

Fabric: An Agentic AI Workspace for Natural-Language Data Analysis and Multi-Tool Task Execution

Sakshi Mahendra Pawar¹, Prajwal Anil Kanade², Rutika Ashok Padir³, Shivam Santosh Wajage⁴

Students, AI and Data Science¹⁻⁴

Jaihind College of Engineering, Kuran, India

Abstract: *Agentic artificial intelligence systems combine language-model interaction with tools that perform operations beyond text generation. In data analytics, an integrated workspace must support dataset handling, analytical computation, visualization, and controlled tool execution while making the limits of each capability clear. This paper presents Fabric, a software project intended to bring conversational model access and data-oriented workflows into a common workspace. The paper documents its architecture and a preliminary local evaluation on Windows 11 using Node.js v24.15.0. JavaScript syntax validation passed, and a targeted evaluation suite reported seven passing tests. A synthetic local benchmark processed 10,000 rows and verified a sum of 505,000. The broader test suite reported 415 passing tests and four failures, including a Windows filesystem-permission verification failure and spreadsheet-processing failures. These observations support only narrow claims about selected components. They do not establish end-to-end model accuracy, secure execution isolation, production readiness, or performance under real-world workloads. The paper reports the methodology, results, limitations, and further validation required for a rigorous assessment of Fabric as an AI-assisted data-analysis workspace.*

Keywords: Agentic AI, data analytics, natural-language data analysis, software evaluation, tool execution, AI workspace.

I. INTRODUCTION

An increasing proportion of language-model applications are built upon tool-use capabilities. Rather than providing purely textual output, these systems integrate functionalities such as web searching, mathematical calculation, file analysis, and the execution of application services, with the model autonomously determining when to invoke these resources. Data analysis represents a primary use case for this paradigm; a standard workflow involves importing a dataset, examining variables, sanitizing data, calculating metrics, filtering records, generating visualizations, and interpreting results. When these tasks are distributed across disparate software environments, users must manually manage data and context transitions between tools.

FABRIC is a software initiative designed to minimize this friction by consolidating these activities within a unified workspace. The core objective is to reduce the necessity for context-switching between applications; however, this serves strictly as a design rationale. As no formal user study has been conducted, this paper makes no claims regarding quantitative efficiency gains. A second premise informs the remainder of the research.

This paper makes three contributions. It describes the architectural layers of FABRIC and the path a request follows through them. It reports a preliminary local evaluation consisting of syntax validation, a targeted functional suite, a synthetic data-processing benchmark and the full automated suite. It also records the failures that remain open and specifies the evidence required before stronger claims about reliability or security can be made. The study is not a comparison of language models, a usability study or a production-readiness assessment.



II. RELATED WORK

A. Language Models and Tool Use

ReAct interleaves reasoning traces with actions so that a model can use observations from its environment while solving a task [1]. Toolformer studies how a model can learn to decide which external APIs to call, when to call them and how to use the results [2]. ToolLLM extends this line to a large collection of real-world APIs and to instructions that need several tools [3]. A survey of LLM-based agents groups the field into tool use, planning and feedback learning [4]. Taken together, these works explain why tool selection, observation handling, execution control and result presentation are central design concerns for an assistant such as FABRIC.

FABRIC is considered here as an application-level workspace, not as a new reasoning or training method. This paper does not claim that FABRIC reproduces the algorithms or training procedures of the works cited above.

B. Evaluating Data-Analysis Agents

A data-analysis workspace has several correctness boundaries. File ingestion must preserve records and column structure, transformations must carry out the requested operation, computed values must match expected values, and charts must reflect the processed data. InfiAgent-DABench evaluates language-model agents on end-to-end data-analysis tasks and illustrates why correctness criteria must be defined per task [5]. When a language model is involved, interpreting the request and writing the explanation add further sources of error. A single passing test therefore says little about overall quality, which is why this paper reports its checks in separate categories.

C. Retrieval-Augmented Generation

Retrieval-augmented generation combines a generative model with retrieved external information [6]. FABRIC's architecture includes web-research and retrieval-related components. Section V explains why their presence in the source tree is not treated as evidence of a working retrieval pipeline.

Problem Statement and Objectives

This work is motivated by two primary challenges. First, the issue of fragmentation: leveraging language models for analytical tasks typically necessitates the inefficient transfer of data and context across disjointed tools. Second, the existence of an evidence gap: existing integration frameworks, which interface models with external tools and data, are inherently prone to failure; yet, these systems are frequently characterized by their feature sets rather than by rigorous empirical validation of their capabilities.

The primary objectives of FABRIC are as follows: (i) to provide a unified workspace that integrates conversational model access, file management, data analysis, and multi-tool execution; (ii) to decouple model access from the core application architecture, thereby enabling the interoperability of local and cloud-based model providers; and (iii) to mediate tool execution through robust validation and security protocols. The secondary objectives of this paper are to provide a transparent evaluation—clearly distinguishing between demonstrated capabilities and theoretical configurations—and to establish a validation framework predicated on addressing identified system vulnerabilities.

System Architecture

A. Layered Overview

Fig. 1 shows the architecture of FABRIC as supplied by the project authors; it has been redrawn for legibility with no change to its content. The design is a set of cooperating layers rather than a single model. The user layer offers chat, voice input, file upload, data analysis and tool access. The frontend exposes a chat interface, file upload, voice input, a dashboard and settings. The backend server contains API routes, session management, security and validation, file handling and request routing. Below the interface sit the model layer and the agent and tool layer, supported by data storage, external services and system utilities.



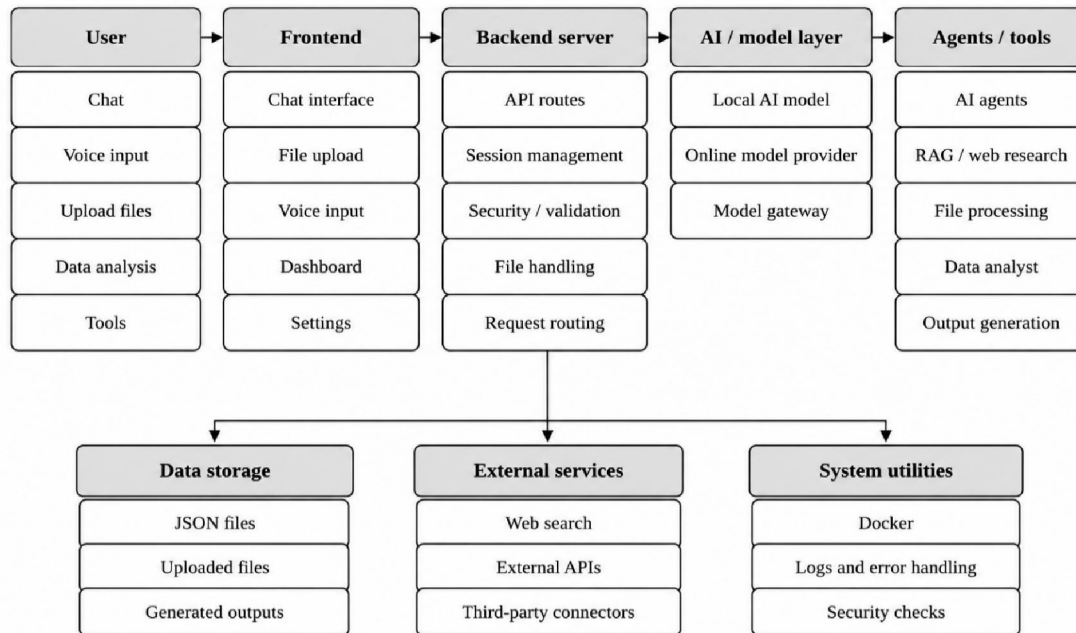


Fig. 1. Layered architecture of FABRIC (redrawn from the author-supplied diagram).

B. Model and Agent Layers

The model layer lists a local AI model, an online model provider and a model gateway. Placing a gateway between the application and the runtimes is what allows model access to be treated as a replaceable concern. The agent and tool layer lists AI agents, RAG and web research, file processing, a data analyst component and output generation. The backend routes each request to one of these components after validation.

C. Storage, External Services and Utilities

The diagram shows JSON files, uploaded files and generated outputs as the storage layer. External services comprise web search, external APIs and third-party connectors. System utilities comprise Docker, logging and error handling, and security checks. The project configuration additionally contains components for PostgreSQL, Redis with queue-related functionality, encrypted object storage and vector search. Whether these are active in the evaluated build, or are optional extensions beside the JSON-file storage in the diagram, is not established by the material available for this paper.

III. METHODOLOGY

Request Workflow

The request process comprises the six stages illustrated in Fig. 2. Initially, the user submits a request via a text-based query or data upload. The system subsequently interprets the intent and routes the request to the designated tool or model. Following execution and data processing, the system synthesizes the partial results to generate a final output, such as text, a summary, or a chart. As each stage is subject to independent failure, the evaluations presented in Sections VI and VII analyze these components separately.



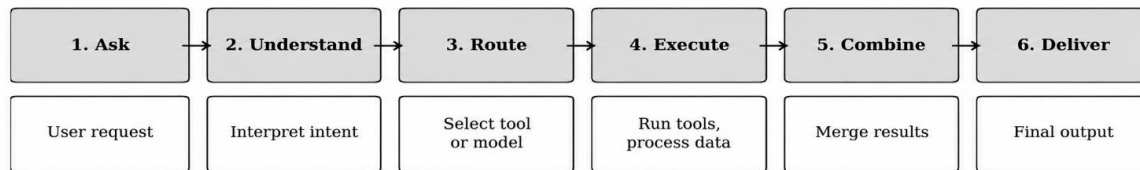


Fig. 2. Request workflow of FABRIC.

B. Model-Provider Integration

The provider layer is architected to decouple model access from broader system functionality. Within the local interface under review, Qwen2.5-Coder 3B was identified as the selected model, with Ollama serving as the local model provider. These details serve solely to document the configured environment. This study did not include an evaluation of live model outputs, a comparative analysis of providers, or an assessment of model-generated explanations.

C. Data-Analysis and Visualization Path

For a tabular task the intended path is to read the input, interpret the requested operation, compute a result and present it as a summary or chart. The synthetic benchmark isolates one narrow part of this path, deterministic summation over generated rows. It does not show that all spreadsheet formats, transformations or natural-language requests are handled correctly.

D. Tool Execution and File Handling

Tool execution is mediated by validation and security checks. The targeted evaluation includes cases for invalid-column handling, tool-abuse handling and behaviour related to server-side request forgery (SSRF). File handling is a dependency of every analytical workflow, because a malformed file that cannot be parsed prevents the analysis stage from being reached.

E. Supporting Infrastructure

The retrieval and storage components named in Section IV illustrate a distinction used throughout this paper. Configuration for a vector index does not establish a working RAG pipeline, and the presence of a database or queue setting does not prove integration. End-to-end operation depends on environment variables, service availability, permissions, schema compatibility and error handling, so each integration needs its own reproducible test.

Experimental Setup

The evaluation was limited to local software checks and one synthetic data-processing task. No user study, production load test or systematic live-model accuracy test was performed. Table I lists the environment. Four categories of check were run and are reported separately, because they exercise different code paths and should not be merged into one success rate: JavaScript syntax validation through the project's check script; the targeted suite (npm run test:evaluation), covering arithmetic, summation, filtered summation, row counting, invalid-column handling, tool-abuse handling and SSRF-related handling; the synthetic benchmark (npm run benchmark:analyst); and the full automated suite (npm test).

TABLE I: Reported Evaluation Environment

Parameter	Reported configuration
Operating system	Windows 11 Home, 64-bit
Processor	Intel Core i5-13420H
Memory	16 GB RAM



Runtime	Node.js v24.15.0
Package manager	npm 11.12.1
Model service	Ollama v0.40.2
Project revision	Fabric v1.10.4
Model selected in interface	Qwen2.5-Coder 3B

IV. RESULTS AND DISCUSSION

A. Targeted Tests and Synthetic Benchmark

The syntax check completed successfully and the targeted suite reported seven passing tests with no failures. The arithmetic and row-count cases are narrow checks of deterministic behaviour, and the invalid-column, tool-abuse and SSRF-related cases exercise selected validation paths. They do not show correctness for all inputs, and they do not show that a live model chose and ran the right tool in an end-to-end task.

The synthetic benchmark processed 10,000 rows from a 141,186-byte input and returned a sum of 505,000, equal to the expected value (Table II). The benchmark ran three trials with four visual outputs per trial. Timings varied between runs, so no latency figure is reported and none should be read as a performance guarantee. The result is a correctness check for one summation workload. It says nothing about throughput on large, irregular or real datasets, database or queue behaviour, or model quality.

TABLE II: Synthetic Benchmark Results

Metric	Reported result
Rows processed	10,000
Input size	141,186 bytes
Expected sum	505,000
Actual sum	505,000
Correctness flag	true
Reported trials	3
Visual outputs per trial	4

B. Open Failures in the Full Suite

The full suite reported 419 tests, of which 415 passed and four failed. One failure concerns the inability to verify the expected private-directory permissions on Windows, which leaves that security-relevant behaviour unverified in the tested environment. The other three arose in spreadsheet processing through the ExcelJS streaming reader, where an error occurred while reading a sheets property from an undefined value. The available evidence does not show whether the cause lies in application logic, the dependency, the test fixtures or the platform, and assigning one would need further debugging. Table III separates the successful checks from these open items.

TABLE III: Summary of Test Outcomes

Evaluation item	Reported outcome
Syntax validation	Passed
Targeted evaluation suite	7 passed, 0 failed
Synthetic benchmark	Expected sum matched actual sum
Full automated suite	415 passed, 4 failed
Spreadsheet processing	Unresolved
Windows file-permission verification	Unresolved



C. Discussion

The results support a limited engineering conclusion: selected components behaved as expected on the tested inputs. They do not support the claim that FABRIC completes arbitrary analysis tasks, because a passing unit test seldom covers the full chain from natural-language request through model interpretation, tool selection, file parsing, computation and explanation. The failures matter for the intended use. Ingestion is usually the first step of an analysis, so a parsing error can block the work even when the calculation layer is sound, and a permission check that cannot be verified should be reported as unresolved rather than counted as a pass.

The counts above belong to one source revision and one local run. For a final submission the revision, command output, environment and test artefacts should be archived, and if the tests are rerun after code changes every count and failure description must be updated together.

Applications and Advantages

The intended use is assisted analysis of tabular data by students and small teams: uploading a file, asking for counts, sums, filtered subsets or charts, and receiving the output in the same workspace. Because the model layer supports a local runtime, a deployment that uses only a local model would not need to send prompts to an external provider. This is a design possibility that has not been tested, and online providers and external services are also part of the architecture. The practical advantages that can be stated from the evidence are modest: a single interface, a replaceable model layer, and an evaluation structure that distinguishes deterministic checks from model-dependent behaviour. Benefits such as saved time or improved accuracy would need a user study.

Limitations and Future Scope

The evaluation ran on one Windows configuration. The benchmark used synthetic data and one aggregate operation, and the targeted suite had seven cases. Four failures remain open and their causes are undetermined. There is no systematic assessment of live-model accuracy, usability, multi-user load, service recovery or other operating systems, and secure isolation of arbitrary code execution has not been independently verified.

Future work begins with diagnosing the spreadsheet-reader failures and writing a reproducible Windows permission test, followed by a rerun of the full suite on the exact release candidate. The next stage is an end-to-end protocol with fixed input files, natural-language requests and independently computed answers, covering counting, sums and averages, filtering, grouping and missing-value detection. Results should be reported per stage: operation-selection accuracy, calculation correctness, visualization validity, explanation faithfulness and overall task completion. Failure-condition tests should cover a missing provider, an unavailable database, invalid credentials, a malformed upload and an interrupted analysis.

V. CONCLUSION

This paper introduces FABRIC, a layered AI workspace that integrates conversational model access, data analysis, file management, and multi-tool execution. A preliminary local evaluation confirmed successful syntax validation, the completion of seven targeted tests, and the accurate summation of a 10,000-row synthetic dataset. Furthermore, the system successfully passed 415 out of 419 tests in the full suite. Four unresolved failures related to spreadsheet processing and Windows permission verification persist, limiting the current evidence to the tested components. Subsequent debugging, comprehensive integration and end-to-end model evaluations, and rigorous security validation are required to establish FABRIC as a reliable solution for production-grade analytical workloads.

ACKNOWLEDGMENT

The authors wish to express their sincere gratitude to their project supervisor for their invaluable guidance, constructive feedback, and consistent encouragement throughout the duration of this research. The authors extend their appreciation to Prof.S.K.Said, Head of the Department of Artificial Intelligence and Data Science at Jaihind College of Engineering,



Kuran, for facilitating the academic environment and resources necessary to conduct this study. Furthermore, the authors thank the teaching and administrative staff of the department for their support and insightful contributions during the project's development.

The authors are grateful to the Principal, Dr.A.B.Ubale, and the management of Jaihind College of Engineering for providing the institutional facilities and for fostering a culture that encourages student research and publication. Finally, the authors express their heartfelt gratitude to their families and friends for their ongoing motivation and support.

REFERENCES

- [1] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023. [Online]. Available: <https://openreview.net/pdf?id=WwGWMAktQY>
- [2] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. [Online]. Available: <https://proceedings.neurips.cc/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html>
- [3] Y. Qin et al., "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html
- [4] X. Li, "A review of prominent paradigms for LLM-based agents: Tool use (including RAG), planning, and feedback learning," *arXiv preprint arXiv:2406.05804*, 2024. [Online]. Available: <https://arxiv.org/abs/2406.05804>
- [5] X. Hu et al., "InfiAgent-DABench: Evaluating agents on data analysis tasks," in *Proc. 41st Int. Conf. Machine Learning, Proc. Machine Learning Research*, vol. 235, pp. 19544–19572, 2024. [Online]. Available: <https://proceedings.mlr.press/v235/hu24s.html>
- [6] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- [7] Fabric Development Project, "Fabric v1.10.4: Source code, automated tests, and synthetic benchmark," software archive, 2026. [Online]. Available: <https://drive.google.com/file/d/1RbeKs0LQtXhd2KGdbsQGpLU2vdNO3iQd/view?usp=drivesdk>

