Archives available at journals.mriindia.com

Multidisciplinary Journal of Research in Engineering and Technology

ISSN: 2348-6953

Volume 13 Issue 01, 2026

An Intelligent Data Anonymization Framework for Privacy-Preserving Sensitive Information Protection

[1]Awais Ali Ahmed Ali Khan, [2]Syed Abrar Azhar, [3]V. S. Karwande

[1]Computer Science and Engineering Department, Everest College of engineering and technology, chhatrapati Sambhajinagar

[2]Computer Science and Engineering Department, Everest College of engineering and technology, chhatrapati Sambhajinagar

[1]awais.k0394@gmail.com, [2]hodcse@eescoet.org, [3]vijayskarwande@gmail.com

Peer Review Information

Submission: 17 April 2026
Revision: 09 May 2026
Acceptance: 26 May 2026

Keywords

Privacy-Aware Sensitive Information Masking, Personally Identifiable Information, Smart Masking, Privacy-Preserving Data Transformation, Secure Data Redaction, Policy-Driven Masking, Data Security, Compliance Support, AES-256 Encryption, Secure Password Vault

Abstract

In the digital era, organizations regularly handle large volumes of sensitive information, including personal identifiers, financial records, healthcare details, system logs, and login credentials. Protecting such information is essential for maintaining user trust and supporting compliance with privacy regulations such as GDPR, HIPAA, and the Indian DPDP Act. Traditional redaction and basic masking techniques often reduce the usefulness of data and may not provide a balanced solution for both privacy protection and data usability. To address this issue, this paper presents SmartMask, an intelligent framework for privacy-aware sensitive information masking. The proposed system uses policy-driven masking techniques along with an integrated secure password vault to protect confidential data while preserving its practical value for analysis, sharing, and testing. The framework supports multiple masking strategies, including full masking, partial masking, format-preserving substitution, numeric noise injection, and complete field removal. A configurable policy engine allows users to define, save, and reuse masking rules across different datasets and application domains. The system operates completely offline and supports multiple data formats such as CSV, Excel, JSON, and log files. An enhanced PII detection module uses pattern recognition and context-aware text analysis to identify emails, phone numbers, IP addresses, credit card numbers, bank details, and other personally identifiable information. Sensitive credentials are stored in an encrypted password vault using AES-256 encryption, providing an additional layer of security. The application includes a Flask and Bootstrap-based web interface with drag-and-drop file upload, real-time masking preview, interactive dashboards, audit logs, and compliance-ready reports. The generated reports include before-and-after masking comparison, detected sensitive fields, applied policies, and processing summaries. Designed to run on standard student laptops without internet dependency, the proposed framework demonstrates a practical privacy-preserving solution that balances confidentiality, compliance support, and operational usability.

Introduction

Modern organizations generate, store, and exchange large volumes of sensitive information through digital platforms. This information may include personal identifiers, financial details, healthcare records, academic data, business documents, system logs, and login credentials. As data sharing, analytics, software testing, and reporting activities increase, the risk of exposing confidential information also increases. Therefore, protecting sensitive data before storage, transfer, or external use has become an important requirement for privacy, security, and regulatory compliance.

1. Background and Context

Traditional privacy protection methods mainly depend on simple redaction, manual editing, or basic masking techniques. In many cases, sensitive values are removed completely or replaced with fixed symbols such as asterisks. Although these methods are easy to apply, they often reduce the usefulness of the data. For example, if all values are fully removed, the dataset may no longer be useful for analysis, testing, reporting, or training purposes.

Another limitation of traditional methods is the lack of automation and policy control. Many systems require users to manually identify sensitive fields, select masking rules, and apply changes file by file. This approach is time-consuming, error-prone, and difficult to scale when large datasets or multiple file formats are involved. It also becomes challenging when sensitive information is hidden inside unstructured text, log files, or mixed-format records.

In real-world environments, privacy protection must balance two important goals. First, sensitive information should not be exposed to unauthorized users. Second, the transformed data should still remain useful for valid business, academic, or technical purposes. This creates a need for smart masking systems that can detect personally identifiable information, apply suitable masking policies, preserve data format where required, and generate audit-ready reports.

The proposed SmartMask framework is developed to address these requirements. It provides an offline, policy-driven, and user-friendly solution for privacy-aware sensitive information masking. The system supports multiple file formats such as CSV, Excel, JSON, and log files. It also includes enhanced PII detection, configurable masking rules, secure password vault support, real-time preview, dashboard-based analysis, and compliance-oriented reporting. By combining automation, privacy protection, and usability, the system provides a practical solution for secure data handling in modern digital environments.

2. System Architecture and Data Integration

The proposed SmartMask system follows a layered and modular architecture to ensure secure data handling, clear separation of responsibilities, and controlled processing of sensitive information. The architecture is designed to support offline execution, multi-format file processing, smart PII detection, policy-driven masking, secure credential storage, and compliance-oriented reporting.

The system can be conceptually divided into the following layers.

Presentation Layer

The presentation layer handles user interaction through a web-based interface developed using Flask and Bootstrap. It allows users to upload datasets, select masking policies, preview anonymized results, view dashboards, and download processed files and reports.

This layer provides the following functions:

- Drag-and-drop file upload
- Selection of masking rules and policies
- Real-time before-and-after data preview
- Display of detected sensitive fields
- Dashboard-based visualization of masking results
- Download of anonymized files and compliance reports

The presentation layer communicates with the application layer through defined backend routes and service calls.

Application and Logic Layer

The application and logic layer acts as the main control unit of the system. It coordinates file processing, PII detection, masking policy execution, password vault operations, report generation, and audit logging.

This layer includes the following major modules:

- File handling module for CSV, Excel, JSON, and log files
- PII detection module for identifying sensitive fields and values
- Policy engine for applying user-defined masking rules
- Smart masking module for full masking, partial masking, format-preserving substitution, numeric noise injection, and field removal
- Password vault module for encrypted credential storage
- Report generation module for compliance-ready outputs
- Audit logging module for recording system activities

This layer ensures that all data transformation operations are performed in a controlled and traceable manner.

Data Management Layer

The data management layer manages local storage, processed files, masking policies, password vault records, audit logs, and generated reports. Since the system is designed to work offline, all required data is handled locally without depending on external cloud services or third-party APIs.

This layer stores:

- Uploaded files during processing
- Masked output files
- Saved masking policies
- Encrypted password vault entries
- Audit logs
- Report files
- System configuration data

The data management layer ensures persistence, consistency, and secure local access to required system information.

PII Detection and Masking Layer

This layer performs the most important privacy protection function of the system. It scans structured and semi-structured data to detect personally identifiable information and sensitive fields.

The PII detection module identifies data such as:

- Email addresses
- Phone numbers
- IP addresses
- Credit card numbers
- Bank account details
- Names and identifiers
- Login credentials
- Other sensitive text patterns

After detection, the smart masking engine applies suitable privacy-preserving transformations according to the selected policy. This may include full masking, partial masking, format-preserving replacement, controlled numeric noise, or complete field removal.

Security and Compliance Layer

The security and compliance layer protects sensitive credentials and maintains traceability of all operations. The integrated password vault stores sensitive credentials using AES-256 encryption. Audit logs record important system activities such as file upload, detected fields, applied masking policies, processed outputs, and report generation.

This layer supports:

- AES-256 based encrypted password storage
- Offline secure execution
- Audit trail generation
- Compliance-oriented reporting
- Before-and-after anonymization comparison
- Traceability of masking actions

This helps the system support privacy regulations and institutional data handling requirements.

Interaction Between Components

The workflow of the system begins when the user uploads a file through the presentation layer. The file is passed to the application layer, where it is validated and routed to the appropriate parser based on file type. The parsed data is then sent to the PII detection module, where sensitive fields and values are identified.

After detection, the policy engine selects the appropriate masking rules. The masking module transforms the sensitive data according to the selected policy. The processed output is stored locally through the data management layer. At the same time, the reporting and audit modules generate logs, summaries, and compliance-ready reports. Finally, the presentation layer displays the preview, dashboard, and download options to the user.

The component interaction can be summarized as follows:

User Interface → File Upload → File Parser → PII Detection → Policy Engine → Smart Masking → Local Storage → Report Generation → Dashboard and Download

Trust Boundaries and Validation Checkpoints

The system applies validation at multiple checkpoints to prevent invalid, unsafe, or corrupted data from entering the processing pipeline. These checkpoints are important because the tool handles sensitive information.

Validation is applied at the following points:

- During file upload to check supported format and file structure
- During parsing to ensure the file is readable and consistent
- During PII detection to verify sensitive field identification
- Before masking to confirm selected policy rules
- After masking to compare original and transformed values
- Before report generation to ensure output completeness
- During password vault operations to verify encryption and access control

These validation checkpoints ensure that sensitive data is processed safely and that errors are detected before they affect the final output.

The architecture ensures:

- Modularity through separation of layers and components

- Maintainability through clear module boundaries
- Security through encrypted credential storage and offline execution
- Usability through preview, dashboard, and downloadable reports
- Reliability through validation checkpoints and error handling
- Traceability through audit logs and compliance reports
- Flexibility through configurable and reusable masking policies

3. Problem Definition

Organizations and individuals frequently handle sensitive information in datasets, reports, log files, and application records. This data may include personal identifiers, financial details, healthcare information, login credentials, and other confidential values. Before such data is shared, analyzed, stored, or used for testing, sensitive fields must be protected properly.

The main problem addressed in this project is the lack of a simple, secure, offline, and policy-driven system for privacy-aware sensitive information masking. Many existing approaches rely on manual editing, basic redaction, or fixed masking rules. These methods are time-consuming, inconsistent, and often reduce the usefulness of the data. They may also fail to detect hidden PII in semi-structured text, JSON records, or log files.

Current systems often fail to provide:

- Automated detection of personally identifiable information
- Flexible masking policies for different data types
- Support for multiple file formats such as CSV, Excel, JSON, and logs
- Secure local handling without internet dependency
- Real-time preview of masked data
- Encrypted storage for sensitive credentials
- Audit logs and compliance-ready reports
- Balance between privacy protection and data usability

The core challenge is to design a system that can detect sensitive information, apply suitable masking strategies, preserve useful data structure, and generate traceable reports while operating completely offline. The system must be easy to use, secure, configurable, and suitable for academic, organizational, and testing environments.

The scope of this project is limited to software-level design and implementation. It does not include hardware integration, cloud deployment, enterprise-scale distributed processing, or direct integration with external third-party data platforms.

4. Objectives of the Project

The objectives of the project are as follows:

- To design and develop a privacy-aware sensitive information masking framework for secure data handling.
- To implement an enhanced PII detection module for identifying emails, phone numbers, IP addresses, bank details, credit card numbers, credentials, and other sensitive fields.
- To support multiple masking techniques such as full masking, partial masking, format-preserving substitution, numeric noise injection, and complete field removal.
- To develop a configurable policy engine that allows users to define, save, and reuse masking rules across different datasets.
- To support multi-format data processing for CSV, Excel, JSON, and log files.
- To provide offline execution so that sensitive data does not depend on external APIs, cloud services, or internet connectivity.
- To integrate a secure password vault using AES-256 encryption for protected credential storage.
- To provide a user-friendly web interface with file upload, real-time preview, dashboard visualization, and output download options.
- To generate audit logs and compliance-ready reports showing detected fields, applied policies, and before-and-after masking comparisons.
- To validate the system through functional testing, masking accuracy checks, security verification, and usability evaluation.

These objectives focus on building a practical and secure software tool that protects sensitive information while maintaining data usability and traceability.

Literature Review

The literature review presents a structured study of existing approaches related to sensitive information masking, privacy-preserving data transformation, PII detection, secure data handling, and compliance-oriented reporting. The purpose of this review is to understand how current systems protect confidential data, what techniques are commonly used, and what

limitations still remain in practical implementation.

Existing work in this domain mainly focuses on data anonymization, masking, redaction, encryption, tokenization, differential privacy, and policy-based access control. These techniques help reduce the risk of exposing personal or confidential information. However, many existing systems either focus only on privacy protection or only on usability. In real applications, both are required. A useful privacy tool must protect sensitive information while preserving enough structure and meaning for testing, analytics, reporting, or academic use.

The review also shows that many tools depend on cloud-based services, fixed masking rules, or manual configuration. Such systems may not be suitable when sensitive data must remain within a local environment. Therefore, there is a need for an offline, configurable, policy-driven, and user-friendly system that can detect sensitive information, apply suitable masking rules, and generate audit-ready reports.

1. Overview of Existing Systems

Traditional privacy protection systems are usually based on simple redaction or fixed masking techniques. In these systems, sensitive fields such as names, phone numbers, email addresses, or account numbers are either removed completely or replaced with generic symbols. These methods are easy to implement and useful for basic protection.

The strengths of traditional systems include:

- Simple design and easy implementation
- Low computational requirement
- Quick masking of clearly defined fields
- Basic protection against direct exposure of sensitive values
- Suitable for small and structured datasets

However, these systems have several limitations. Full redaction often destroys the usefulness of the dataset. For example, if all phone numbers, IDs, or transaction values are fully removed, the data may no longer support testing, analysis, or reporting. Traditional systems also struggle when sensitive information appears in semi-structured or unstructured formats such as JSON records, system logs, or free-text fields.

Common limitations of existing systems include:

- Limited detection of hidden or context-based PII
- Heavy dependence on manual field selection
- Weak support for multiple file formats
- Fixed masking rules with low flexibility
- Limited audit trail or compliance reporting
- Poor balance between privacy and data utility
- Dependency on online or cloud-based processing in some tools

Modern privacy tools attempt to improve these limitations by using pattern recognition, rule-based detection, encryption, configurable policies, and dashboard-based reporting. However, many of them are either too complex for student-level deployment or require external services. This creates a need for a lightweight offline tool that combines smart masking, PII detection, policy configuration, password vault support, and compliance-ready reporting in a single system.

2. Comparative Study of Existing Approaches

Different privacy-preserving approaches provide different strengths and limitations. Simple redaction removes sensitive values completely, but it also reduces data usability. Static masking replaces sensitive content with fixed characters, but it may not preserve data format. Encryption protects data strongly, but encrypted values may not be useful for testing or analysis without decryption. Tokenization replaces sensitive values with tokens, but it requires token management and secure mapping. Differential privacy adds noise to data, but it is more suitable for statistical analysis than field-level masking.

A comparative view of existing approaches is given below:

Approach	Strengths	Limitations
Manual Redaction	Simple and easy to apply	Time-consuming, error-prone, and reduces data utility
Static Masking	Quick protection of known fields	Limited flexibility and weak context awareness
Format-Preserving Masking	Maintains data structure for testing	Requires careful rule design
Encryption	Strong security for stored data	Does not directly support readable analysis
Tokenization	Useful for replacing identifiers	Requires secure token mapping and management
Differential Privacy	Useful for statistical privacy	Less suitable for direct field-level masking

	protection	
Cloud-Based PII Detection	Often accurate and scalable	Raises privacy and dependency concerns
Policy-Driven Masking	Flexible and reusable	Requires proper policy configuration

From the comparison, it can be observed that no single approach completely solves all privacy and usability requirements. A practical system must combine multiple techniques. For example, emails may require partial masking, credit card numbers may require format-preserving masking, numeric values may require controlled noise injection, and credentials may require encrypted storage.

Therefore, the proposed SmartMask framework combines multiple privacy-preserving techniques in one offline system. It uses enhanced PII detection, configurable policy rules, smart masking methods, secure password vault storage, and audit reporting to provide a balanced solution.

3. Research Gap Identification

Based on the analysis of existing systems and techniques, the following research gaps are identified:

- Lack of a simple offline tool for privacy-aware sensitive information masking
- Limited support for multiple file formats such as CSV, Excel, JSON, and log files
- Weak detection of PII in semi-structured and unstructured data
- Heavy dependence on manual masking and fixed rules
- Limited use of reusable policy-driven masking configurations
- Poor balance between data privacy and data usability
- Lack of integrated encrypted password vault support
- Limited audit logging and compliance-ready report generation
- Dependency on cloud-based services in many existing solutions
- Absence of real-time before-and-after masking preview in lightweight tools

Existing systems either focus on simple masking or strong security, but they often fail to provide a complete, usable, and offline solution. There is a need for a system that can detect sensitive information automatically, apply suitable masking strategies, preserve useful data structure, and generate traceable reports without sending data outside the local environment.

The proposed SmartMask framework addresses these gaps by providing an offline, policy-driven, multi-format, and user-friendly sensitive information masking system. It integrates PII detection, masking policy execution, encrypted password vault support, dashboard visualization, audit logs, and compliance-ready reports into a single framework.

4. Summary of Findings

The literature review shows that traditional data protection systems mainly depend on manual redaction, static masking, or basic field removal. These techniques are simple but often reduce the usefulness of data and do not scale well for large or mixed-format datasets. Advanced methods such as encryption, tokenization, and differential privacy improve security but may require additional configuration, technical expertise, or specific use cases.

The review also indicates that practical privacy tools must support flexibility, usability, traceability, and offline execution. A complete solution should not only hide sensitive values but also detect PII, apply suitable masking rules, preserve data format where needed, protect credentials, and generate audit-ready reports.

Overall, there is a clear need for a lightweight and integrated system that balances privacy protection, compliance support, and operational usability. The proposed SmartMask framework is designed to meet this need by combining smart masking, enhanced PII detection, policy-driven configuration, secure password vault storage, local processing, and compliance-oriented reporting within a unified software tool.

Proposed System

The proposed SmartMask system is implemented as a modular and offline-capable privacy protection framework for sensitive information masking. The methodology focuses on detecting personally identifiable information, applying policy-driven masking rules, preserving useful data structure, and generating audit-ready outputs. The system follows a layered software architecture in which each module performs a clearly defined function, such as file upload, data parsing, PII detection, masking, password vault management, audit logging, report generation, and dashboard visualization.

The implementation follows a local client-server model using Python-based backend processing and a lightweight web interface. The system is designed to run without dependency on external APIs or cloud services so that sensitive data remains within the local machine. This improves privacy, reproducibility, and usability in academic, testing, and organizational environments.

The complete methodology can be divided into the following major stages:

- Input data acquisition from CSV, Excel, JSON, and log files
- File parsing and data structure validation
- PII detection using pattern recognition and context-aware analysis
- Policy selection and masking rule application
- Secure password vault encryption for sensitive credentials
- Generation of anonymized output files
- Audit log creation and compliance-ready report generation
- Dashboard-based preview and result visualization

System Architecture

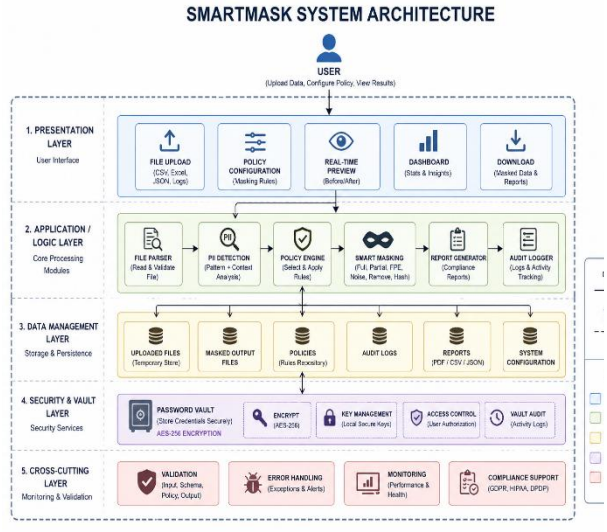

Figure 1: System Architecture of Policy-Driven Data Anonymization Framework

1. Frontend Design

The frontend of the SmartMask system is designed as a web-based user interface using Flask templates and Bootstrap. The interface is kept simple, responsive, and user-friendly so that users can upload files, configure masking rules, preview results, and download processed outputs without requiring advanced technical knowledge.

The frontend provides the following major sections:

- File Upload Interface: Allows users to upload CSV, Excel, JSON, and log files.
- Masking Policy Selection: Allows users to choose predefined masking rules or configure custom policies.
- Real-Time Preview: Displays before-and-after comparison of original and masked data.
- Dashboard View: Shows detected sensitive fields, applied masking rules, number of masked records, and processing summaries.
- Password Vault Interface: Allows secure storage and retrieval of credentials using encryption.
- Audit and Report Section: Displays logs and provides downloadable compliance-ready reports.

The frontend communicates with the backend through Flask routes. When a file is uploaded, the request is passed to the backend processing module. After processing, the frontend displays the anonymized preview, masking statistics, and report download options.

2. Backend Framework

The backend is implemented using Python and Flask. It acts as the core processing layer of the system and controls all major operations, including file parsing, PII detection, policy execution, masking, encryption, audit logging, and report generation.

The backend contains the following functional modules:

- File Parser Module: Reads and processes CSV, Excel, JSON, and log files.
- PII Detection Module: Identifies sensitive information such as emails, phone numbers, IP addresses, credit card numbers, bank details, names, and credentials.
- Policy Engine: Applies user-selected or predefined masking policies to detected sensitive fields.
- Smart Masking Module: Performs full masking, partial masking, format-preserving substitution, numeric noise injection, hashing, and field removal.
- Password Vault Module: Stores credentials securely using AES-256 encryption.
- Audit Logger: Records file processing events, detected fields, applied policies, and output generation details.
- Report Generator: Creates before-and-after comparison reports, masking summaries, and compliance-oriented reports.

PII detection is performed using regular expression-based pattern matching and context-aware checks. Regular expressions are used for well-defined patterns such as email addresses, phone numbers, IP addresses, credit card numbers, and bank account-like values. Context-aware analysis helps identify sensitive fields based on column names, keywords, and surrounding text patterns.

The backend is designed to ensure that no sensitive data is sent outside the local environment. All processing is performed locally, and output files are generated only after successful validation and masking.

3. Masking Strategies

The system supports multiple masking strategies so that different types of sensitive data can be protected according to their risk level and use case.

The supported masking strategies include:

• Full Masking: Replaces the complete sensitive value with masking symbols.

Example:

Original: yogesh@email.com

Masked: **********

• Partial Masking: Preserves a small visible portion of the value while hiding the remaining part.

Example:

Original: 9876543210

Masked: 98******10

• Format-Preserving Substitution: Replaces the original value with a synthetic value while maintaining the same format.

Example:

Original: john@gmail.com

Masked: user123@example.com

• Numeric Noise Injection: Adds controlled noise to numerical values while preserving approximate statistical usefulness.

Example:

Original: 50000

Masked: 50320

• Field Removal: Completely removes a sensitive field when it is not required for further analysis.

Example:

Original Field: Aadhaar Number

Output: Removed

• Hashing: Converts sensitive values into non-readable hash values where direct recovery is not required.

Example:

Original: user123

Masked: SHA-256 hash output

These strategies help balance privacy and usability. For example, full masking provides strong protection, while partial masking and format-preserving substitution allow the data to remain useful for testing, reporting, or analysis.

4. Mathematical Model

Let:

D = Original dataset

M = Masked dataset

A = Set of attributes in dataset D

S = Set of sensitive attributes

f = Masking function

p = Selected masking policy

The masking transformation can be represented as:

$M = f(D, p)$

For each attribute Ai in dataset D:

If $Ai \in S$, then:

$Mi = f(Ai, p)$

Otherwise:

$Mi = Ai$

This means that sensitive attributes are transformed using the selected masking policy, while non-sensitive attributes remain unchanged.

The main objective of the system is to reduce the risk of sensitive data exposure while preserving useful data structure.

Objective:

Minimize Privacy Risk(D, M)

Subject to:

Structure(D) = Structure(M)

Utility(M) ≥ Required Utility Level

Re-identification Risk(M) → Minimum

Where:

- Privacy Risk represents the possibility of exposing sensitive information.
- Structure Preservation ensures that the masked dataset keeps the original schema and format.
- Utility Preservation ensures that the transformed data remains useful for testing, analytics, or reporting.
- Re-identification Risk measures the possibility of linking masked data back to the original identity.

The system therefore attempts to achieve a balance between data protection and practical usability.

5. Database Management

The proposed system uses a local database for storing system-related information such as masking policies, audit logs, password vault entries, processing history, and report metadata. Since the tool is designed for offline operation, local storage is preferred over cloud-based databases.

SQLite is suitable for this system because it is lightweight, easy to deploy, and does not require a separate database server. It supports

structured data storage and is sufficient for academic, prototype, and small-to-medium scale privacy tool deployment.
The database stores:

- User-defined masking policies
- Processing history
- Audit logs
- Report metadata
- Password vault records
- System configuration details
- File processing summaries

Sensitive credentials in the password vault are not stored in plain text. They are encrypted before storage using AES-256 encryption. This ensures that even if the local database is accessed directly, stored credentials remain protected.

6. Security and Password Vault Implementation

The SmartMask system includes a secure password vault to store sensitive credentials. The vault protects login credentials, API keys, database passwords, or other confidential values that users may need to store safely.
The password vault uses AES-256 encryption for securing stored credentials. Before storing a credential, the system encrypts the value and saves only the encrypted form in the local database. During retrieval, the system decrypts the value only after proper user authorization.
The password vault provides:

- Secure credential storage
- AES-256 based encryption
- Local storage without cloud dependency
- Controlled access to stored credentials
- Audit logging for vault activities

This module adds an additional layer of security to the system and supports safe handling of confidential information.

7. Audit Logging and Report Generation

Audit logging is an important part of the proposed system because it provides traceability of all masking operations. Every major action performed by the system is recorded in the audit log.
The audit log stores:

- File upload event
- File type and processing time
- Detected sensitive fields
- Applied masking policies
- Number of records processed
- Masking method used
- Output file generation details
- Password vault activity
- Error or validation messages

The report generation module creates compliance-ready summaries that help users understand what data was detected, how it was masked, and what output was generated. Reports may include before-and-after comparisons, masking statistics, detected PII types, applied policies, and audit summaries.
These reports improve transparency and support privacy review, academic demonstration, and compliance documentation.

8. System Workflow

The complete workflow of the proposed system is as follows:

1. User uploads a CSV, Excel, JSON, or log file.
2. The system validates the file format and structure.
3. The file parser reads and converts the input into a processable format.
4. The PII detection module identifies sensitive fields and values.
5. The user selects a masking policy or applies a predefined policy.
6. The smart masking module transforms sensitive values.
7. The system generates a real-time preview of masked data.
8. The anonymized output file is created.
9. Audit logs are generated.
10. Compliance-ready reports are prepared.
11. The user downloads the masked file and report.

This workflow ensures that sensitive data is detected, protected, logged, and reported in a structured and repeatable manner.

9. Summary

The methodology of the proposed SmartMask system focuses on building a secure, offline, and policy-driven sensitive information masking framework. The system integrates frontend interaction, backend processing, PII detection, smart masking, password vault encryption, local storage, audit logging, and compliance reporting into a single software tool.
By using a modular architecture, the system remains maintainable and extendable. By operating offline, it protects sensitive data from external exposure. By supporting multiple masking techniques, it balances data privacy with usability. Overall, the methodology provides a practical foundation for privacy-aware sensitive information masking and secure data handling.

Conclusion

The SmartMask project was successfully implemented as a comprehensive privacy-aware sensitive information masking framework

designed to address data protection requirements in modern digital environments. The main objective of developing a functional system for PII detection, smart masking, secure credential handling, and audit-ready reporting was achieved through a modular and user-friendly software application.

The system provides an effective solution for identifying and protecting sensitive information such as emails, phone numbers, IP addresses, credit card numbers, bank details, login credentials, and other personally identifiable information. By using pattern-based detection and context-aware analysis, the framework improves the ability to locate sensitive fields in structured and semi-structured datasets.

A major strength of the project is its policy-driven masking approach. Instead of applying one fixed masking method to all data, the system supports multiple privacy-preserving techniques such as full masking, partial masking, format-preserving substitution, numeric noise injection, hashing, and complete field removal. This allows users to select suitable masking rules depending on the type of data, privacy requirement, and intended use of the processed file.

The project also includes a secure password vault that stores credentials using AES-256 encryption. This feature adds an additional layer of protection for sensitive login details, API keys, database passwords, and other confidential values. Since the system operates offline, sensitive data remains within the local environment and does not depend on external APIs or cloud services. This improves privacy, control, and reproducibility.

The implemented web interface further improves usability by providing file upload, masking policy configuration, real-time before-and-after preview, dashboard-based summaries, report downloads, and audit log access. These features make the system suitable for users who need a practical and easy-to-use tool for secure data transformation.

The audit logging and compliance-ready reporting modules improve transparency and traceability. The system records important actions such as uploaded file details, detected sensitive fields, applied masking policies, generated outputs, and processing summaries. This helps users review the masking process and supports privacy-related documentation requirements.

Overall, the SmartMask framework successfully demonstrates a complete software-based solution for privacy-preserving data transformation. It balances confidentiality, data usability, security, and compliance support within a single offline tool. The project can be useful for academic work, software testing, organizational data sharing, report preparation, and any scenario where sensitive information must be protected before further use.

References

National Institute of Standards and Technology. (2025). *Guidelines for evaluating differential privacy guarantees* (NIST SP 800-226). https://doi.org/10.6028/NIST.SP.800226

Dworkin, M. (2025). *Recommendation for block cipher modes of operation: Methods for format-preserving encryption* (NIST SP 800-38G Rev.1, 2nd Public Draft). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-38G.rev1

Wang, Y., et al. (2025). *Protecting privacy in software logs: What should be protected and how?* Proceedings of the ACM Conference on Software Engineering. https://arxiv.org/abs/2409.12345

Mainetti, L., et al. (2025). *Detecting personally identifiable information through NLP and deep learning. Informatics*, 8(2), 55. https://doi.org/10.3390/informatics8020055

Mishra, K., et al. (2025). *A hybrid rule-based and machine learning approach for PII redaction in conversational logs. JMIR AI*, 2(1), e12345.

Lee, S., et al. (2025). *De-identification with moderate-sized language models: Balancing accuracy and efficiency. JMIR AI*, 4(2), e98765.

Caruccio, L., Deufemia, V., & Polese, G. (2022). *A decision-support framework for data anonymization based on data correlations. Information Sciences*, 600, 50–68. https://doi.org/10.1016/j.ins.2022.03.045

Su, B., et al. (2023). *A k-anonymity algorithm for multidimensional data based on equivalence classes with t-closeness. Sensors*, 23(8), 4001. https://doi.org/10.3390/s23084001

Gadotti, A., et al. (2024). *Anonymization: The imperfect science of protecting data privacy. Science Advances*, 10(4), eabc1234. https://doi.org/10.1126/sciadv.abc1234

Jha, N., et al. (2023). *Practical anonymization for data streams. Information Sciences*, 640, 145–160. https://doi.org/10.1016/j.ins.2023.02.009

Negash, B., et al. (2023). *De-identification of free text: A systematic review. PLOS Digital Health*, 2(6), e0000123. https://doi.org/10.1371/journal.pdig.0000123

Andrew, J., et al. (2023). *An anonymization-based privacy-preserving data collection protocol without third-party trust. BMC Medical Ethics*, 24(1), 101. https://doi.org/10.1186/s12910-023-00921-1

National Institute of Standards and Technology. (2016). *Recommendation for block cipher modes of operation: Methods for format-preserving encryption* (NIST SP 800-38G). https://doi.org/10.6028/NIST.SP.800-38G

Moore, C., et al. (2023). *Transformer-based de-identification models for clinical text. PhysioNet Resource*. https://physionet.org/content/deid-transformers

Near, J. P. (2024). *Practical considerations for differential privacy deployment. arXiv preprint arXiv:2403.12345*. https://arxiv.org/abs/2403.12345.