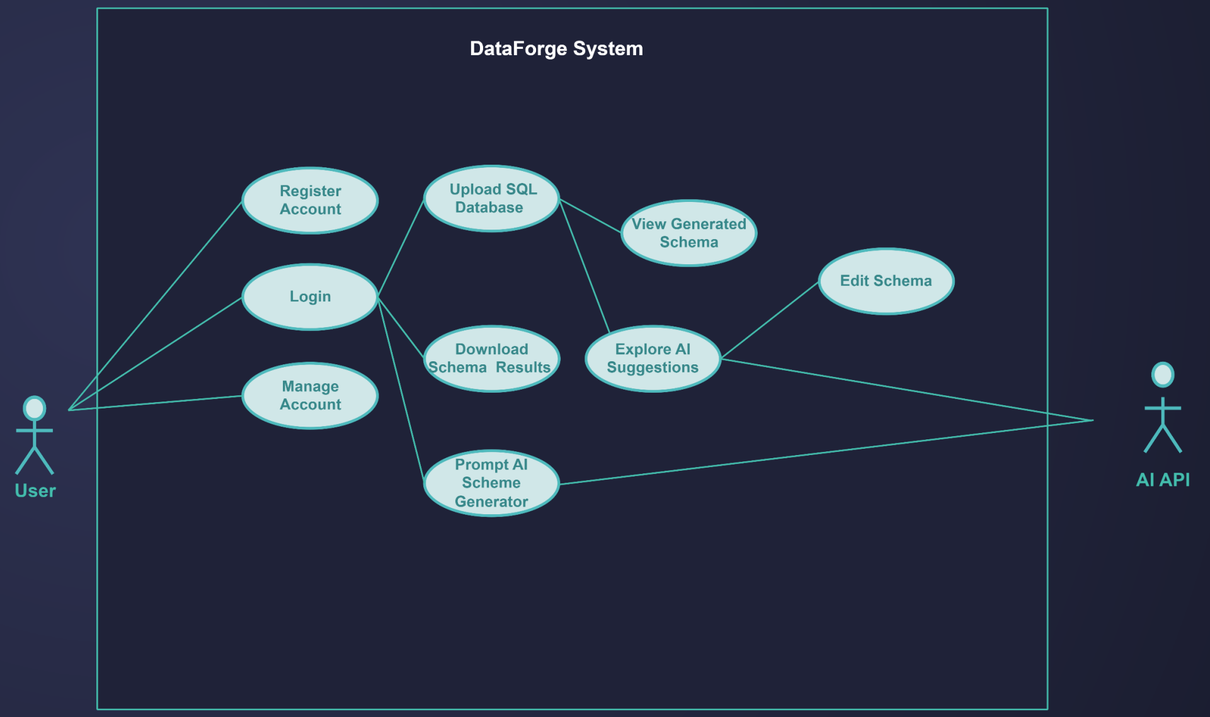
Use Case Diagram

Actors:

User
AI API (Gemini)

Use Cases:

- Register Account
- Login
- Upload SQL Schema
- View Generated Schema
- Explore AI Suggestions
- Edit Schema
- Prompt AI for Enhancements
- Download Schema Report
- Manage Account



Software & Hardware Tools

Software Tools

Frontend:

React, ReactFlow, Framer Motion, Axios, Tailwind CSS

Backend:

Django, Django REST Framework (DRF), Python 3.12

Database:

PostgreSQL

AI Services:

OpenAI API , Gemini API

Development Tools:

Visual Studio Code, Git, Postman

Deployment:

Docker, AWS

Hardware Tools

Development Machines:

PersonalComputers with sufficient processing power.

Servers:

Cloud Servers (AWS EC2)

Storage:

Cloud Storage Services (AWS S3)

Timeline Section

Weeks	Phase	Tasks
1-2	Project Planning & Requirement Analysis	Define scope, gather requirements, design system architecture, select tools.
3-4	Backend Development - Parsing & Generation	Implement SQL parsing, develop schema generation logic, set up Django models and APIs.
5-6	AI Integration & Enhancements	Integrate AI services, develop suggestions for missing elements, incorporate AI into schema.
7-8	Frontend Development - Visualization & Editing	Develop React components, implement ReactFlow for visualization, integrate SchemaEditor.jsx.
9	Testing & Quality Assurance	Unit testing, frontend testing, implement error handling, validate data integrity.
10	Deployment & Documentation	Containerize with Docker, deploy to cloud, prepare comprehensive documentation.
11	User Feedback & Iteration	Conduct user testing, gather feedback, address issues, optimize performance.
12	Final Review & Presentation Preparation	Finalize components, prepare presentation materials, conduct final review.

1-2 Weeks
Project Planning & Requirement Analysis

3-4 Weeks
Backend Development

5-6 Weeks
AI Integration & Enhancements

7-8 Weeks
Frontend Development

9 Weeks
Testing & Quality Assurance

10 Weeks
User Feedback & Iteration

11-12 Weeks
User Feedback & Final Review

Conclusion

Summary

DataForge automates and optimizes data warehouse schema generation.

Combines backend processing, AI enhancements, and interactive frontend visualization.

Empowers users with customization capabilities for tailored schema designs.

Benefits

Reduces manual effort and errors.

Enhances scalability and performance of data warehouses.

Provides AI-driven insights and optimizations.

Thanks