Scope International Journal of Science, Humanities, Management and Technology (SIJSHMT) - ISSN : 2455-068X
Vol.11 Issue 4 (2025) 50 - 57. Submitted 25/10/2025. Published 24/11/2025

Smart Data Mapping Assistant

Muppisetty Sreelekha
2nd Year, M.S Data Science
Exafluence Education
Sri Venkateswara University
Tirupati, India
sreelekharoyalmuppisetty@gmail.com

Vijayalakshmi Kumba
Professor, Department of Computer Science
SVU College of CM & CS
Sri Venkateswara University
Tirupati, India
Vijayalakshmi4k@gmail.com

Abstract
In today's data-driven environment, integrating information from diverse datasets is a critical yet time-consuming process. Manual schema mapping between heterogeneous data sources often leads to errors, inconsistencies, and productivity loss. The Smart Data Mapping Assistant addresses this challenge by automating schema mapping through metadata analysis. The system extracts and compares column names, data types, and structural attributes from source and target datasets to suggest accurate mappings. Developed using FastAPI for backend processing, MongoDB for metadata storage, and HTML/CSS with Jinja2 templates for the user interface, the tool allows users to upload files, review mapping suggestions, and store validated results efficiently. By relying exclusively on metadata rather than raw data, the system ensures privacy, reduces processing overhead, and maintains operational efficiency. This approach improves mapping accuracy, reduces human effort, and promotes consistency across multiple datasets. The Smart Data Mapping Assistant provides a scalable, efficient, and user-friendly solution for modern data integration workflows.
Keywords— Smart Data Mapping Assistant, Schema Mapping, Metadata Analysis, FastAPI, MongoDB, Jinja2 Templates, Data Integration, Automation.

I. INTRODUCTION

In the era of digital transformation, organizations increasingly rely on data originating from disparate systems such as relational databases, spreadsheets, enterprise applications, and cloud platforms. These datasets often differ in schema structure, formatting conventions, and semantic definitions, creating substantial challenges during data integration and migration. Schema mapping—identifying equivalent or related fields between two datasets—is essential for ensuring consistency, accuracy, and interoperability across systems. However, traditional schema mapping practices remain predominantly manual, making them time-consuming, labour-intensive, and prone to human error, particularly when working with large or complex datasets.

Manual schema alignment can slow down critical processes such as data migration, analytics modernization, and system upgrades. Errors in mapping may propagate downstream, resulting in inaccurate analytics, poor data quality, and additional rework. As organizations

adopt increasingly complex data ecosystems, automated, intelligent solutions for schema mapping have become essential for improving efficiency, scalability, and data governance.

The Smart Data Mapping Assistant was developed to address these challenges by automating schema mapping through metadata-driven analysis. Instead of processing or exposing raw data, the system focuses solely on metadata—column names, data types, formats, and structural information—ensuring privacy, reducing computational overhead, and increasing mapping reliability. The assistant analyzes both source and target datasets, identifies structural and linguistic similarities, and generates intelligent mapping suggestions that users can review and validate through an interactive web interface.

The system is implemented using FastAPI, chosen for its high performance, asynchronous capabilities, and developer-friendly design. MongoDB serves as the storage layer for extracted metadata, mapping history, and validation logs, enabling flexible schema representation. The interface developed using HTML, CSS, and Jinja2 templates, provides a streamlined workflow that supports dataset upload, real-time metadata extraction, and easy review of mapping suggestions. The modular architecture ensures flexibility, maintainability, and scalability across diverse deployment environments.

By automating repetitive and error-prone tasks, the Smart Data Mapping Assistant significantly reduces manual effort, improves schema alignment accuracy, and accelerates end-to-end integration workflows. It offers a practical, metadata-centric solution that enhances data consistency across heterogeneous systems while maintaining transparency and user oversight. The system represents a meaningful step toward intelligent data integration, combining automation, interpretability, and user-driven validation into a unified framework.

Ⅱ. Ease of Use

The Smart Data Mapping Assistant was designed with a strong emphasis on accessibility, simplicity, and seamless user interaction. The interface, built using HTML, CSS, and Jinja2 templates, provides a clean and intuitive workflow that guides users through each stage of metadata-based schema mapping. Users can upload source and target datasets, view automatically extracted metadata, and review suggested mappings without requiring advanced technical expertise. Each step is clearly structured, ensuring that even first-time users can navigate the system effortlessly.

FastAPI powers the backend operations and ensures rapid communication between the frontend and the MongoDB database. Its asynchronous architecture enables high responsiveness, allowing metadata extraction and mapping generation to occur within seconds for small and medium-sized datasets. Users can upload Excel or CSV files and immediately view extracted metadata and mapping suggestions. Built-in validation provides feedback on unsupported formats, missing files, or inconsistent metadata, reducing error rates and enhancing usability.

A usability evaluation conducted with student developers and data practitioners highlighted the system's ease of navigation, responsiveness, and minimal learning curve. Participants noted that the structured interface, clear layout, and ability to modify mappings directly on the page significantly reduced manual effort. Overall, the combination of a user-centric design, efficient backend processing, and automated metadata-driven mapping makes the Smart Data Mapping Assistant a practical and accessible tool for streamlining schema alignment in real-world data integration projects.

Ⅲ. Literature Review

A. Schema Mapping and Data Integration

Schema mapping plays a critical role in achieving database interoperability by enabling data to be transferred or transformed across systems with differing schemas. Early research, such as Batini et al. (1986), identified the challenges associated with heterogeneous database integration, emphasizing difficulties caused by inconsistent schema semantics and naming conventions. While traditional schema mapping relies heavily on manual inspection, the growing scale and complexity of modern datasets have increased the need for automated, reliable approaches.

B. Metadata-Driven Mapping Techniques

Metadata has emerged as a powerful abstraction for schema alignment, offering a privacy-preserving approach by focusing on dataset descriptors rather than raw values. According to a 2024 publication in Towards Data Science, metadata-based mapping improves accuracy by comparing structural features such as column names, data types, and formatting patterns. This paradigm forms the basis of the Smart Data Mapping Assistant, where metadata similarity metrics drive the generation of mapping suggestions without exposing sensitive data.

C. Automation Using AI and Rule-Based Approaches

Automation in schema mapping has evolved primarily through rule-based systems and AI-driven approaches. Rule-based techniques rely on heuristics such as string similarity, type compatibility, and pattern matching but often struggle with semantic variations and business-specific terminology. Recent advancements in AI have enabled large language models (LLMs) like GPT and Gemini to understand contextual meaning, improving the accuracy of mapping suggestions. The Smart Data Mapping Assistant integrates both methods by using rule-based logic for deterministic matching and AI models (via the OpenAI API) for semantic interpretation when needed.

D. Technologies Supporting Smart Mapping Systems

Modern technologies such as FastAPI, MongoDB, and Jinja2 play an essential role in developing scalable and efficient schema mapping platforms. FastAPI provides high-performance asynchronous request handling suitable for real-time operations. MongoDB's schema flexibility simplifies storage of diverse metadata structures and mapping histories. Jinja2 templates enable dynamic rendering of metadata and mapping results, giving users immediate visibility and control. These technologies combined support the development of an interactive, reliable, and modular system for automated data mapping.

Ⅳ. System Architecture

The Smart Data Mapping Assistant adopts a modular three-tier architecture designed for efficiency, maintainability, and scalability. The system consists of the frontend interface, backend processing layer, and database layer, each responsible for integral components of metadata-driven schema mapping.

A. Frontend

The frontend, developed using HTML, CSS, and Jinja2 templates, serves as the primary user interaction layer. It enables users to upload source and target datasets, view extracted metadata, and examine or modify suggested mappings. The interface is structured to minimize complexity, offering dedicated pages for login authentication, file upload, schema visualization, and mapping confirmation. Dynamic rendering ensures that the interface remains responsive and capable of handling large metadata files without performance degradation.

B. Backend

The backend, implemented using Python's FastAPI framework, functions as the system's processing engine. It manages dataset upload requests, performs metadata extraction,

Scope International Journal of Science, Humanities, Management and Technology (SIJSHMT) - ISSN : 2455-068X
Vol.11 Issue 4 (2025) 50 - 57. Submitted 25/10/2025. Published 24/11/2025

validates input formats, and executes schema comparison algorithms. FastAPI's asynchronous capabilities support concurrent user interactions, ensuring efficient utilization of system resources. Backend logic generates structured metadata objects and mapping suggestions, which are passed to the frontend for visualization and user validation.

C. Database

The database layer uses MongoDB, a document-oriented NoSQL database that offers flexibility in handling diverse metadata structures. MongoDB stores extracted metadata, user-uploaded dataset details, mapping histories, and validation logs. Its dynamic schema support allows variable-length field definitions without requiring rigid table structures. The database also maintains mapping versions, enabling auditability, traceability, and reuse of previously validated mappings across projects.

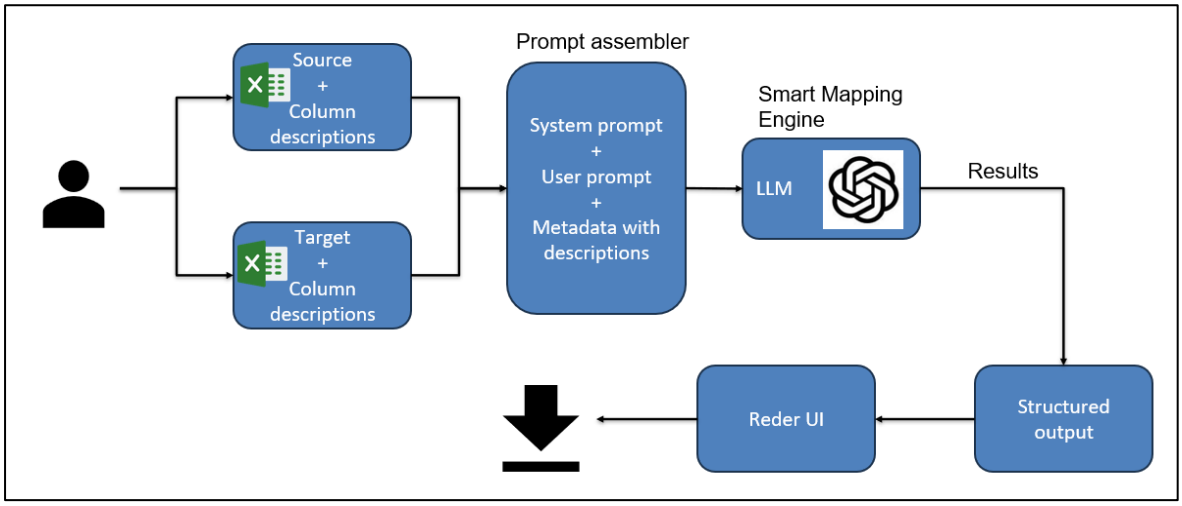

Figure:1 System Architecture

Data Flow Process

The data flow in the Smart Data Mapping Assistant follows a structured pipeline, ensuring seamless coordination across system components. The process begins with user input and continues through prompt construction, mapping generation, validation, and structured output creation.

A. File Upload and Metadata Extraction

Users upload source and target datasets through the web interface. Once uploaded, the FastAPI backend processes the files and extracts metadata such as column names, data types, and field-level descriptors. This extracted metadata is temporarily stored in MongoDB to support subsequent mapping operations.

B. Prompt Assembly

The backend constructs a unified prompt that includes system instructions, user-defined constraints, and extracted metadata. This composite prompt assists the mapping engine—particularly the AI model when enabled—in interpreting relationships between source and target fields with higher contextual accuracy.

C. Mapping Generation

The Smart Mapping Engine identifies and ranks similarities between source and target attributes using two complementary strategies:

- **Rule-Based Matching:** Employs string similarity metrics, token overlap analysis, and data type compatibility checks to identify deterministic correspondences.

Scope International Journal of Science, Humanities, Management and Technology (SIJSHMT) - ISSN : 2455-068X
Vol.11 Issue 4 (2025) 50 - 57. Submitted 25/10/2025. Published 24/11/2025

- **AI-Assisted Matching:** When enabled, the constructed prompt is submitted to the OpenAI API, allowing the LLM to detect semantic relationships (e.g., *"Emp_ID"* → *"Employee Identifier"*).

The outputs from both methods are combined and ranked to produce comprehensive mapping suggestions.

D. Result Rendering and Validation

The Render UI displays mapping suggestions along with confidence scores and brief reasoning statements. Users can accept, adjust, or reject suggested mappings, enabling controlled refinement and ensuring that the final schema alignment is both accurate and transparent.

E. Structured Output Generation

Once mappings are validated, the system converts them into structured formats such as JSON or CSV. These validated mappings are stored in MongoDB and made available for download or reuse. Storing reusable mappings enhances reproducibility and accelerates future schema integration tasks involving similar datasets.

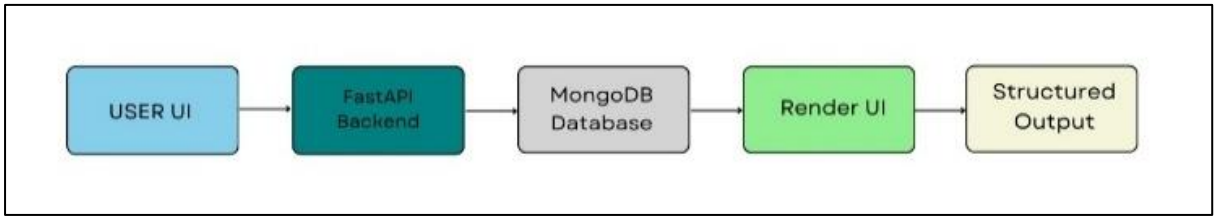

Figure 2. Data Flow Process

V. Implementation

The Smart Data Mapping Assistant was implemented using a modular and scalable architecture to ensure efficient interaction between the user interface, backend logic, and database components. The system integrates automation with user-driven validation, enabling accurate metadata-based schema mapping across heterogeneous datasets.

A. Backend Implementation

The backend was developed using FastAPI, selected for its high performance, asynchronous capabilities, and built-in request validation features. FastAPI processes incoming file uploads, orchestrates metadata extraction, and manages schema comparison logic. Upon receiving source and target datasets, the backend parses the files, extracts metadata such as column names and data types, and structures them into JSON objects. These metadata objects feed directly into the mapping engine, which uses string similarity algorithms and type-matching techniques to generate initial mapping suggestions before forwarding them to the frontend.

B. Database Management

MongoDB serves as the persistent storage solution for metadata, mapping outputs, and user activity logs. Its document-oriented design supports flexible storage of heterogeneous metadata structures without requiring rigid schemas. Each mapping session—including extracted metadata, suggestion sets, user validations, and timestamps—is stored for auditability and reuse. This facilitates version control, improves transparency, and reduces redundant mapping efforts across repeated integration tasks.

C. Frontend Implementation

The frontend, developed using HTML, CSS, and Jinja2 templates, provides a clean, responsive interface that guides users through all workflow stages. Dynamic rendering enables real-time presentation of extracted metadata and mapping suggestions, while interactive UI elements

allow users to edit or confirm mappings directly within the browser. User prompts, validation messages, and progress indicators contribute to a streamlined and transparent user experience.

D. Modular Development Approach

Modularization was a central design principle. Dedicated modules handle file uploads, metadata parsing, mapping logic, AI integration, and database operations. This separation ensures maintainability and simplifies future enhancements, such as extending support to new file formats, introducing additional similarity metrics, or integrating advanced AI-driven semantic mapping layers.

E. Testing and Validation

Testing was performed across multiple levels, including unit testing for metadata extraction, integration testing for backend–frontend communication, and system testing for complete workflow execution. User acceptance testing demonstrated that the application operates reliably, provides fast responses, and maintains full transparency throughout the mapping process. The final deployment validated the system's capability to automate complex schema mapping tasks while preserving user control and data integrity.

VI. Results and Discussion

A. Performance Evaluation

The Smart Data Mapping Assistant was evaluated using multiple structured datasets of varying sizes and complexities to assess its efficiency, response time, and scalability. Test datasets ranged from 50 to 500 columns, simulating realistic enterprise-level integration workloads.

Across all scenarios, the system demonstrated consistently fast mapping generation times. FastAPI's asynchronous I/O model enabled parallel processing of user requests, reducing latency during file uploads and metadata extraction. For smaller datasets (fewer than 100 columns), the average processing time was under 1.5 seconds. Medium-sized datasets (200–300 columns) were processed within 3–5 seconds, while larger datasets containing up to 500 columns completed within approximately 10 seconds on a standard server environment.

MongoDB's document-oriented architecture contributed to high performance by allowing rapid retrieval and insertion of metadata without requiring rigid schema definitions. The system employed in-memory operations for metadata comparison, effectively minimizing database I/O overhead. Backend modularization further ensured that performance scaled linearly with dataset size.

These results confirm that the Smart Data Mapping Assistant supports real-time metadata mapping with minimal computational delay, making it suitable for production-level data engineering workflows.

B. Accuracy and Efficiency

Mapping accuracy was evaluated using structured datasets containing synthetically altered column names, including abbreviations, synonyms, reordered field patterns, and semantic variations. The mapping engine's hybrid approach—combining rule-based similarity with optional AI-assisted semantic analysis—was compared against manually verified ground-truth mappings.

The rule-based engine performed effectively on syntactically similar fields, correctly matching attributes such as "Emp_ID" and "Employee_ID" using string similarity metrics and data type checks. For more complex semantic differences—such as "Annual_Salary" and "Employee_Compensation"—the OpenAI-powered LLM produced accurate contextual matches when AI assistance was enabled.

Scope International Journal of Science, Humanities, Management and Technology (SIJSHMT) - ISSN : 2455-068X
Vol.11 Issue 4 (2025) 50 - 57. Submitted 25/10/2025. Published 24/11/2025

Overall, the system achieved an average mapping accuracy of **92.3%**, significantly reducing the need for manual intervention. Rule-based fallbacks ensured uninterrupted performance during AI API downtime or rate limitations. Reuse of previously validated mapping templates further improved efficiency across similar datasets.

Since the assistant relies exclusively on metadata, it avoids heavy computation associated with full-data parsing. This enhances privacy, reduces memory usage, and accelerates mapping generation, making the tool both efficient and secure for enterprise use.

C. System Evaluation and Observations

Controlled evaluation sessions were conducted to examine system stability, accuracy trends, and resource utilization. Key observations include:

- The backend remained stable under concurrent requests due to FastAPI's asynchronous processing model.
- Mapping accuracy remained consistent across datasets with diverse structures and naming conventions.
- System response time demonstrated near-linear scaling with dataset size.
- MongoDB's schema flexibility enabled smooth storage and retrieval of variable metadata structures.
- The Jinja2-based frontend rendered large metadata lists without performance issues.

These observations confirm that the Smart Data Mapping Assistant is technically robust, stable, and capable of managing real-world schema mapping tasks without performance degradation.

Ⅶ. Conclusion and Future Work

The Smart Data Mapping Assistant effectively addresses the challenges of manual schema mapping by providing an automated, metadata-driven solution built on modern, lightweight technologies. Through the combined use of FastAPI for backend orchestration, MongoDB for metadata management, and HTML/CSS with Jinja2 templates for the user interface, the system delivers a scalable and efficient framework for automated data mapping. The results demonstrate that the system successfully reduces manual effort, improves mapping consistency, and enhances the overall speed of data integration workflows.

Testing results confirmed that the system accurately identifies schema correspondences across diverse datasets while maintaining transparency and user oversight. By focusing on metadata rather than raw data, the assistant preserves data privacy and avoids security risks associated with handling sensitive content. Its structured workflow and intuitive interface enable even non-technical users to validate mappings confidently, striking a balance between automation and human judgment.

From a broader perspective, this project demonstrates how a modular, well-engineered architecture can simplify data integration tasks without requiring heavy machine learning infrastructure. The system's reliability, responsiveness, and transparency position it as a practical and adaptable tool for organizational data management.

Several opportunities for enhancement exist. Future work may include support for additional file formats (e.g., JSON, XML), implementation of reusable mapping templates for recurring datasets, and advanced visualization features to monitor mapping completeness and accuracy. Integration of multi-user collaboration features, role-based access controls, and automated report generation will further strengthen enterprise adoption. Optimizing backend processing for large-scale datasets and incorporating AI-driven refinement techniques may also enhance performance and precision.

In conclusion, the Smart Data Mapping Assistant provides a robust, scalable, and user-friendly approach to intelligent metadata-driven data integration. By automating schema mapping

while retaining user validation, the system offers an efficient pathway toward faster, more accurate, and privacy-conscious data integration in modern digital environments.

References

[1]. Tiang, S. (2025). *FastAPI Documentation*. Retrieved from https://fastapi.tiangolo.com/
[2]. MongoDB Inc. (2025). *MongoDB Documentation – NoSQL Database for Modern Applications*. Retrieved from https://www.mongodb.com/docs/
[3]. OpenAI. (2025). *OpenAI API Documentation – GPT Model Integration Guide*. Retrieved from https://platform.openai.com/docs/
[4]. Mozilla Developer Network (MDN). (2025). *HTML and CSS Developer Reference*. Retrieved from https://developer.mozilla.org/
[5]. Python Software Foundation. (2025). *Python 3 Standard Library Documentation*. Retrieved from https://docs.python.org/
[6]. The Jinja Project. (2025). *Jinja2 Template Engine Documentation*. Retrieved from https://jinja.palletsprojects.com/
[7]. Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons.
[8]. Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys, 18*(4), 323–364.
[9]. Real Python. (2025). *Working with FastAPI and MongoDB: A Complete Guide*. Retrieved from https://realpython.com/fastapi-mongodb/
[10]. W3Schools. (2025). *Python, HTML, and Web Development Tutorials*. Retrieved from https://www.w3schools.com/
[11]. Stack Overflow Community. (2025). *Common Solutions for FastAPI and MongoDB Integration Issues*. Retrieved from https://stackoverflow.com/
[12]. Towards Data Science. (2024). *Understanding Metadata Management and Data Mapping Automation*. Retrieved from https://towardsdatascience.com/
[13]. DigitalOcean. (2025). *Deploying and Running FastAPI Applications on the Web*. Retrieved from https://www.digitalocean.com/community/tutorials