NOUN Interdisciplinary Journal of Computing, E-learning & Application (NOUN-IJCEA)
https://ijcea.nou.edu.ng
Volume 1 Number 1, April 2026 pp. 121-132

A Unified Feedback-Driven Data Preparation Framework Using Persistent Feedback Artefacts

*[1,2] Sa'adatu Abdulkadir, [1] Philip O. Odion, [1] Darius T. Chinyio, [1] Isah R. Saidu, and [3] Muhammad A. Ahmad

[1]Department of Computer Science, Nigerian Defence Academy, Kaduna – Nigeria
[2]Department of Informatics, Kaduna State University, Kaduna – Nigeria
[3]Department of Secure Computing, Kaduna State University, Kaduna – Nigeria
*Corresponding Author: saa.abdul@kasu.edu.ng

ABSTRACT

Data preparation often requires analysts to make corrections that depend on the meaning and intended use of the data, not only its format. In many preparation tools, those corrections remain in the current file, script, or processing session and are unavailable when a similar problem appears in a later dataset. This study develops a Unified Feedback-Driven Data Preparation framework that incorporates the Persistent Feedback Artefact Model (PeFAM) into an end-to-end preparation workflow. The framework links dataset ingestion, staging, validation, transformation, feedback capture, persistent storage, analyst review, selective reuse, and export. A web prototype was implemented in PHP within a local XAMPP environment. CSV files were used for input and output, while MySQL records stored dataset settings, current-session feedback, and persistent feedback artefacts. The prototype retained validated corrections together with their original values and available contextual information. During a later task, it retrieved an artefact when the stored original value appeared in the current dataset and the affected attribute matched by exact column name or configured semantic role. The analyst then decided whether to select the suggestion. Selected items were copied to the active feedback store for use during transformation, while unselected items were not applied. The demonstration shows that these functions can operate within one preparation environment. The study does not evaluate efficiency, accuracy, scalability, or usability; those outcomes require a separate controlled evaluation. Its contribution is the operational connection between persistent artefact management and the core stages of data preparation.

Keywords: Adaptive Data Preparation, Context-Aware Retrieval, Feedback Reuse, Human-in-the-Loop, Persistent Feedback Artefacts

1. INTRODUCTION

Data preparation involves more than correcting obvious formatting errors. Analysts must often decide whether a value is genuinely incorrect, which transformation is suitable, and whether an apparent inconsistency reflects a valid domain convention. Rule-based preprocessing and Extract–Transform–Load workflows can automate routine operations, but they are less effective when schemas change or a decision depends on knowledge that is not available from the data structure alone (Azeroual, 2020; Fernandes et al., 2023; Konstantinou & Paton, 2020; Paton, 2019). Human-in-the-Loop methods address this limitation by allowing users to inspect results, correct values, reject unsuitable transformations, and contribute contextual knowledge during preparation (Mosqueira-Rey et al., 2022; Rezig et al., 2019).

A correction made during one preparation task is not always available when the same problem appears again. In many systems, feedback remains in the current spreadsheet, script, interface, model, or execution history. The corrected output may be retained, but the reason for the correction, the affected attribute, the source dataset, and the validation decision may not be preserved in a reusable form. The analyst may therefore need to interpret and correct the same issue again in another dataset. A systematic review of 29 feedback-driven data-preparation studies found that feedback was used during individual cleaning, schema matching, transformation, and validation activities. Persistent storage and reviewed reuse across tasks or datasets received much less attention (Abdulkadir et al., 2026a).

The Persistent Feedback Artefact Model (PeFAM) addressed the representation of such corrections (Abdulkadir et al., 2026b). It defines a feedback artefact containing the original and corrected values, context, metadata, temporal information, and provenance. PeFAM also provides similarity and applicability logic for identifying an artefact that may be relevant to a later issue. The model does not, however, define the complete system needed to create the artefact during preparation, retain it in a repository, retrieve it during a later task, present it for review, and record what happens after the analyst makes a reuse decision.

This study develops a Unified Feedback-Driven Data Preparation (UFDP) framework to provide that operational connection. The framework places feedback management within the same workflow used for ingestion, staging, validation, transformation, and export. It separates corrections associated with the current task from validated artefacts retained for later use. A PHP/MySQL prototype was developed in a local XAMPP environment using CSV files as the main input and output format. In the prototype, a stored artefact is retrieved when its previously corrected original value appears in the current dataset and the affected attribute matches either by exact column name or by a configured semantic role. The retrieved item is displayed as a suggestion, and the analyst decides whether to copy it into the active feedback store for transformation.

The objective of this study is to design and demonstrate an end-to-end data-preparation framework that operationalises PeFAM artefact capture, persistent storage, contextual retrieval, analyst review, and selective reuse within a working prototype.

The contribution of the study is the framework and prototype that connect PeFAM artefacts to the operational stages of data preparation. The study does not alter the PeFAM artefact structure or its general similarity logic. It shows how the existing model can be used within a working preparation environment while keeping the final reuse decision with the analyst. The present demonstration is limited to functional and architectural feasibility. It does not measure efficiency, accuracy, retrieval effectiveness, scalability, or usability; those outcomes are addressed through a separate controlled evaluation.

2. LITERATURE REVIEW

Data preparation covers the activities required to make raw data suitable for analysis, including profiling, cleaning, schema alignment, transformation, validation, integration, and duplicate resolution (Fernandes et al., 2023; Njeri, 2022). Many of these activities can be implemented through predefined rules, scripts, or Extract–Transform–Load workflows. Such methods work well when the data structure and the required transformations are already known. They become less reliable when schemas change, unfamiliar values appear, or the correct treatment depends on the meaning of the data rather than its format alone (Azeroual, 2020; Cormier *et al.*, 2025; Konstantinou & Paton, 2020; Paton, 2019; Wojciechowski, 2018). Interactive wrangling tools offer greater flexibility, but users may still switch between separate interfaces or tools for cleaning, transformation, validation, and export (Kasica et al., 2020).

Human-in-the-Loop approaches allow users to contribute information that cannot be inferred reliably from data patterns alone. An analyst may correct a value, confirm that an apparent anomaly is valid, reject a proposed transformation, or explain how an attribute should be interpreted (Mosqueira-Rey et al., 2022; Rezig et al., 2019; Wu et al., 2023). Feedback may be collected explicitly through corrections, approvals, rejections, and annotations, or inferred from interaction behaviour (Narayan et al., 2022). The present study is concerned with explicit corrections that have been reviewed and accepted by a user.

Several systems use such feedback to improve the task currently being performed. The VADA architecture links end-user feedback with data-preparation components and workflow orchestration, while human-centred cleaning approaches allow users to guide repair and transformation decisions (Konstantinou et al., 2019; Rezig et al., 2019). Recent systems also use foundation models or large language models to generate transformations from examples, instructions, or user interaction (Liu et al., 2024; Narayan et al., 2022). These approaches broaden the range of operations that can be suggested during a task. However, the correction or generated transformation often remains within the current interaction, script, model context, or workflow history.

Retaining a transformation history is not the same as retaining a reusable corrective decision. A history may show that a value was changed, but it may not record why the change was accepted, which attribute was affected, who validated it, or whether the same decision is suitable for another dataset. Similarly, a reusable script can repeat an operation without establishing that the operation remains appropriate under different source conventions or domain requirements. Later reuse therefore requires more than persistence alone; the stored correction must retain enough information for its relevance and applicability to be assessed.

The systematic review that preceded this study examined 29 publications on feedback-driven data preparation (Abdulkadir et al., 2026a). Feedback was used mainly in cleaning, schema matching, transformation, and validation. Much less attention was given to retaining validated corrections as independent artefacts and making them available for review in later tasks. Existing research therefore provides useful support for workflow coordination, user judgement, transformation generation, and traceability, but these functions are not commonly combined around the cross-task reuse of a validated correction.

PeFAM addresses the representation required for such reuse (Abdulkadir et al., 2026b). It stores the original and corrected values together with context, metadata, temporal information, and provenance. It also provides contextual-similarity and applicability logic for identifying artefacts that may be relevant to a later issue. A similarity result only makes an artefact eligible for review. It does not authorise automatic application.

PeFAM does not define the complete preparation environment in which the artefact is created, stored, retrieved, reviewed, and applied. It also does not coordinate those activities with ingestion, staging, validation, transformation, outcome recording, and export. The present study addresses this operational gap by incorporating PeFAM into a working data-preparation framework and prototype. Table 1 summarises the difference between the PeFAM study and the framework presented here.

Table 1: Distinction between the PeFAM study in Abdulkadir *et al.* (2026b) and the present framework

Aspect	PeFAM study	Present framework
Main focus	Formal representation of reusable feedback	Operational integration of PeFAM
Principal contribution	Defines artefact structure and applicability logic	Coordinates artefacts within an end-to-end workflow
Process	Representation, persistence requirements, and	Capture, storage, retrieval, review,

Aspect	PeFAM study	Present framework
coverage	contextual assessment	reuse, outcome recording, and export
Evidence	Analytical demonstration of representational adequacy	Prototype-based feasibility demonstration

3. METHODOLOGY

The framework and prototype were developed through five linked phases. These comprised requirements elicitation, conceptual and architectural modelling, incremental prototype development, preliminary user-based testing, and system refinement. Functional and non-functional requirements were identified through the preceding literature review, analysis of existing data-preparation tools and workflows, and interviews with intended users. The requirements guided the conceptual workflow, feedback-reuse architecture, database structures, module interactions, and implementation of the upload, staging, validation, transformation, feedback-management, export, and audit functions. Development proceeded iteratively, with each cycle producing a functional implementation of one or more workflow capabilities that was exercised through representative data-preparation tasks. User comments, observed interaction difficulties, and workflow breakdowns informed revisions to the interface structure, validation prompts, transformation logic, error handling, and presentation of feedback suggestions. This section reports the resulting framework and prototype as evidence of functional integration. Comparative performance and user outcomes are outside the scope of the present study.

3.1 Conceptual Framework

The framework was designed to address two practical problems. The first is the separation of preparation activities across different tools or workflow stages. The second is the loss of user corrections after the task in which they were made. To address both problems, the framework integrates dataset preparation and feedback management into a single operational environment.

Figure 1 organises the framework into four layers. The presentation layer contains the interfaces through which users upload datasets, inspect staged data, respond to validation prompts, review transformations, submit feedback, and export prepared data. The application and workflow layer controls the sequence of preparation activities. It handles ingestion, staging, validation, schema-related operations, transformation, feedback capture, reusable-feedback matching, analyst review, and audit recording.

The data and persistence layer stores the information used by these operations. It includes the operational data store, uploaded datasets, generated outputs, and the persistent feedback artefact repository. The repository retains validated corrections together with the information required to understand their origin and later use. Authentication, access control, configuration, logging, error handling, and session security are treated as cross-cutting services because they support multiple preparation modules.

Feedback enters the framework when an analyst corrects or confirms an issue during preparation. A correction approved for later use is represented as a PeFAM artefact and stored in the persistent repository. It is therefore not lost when the current preparation session ends. When the same original value occurs in a later dataset and the attribute context satisfies the implemented matching conditions, the artefact is presented to the analyst as a possible reusable correction. The analyst reviews the suggestion before deciding whether to use it in the current task.

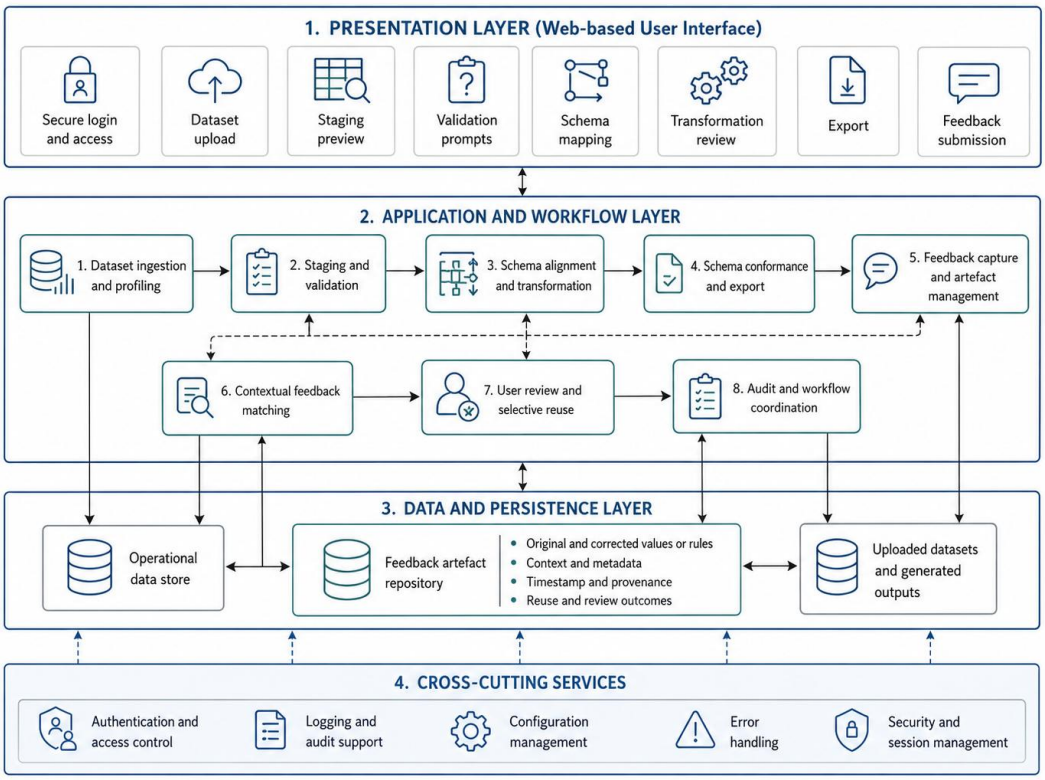


Figure 1: Conceptual architecture of the Unified Feedback-Driven Data Preparation (UFDP) framework

3.2 Design Principles and PeFAM Integration

The framework design was guided by three requirements observed during development. Preparation activities needed to remain connected so that a correction could be linked to the dataset, attribute, and transformation involved. Validated corrections also needed to remain available after the current session ended. Finally, retrieving an earlier correction could not be treated as permission to apply it automatically; the analyst had to review it in relation to the current dataset.

PeFAM provides the representation used to meet these requirements (Abdulkadir et al., 2026b). A feedback artefact is defined as:

$$FA = (v_o, v_c, C, M, t, p) \quad (1)$$

where v_o represents the original value, state, or rule; v_c represents the validated corrected value or modified rule; C contains contextual information; M contains metadata; t records temporal information; and p records provenance. The PeFAM study provides the formal definition and justification of these elements. In the present framework, they provide the information needed to identify what was corrected, where the correction originated, and whether it may be relevant to a later preparation task.

The implementation keeps current-session feedback separate from artefacts retained for future use. The active feedback store contains corrections associated with the preparation run currently in progress. Some of these corrections may still be tentative or may apply only to the current dataset. The persistent feedback artefact repository contains corrections that have been validated and retained for possible reuse.

This distinction affects how the prototype operates. Resetting or completing the current preparation run clears the active feedback associated with that run but does not remove validated artefacts stored earlier. A current correction also does not enter the reusable repository merely because it has been entered in the staging interface. It must first be confirmed and saved as a validated correction. The

separation therefore protects previously retained knowledge while preventing tentative decisions from being suggested in later tasks.

3.3 Contextual Retrieval, Human Review and Selective Reuse

PeFAM defines the broader logic used to determine whether a stored artefact may be relevant to a current preparation issue (Abdulkadir et al., 2026b). The comparison may consider properties such as the affected attribute, semantic role, data type, value pattern, transformation category, source dataset, dataset characteristics, domain convention, and workflow stage. These properties describe the circumstances in which the earlier correction was made and provide a basis for assessing whether it should be considered again.

A contextual match does not mean that the stored correction is automatically suitable for the current dataset. It only identifies the artefact as a candidate for review. Differences in source conventions, attribute meaning, domain requirements, or intended use may make an earlier correction inappropriate even where some contextual properties are similar. The framework therefore separates retrieval from application. The system identifies possible matches, while the analyst decides whether a retrieved artefact should be selected for the current task.

The prototype implemented a narrower matching procedure than the complete PeFAM mechanism. It first checked whether the original value stored in an artefact occurred in the current dataset. It then checked whether the affected attribute matched either by exact column name or by a semantic role configured for both attributes. For example, an artefact created for a column named *Gender* could be retrieved for a column named *Sex* where both columns had been assigned the same semantic role.

Artefacts satisfying these conditions were displayed as reusable-feedback suggestions. The display allowed the analyst to inspect the stored original value, corrected value, and available contextual information before making a decision. A selected suggestion was copied into the active feedback store and became available during transformation. An unselected suggestion was not applied to the current dataset. Where none of the retrieved artefacts was suitable, the analyst could enter a new correction through the staging interface.

3.4 Prototype Implementation

The prototype was developed as a PHP web application in a local XAMPP environment. CSV files were used as the main input and output format, while MySQL stored dataset configurations, active feedback records, and persistent feedback artefacts. Users could either upload a CSV file directly or import Google Form responses through a Google Sheets CSV link.

Before processing a dataset, the user specified the settings required for that task. These included column roles, formatting rules, missing-value treatment, the attribute used for duplicate detection, and optional cross-column consistency checks. The settings were stored with the dataset configuration so that the same application could process files with different structures rather than relying on a single fixed schema.

The preparation sequence included dataset upload, staging, validation, correction, transformation, duplicate review, and export. During staging, the user could inspect values identified by the configured checks and enter a correction directly in the interface. The prototype also allowed the user to review reusable-feedback suggestions before transformation.

Feedback created during the current run was stored separately from artefacts retained from earlier tasks. When a correction was confirmed and saved for possible later reuse, its original value, corrected value, affected attribute, semantic role, dataset information, issue type, timestamp, and provenance were recorded in the persistent repository. In a later task, the prototype checked whether the stored original value occurred in the new dataset and whether the affected attribute matched by

exact column name or configured semantic role. Artefacts meeting these conditions were displayed to the analyst as suggestions.

The analyst could select a suggestion for the current task or leave it unselected. Selected artefacts were copied into the active feedback store and applied during transformation. Unselected suggestions were not used. Where no stored artefact was suitable, the analyst could enter a new correction during staging.

For this implementation, a correction was treated as validated when the analyst entered or confirmed the corrected value and saved it for transformation and possible later reuse. This operational definition reflects the prototype workflow and does not represent independent expert validation.

The prototype was developed to demonstrate that the preparation and feedback-management functions could operate together. This study did not measure preparation time, residual errors, retrieval precision, scalability, concurrent use, or usability. Those outcomes require a separate controlled evaluation.

4. RESULTS AND DISCUSSION

4.1 Prototype Demonstration

The prototype demonstration followed a correction from its creation during one preparation task to its retrieval during a later task. This sequence was used to show how the preparation modules, active feedback store, and persistent artefact repository worked together.

The user began by uploading a CSV file or importing Google Form responses through a Google Sheets CSV link. Before processing started, the user configured the dataset by assigning column roles and specifying relevant preparation rules. These settings could include formatting requirements, missing-value treatment, the field used for duplicate detection, and optional checks involving more than one column. This allowed the same prototype to be used with datasets that did not share an identical schema.

After configuration, the dataset was displayed in the staging interface. The prototype identified issues according to the selected rules, including missing values, invalid formats, and configured cross-column inconsistencies. The analyst could inspect the affected records and enter or confirm corrections directly within the interface.

When a correction was saved for later use, the prototype recorded the original value, the corrected value, the affected attribute, the configured semantic role, the source dataset, the issue type, the timestamp, and any available provenance information. Feedback associated only with the current run remained in the active feedback store. Corrections retained for possible use in later tasks were stored in the persistent feedback artefact repository.

During a later preparation task, the prototype searched the repository when a stored original value appeared in the new dataset. It also checked whether the affected column matched by exact name or by a configured semantic role. Artefacts satisfying these conditions were shown in the reusable-feedback review interface. The analyst could select a suggestion, leave it unselected, or enter a different correction during staging.

A selected artefact was copied into the active feedback store and used during transformation. An unselected suggestion was excluded from the current run. The transformation module then applied the approved feedback together with the configured preparation rules and generated a cleaned CSV file.

The prototype also supported optional comparison with a reference standard dataset and the export of an evaluation report. These functions were included to support the separate empirical evaluation and are not analysed in the present study. Table 2 summarises the relationship between the principal framework requirements, their implementation in the prototype, and the evidence provided by the demonstration.

Table 2: Framework requirements and prototype evidence

Framework requirement	Prototype implementation	Demonstration evidence
Current-session feedback capture	Active feedback records created during staging	Staging interface and saved corrections
Persistent artefact storage	Validated corrections retained in MySQL repository	Repository record available after session reset
Contextual retrieval	Original-value and exact attribute/semantic-role matching	Reusable-feedback suggestion interface
Human applicability review	Analyst selects or leaves suggestions unselected	Review interface shown in Figure 2(c)
Selective reuse	Selected items copied to active feedback store	Feedback rules used in transformation
Outcome traceability	Source and transformation use recorded	Transformation summary and audit records

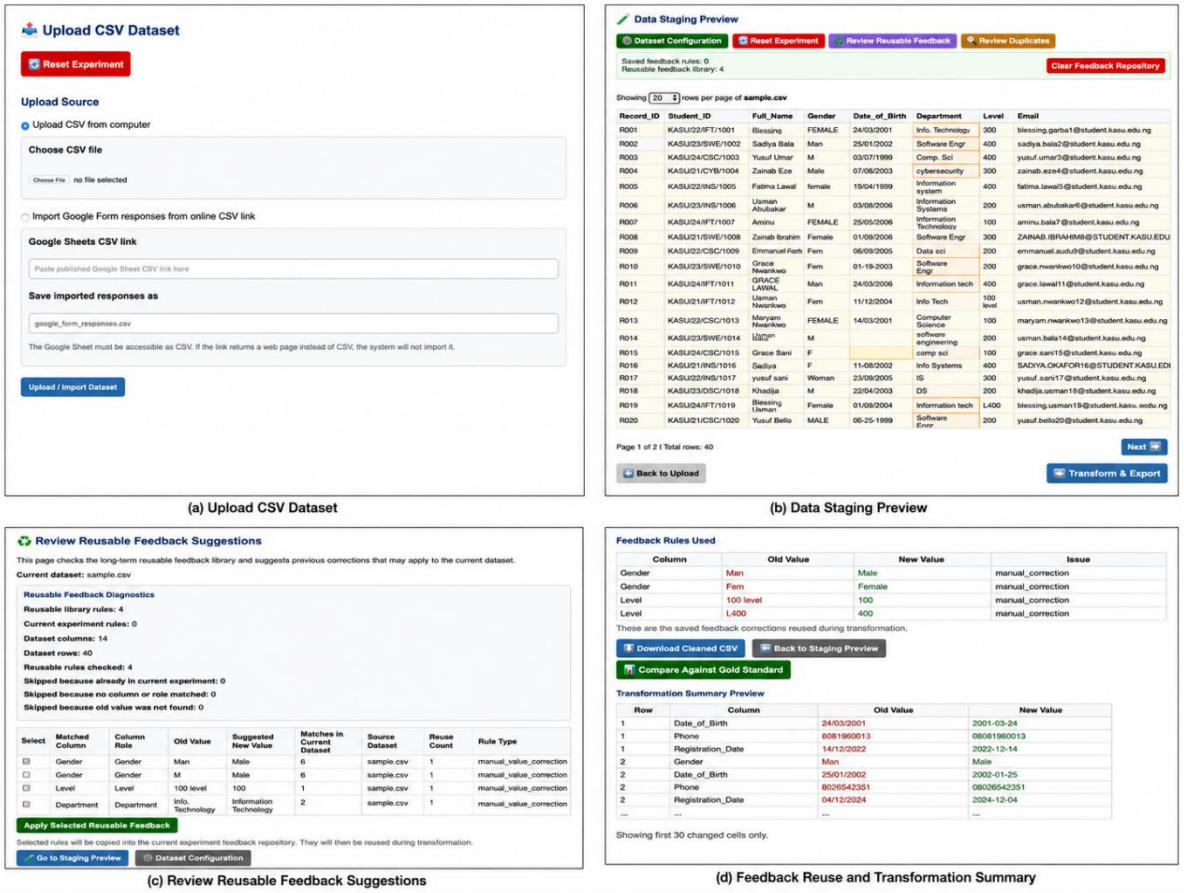
4.2 Feedback Reuse Scenario

The reuse process can be illustrated with a categorical standardisation example. During an initial preparation task, the analyst encounters the value *M* in an attribute configured as representing gender. After confirming that the intended standard form is *Male*, the analyst saves the correction.

The prototype records the original value *M*, the corrected value *Male*, the affected attribute, its configured semantic role, the source dataset, the issue type, the time of the correction, and available provenance information. Because the correction is retained for possible later use, it is stored in the persistent feedback artefact repository rather than remaining only in the active feedback store.

In a later preparation task, the prototype checks whether the stored original value *M* occurs in the current dataset. It then checks the attribute context. A candidate may be retrieved where the current column has the same name as the earlier column or where both columns have been assigned the same semantic role. For example, an artefact created for a column named *Gender* may be presented for a column named *Sex* where both are configured with the same semantic role.

The retrieved artefact is displayed as a suggestion rather than applied automatically. The analyst reviews the stored correction and its available contextual information before deciding whether it is suitable for the current dataset. Where the suggestion is selected, it is copied into the active feedback store and used during transformation. Where it is not selected, it has no effect on the current task. The analyst may instead enter a different correction through the staging interface. Figure 2 shows the prototype interfaces used in this sequence, including dataset upload, data staging, reusable-feedback review, and the feedback-reuse and transformation summary.



(a) Upload CSV Dataset

(b) Data Staging Preview

(c) Review Reusable Feedback Suggestions

(d) Feedback Reuse and Transformation Summary

Figure 2: Prototype demonstration interfaces

4.3 Architectural Discussion and Limitations

The framework adds persistent feedback management to the normal sequence of data preparation activities. In a conventional task, a correction may remain in a spreadsheet, script, transformation history, or individual tool. In the proposed framework, a correction that has been confirmed for possible later use is stored separately as a feedback artefact. This allows it to remain available after the original preparation session has ended.

The separation between the active feedback store and the persistent repository is important to the operation of the framework. Feedback created during the current run may still be incomplete, temporary, or specific to the active dataset. It therefore remains in the active store until the analyst confirms it for possible later use. Previously validated artefacts remain in the persistent repository and are not removed when the current session is reset.

Retrieval is also kept separate from application. The prototype may identify an artefact because the stored original value occurs again and the affected attribute matches by exact name or configured semantic role. This match only places the artefact before the analyst for review. The analyst may select it, leave it unselected, or enter another correction. This arrangement prevents an earlier correction from being treated as universally valid.

The prototype records the reusable artefacts selected for the current task, their source, and their use during transformation. These records allow the analyst to trace which earlier correction contributed to the prepared output. The implementation did not comprehensively record every possible reuse outcome, such as adaptation, later revalidation, conflict between artefacts, or eventual retirement of outdated corrections.

The preparation modules and feedback-management functions were implemented as separate parts of the workflow. In a larger system, ingestion, transformation, validation, repository access, or retrieval

services could therefore be replaced or extended without changing the central sequence of artefact capture, storage, review, and selective reuse. The use of PHP, XAMPP, CSV, and MySQL reflects the prototype environment rather than a requirement of the framework.

Figure 3 summarises the operational sequence demonstrated in the prototype. A dataset is prepared, a user correction is captured, and a validated artefact is stored in the persistent repository. When a related issue occurs in a later task, the prototype retrieves a possible match, presents it for analyst review, and copies a selected suggestion to the active feedback store before transformation and export.

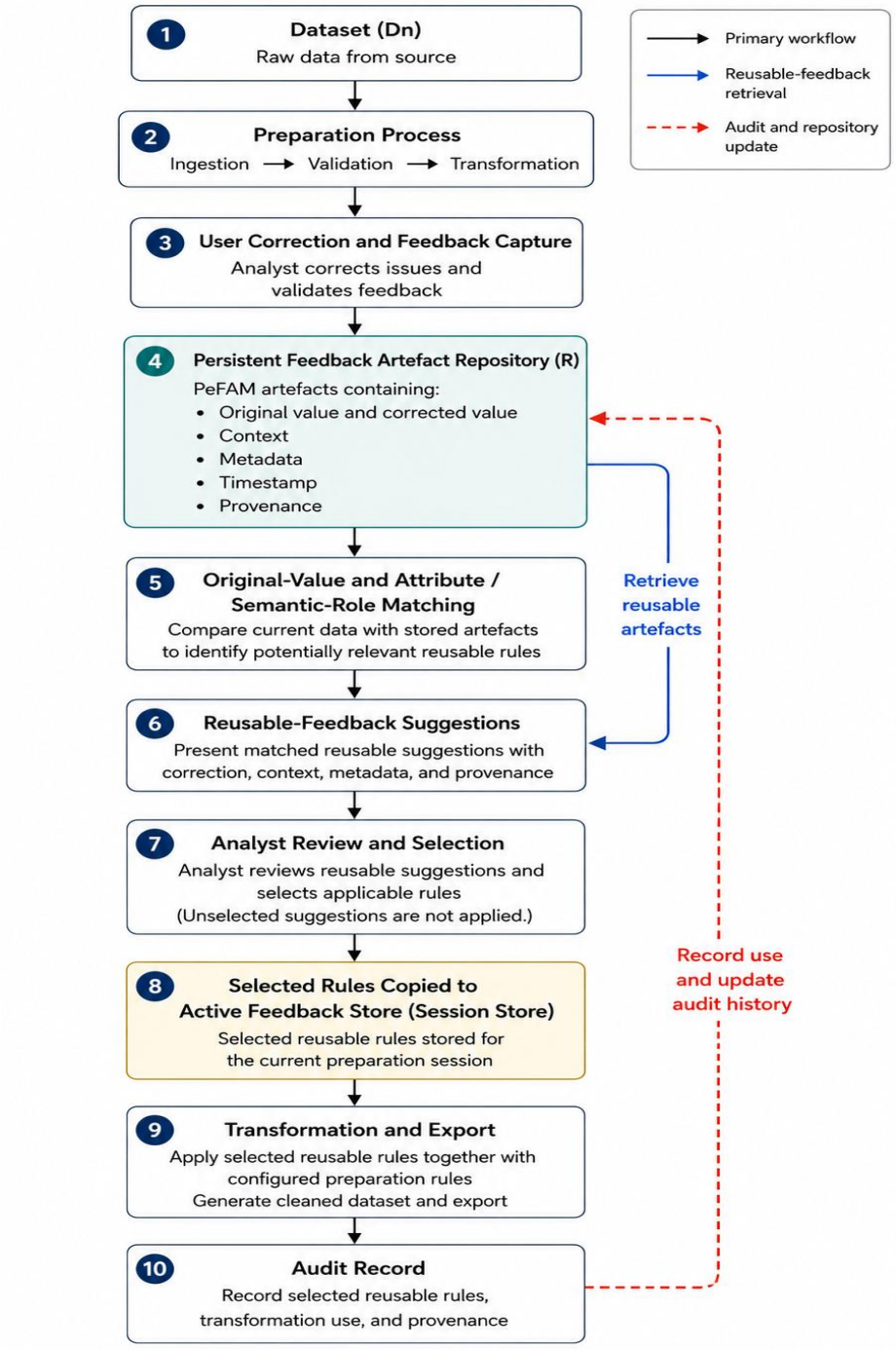


Figure 3: Operational lifecycle of feedback capture, lightweight contextual matching, analyst review, selective reuse, and transformation

The prototype demonstrates that these operations can be integrated into a single working environment, but it does not demonstrate how well they perform under larger or more demanding conditions. It was implemented locally and was not tested with very large datasets, multiple simultaneous users, distributed processing, or long-term repository growth.

Retrieval also depends on the quality of the stored artefact and the configuration of the current dataset. A relevant correction may not be retrieved where its original value does not recur, where the column name differs and no semantic role has been assigned, or where contextual information is incomplete. Conversely, an artefact may be retrieved even though differences in domain meaning or intended use make it unsuitable. Analyst review remains necessary for this reason.

A larger repository would require indexing and filtering rather than examining every stored artefact during each task. Multi-user use would also require access control, conflict management, versioning, revalidation, concurrency handling, and procedures for retiring outdated artefacts. These functions were not evaluated in the present prototype.

The current study therefore establishes functional feasibility rather than performance. Preparation time, residual errors, retrieval quality, reuse effectiveness, scalability, and usability require controlled empirical evaluation using defined datasets, comparison conditions, and participant groups.

5. CONCLUSION

This study implemented PeFAM within a unified data-preparation workflow that connects dataset ingestion, staging, validation, transformation, feedback capture, persistent storage, analyst review, selective reuse, and export. A PHP/MySQL prototype operating in a local XAMPP environment with CSV input and output demonstrated how a validated correction could be retained with its original value and available contextual information, retrieved during a later task when the stored original value recurred and the affected attribute matched by exact column name or configured semantic role, and presented to the analyst for review before reuse. Selected suggestions were copied into the active feedback store and used during transformation, while unselected suggestions were excluded from the current run. The study therefore shows that PeFAM artefacts can be incorporated into a working preparation environment without altering the model's artefact structure or general applicability logic. Its contribution is the operational connection between persistent feedback management and the normal stages of data preparation. The prototype establishes functional feasibility only and does not provide evidence of improved efficiency, accuracy, retrieval quality, scalability, or usability; these outcomes require controlled evaluation with additional datasets, analysts, task conditions, and deployment environments.

ACKNOWLEDGEMENT

This research is supported by the Tertiary Education Trust Fund (TETFund), Nigeria, under the Academic Staff Training and Development (AST&D) intervention.

REFERENCES

Abdulkadir, S., Odion, P. O., Chinyio, D. T., Saidu, I. R., & Ahmad, M. A. (2026a). Feedback-driven automation in data preparation: A systematic literature review. *Science World Journal (SWJ), 21*(1), 290-303. https://doi.org/10.4314/swj.v21i1.39

Abdulkadir, S., Odion, P. O., Chinyio, D. T., Saidu, I. R., & Ahmad, M. A. (2026b). Formalising feedback as reusable knowledge in data preparation systems: A persistent feedback artefact model. *FUDMA Journal of Sciences (FJS)*. Accepted for publication.

Azeroual, O. (2020). Data wrangling in database systems: Purging of dirty data. *Data, 5*(2), Article 50. https://doi.org/10.3390/data5020050

Cormier, K., Gagnier, K., Padron-Uy, J., Sareen, D., Parihar, A., Khmelevsky, Y., Hains, G., & Wong, A. (2025). Data extraction, transformation, and loading (ETL) process automation and data

warehouse implementation for algorithmic trading machine learning modelling. In *2025 IEEE International Systems Conference (SysCon)* (pp. 1–8). IEEE. https://doi.org/10.1109/syscon64521.2025.11014812

Fernandes, A. A. A., Koehler, M., Konstantinou, N., Pankin, P., Paton, N. W., & Sakellariou, R. (2023). Data preparation: A technological perspective and review. *SN Computer Science, 4*(4), Article 425. https://doi.org/10.1007/s42979-023-01828-8

Kasica, S., Berret, C., & Munzner, T. (2020). Table Scraps: An actionable framework for multi-table data wrangling from an artifact study of computational journalism. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 957–966. https://doi.org/10.1109/tvcg.2020.3030462

Konstantinou, N., Abel, E., Bellomarini, L., Bogatu, A., Civili, C., Irfanie, E., Koehler, M., Mazilu, L., Sallinger, E., Fernandes, A. A. A., Gottlob, G., Keane, J. A., & Paton, N. W. (2019). VADA: An architecture for end user informed data preparation. *Journal of Big Data, 6*(74), 1-32. https://doi.org/10.1186/s40537-019-0237-9

Konstantinou, N., & Paton, N. W. (2020). Feedback driven improvement of data preparation pipelines. *Information Systems, 92*, Article 101480. https://doi.org/10.1016/j.is.2019.101480

Liu, L., Hasegawa, S., Sampat, S. K., Xenochristou, M., Chen, W., Kato, T., Kakibuchi, T., & Asai, T. (2024). AutoDW: Automatic data wrangling leveraging large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (pp. 2041–2052). ACM. https://doi.org/10.1145/3691620.3695267

Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., & Fernández-Leal, Á. (2022). Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, *56*(4), 3005–3054. https://doi.org/10.1007/s10462-022-10246-w

Narayan, A., Chami, I., Orr, L., & Ré, C. (2022). Can foundation models wrangle your data? *Proceedings of the VLDB Endowment, 16*(4), 738-746. https://doi.org/10.14778/3574245.357425

Njeri, N. R. (2022). Data preparation for machine learning modelling. *International Journal of Computer Applications Technology and Research*, *11*(06), 231–235. https://doi.org/10.7753/ijcatr1106.1008

Paton, N. (2019). Automating data preparation: Can we? Should we? Must we? *Research Explorer, the University of Manchester*. https://research.manchester.ac.uk/en/publications/automating-datapreparation-can-we-should-we-must-we/

Rezig, E. K., Ouzzani, M., Elmagarmid, A. K., Aref, W. G., & Stonebraker, M. (2019). Towards an end-to-end human-centric data cleaning framework. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics* (pp. 1–7). ACM. https://doi.org/10.1145/3328519.3329133

Wojciechowski, A. (2018). ETL workflow reparation by means of case-based reasoning. *Information Systems Frontiers, 20,* 21-43. https://doi.org/10.1007/s10796-016-9732-0

Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., & He, L. (2023). A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems, 135*, 364–381. https://doi.org/10.1016/j.future.2022.05.014