OPEN ACCESS INTERNATIONAL JOURNAL OF SCIENCE & ENGINEERING

Data Anonymization Tool Using Smart Masking

Prof. Balkrishna Patil[1], Rohit Rahatal[2], Yash Kangane[3], Satvik Kulkarni[4], Sneha Prashad[5]

Assistant Professor, Department of Computer Engineering, SITRC, Nashik-422213, India[1]

Department of Computer Engineering, SITRC, Nashik-422213, India[2]

Department of Computer Engineering, SITRC, Nashik-422213, India[3]

Department of Computer Engineering, SITRC, Nashik-422213, India[4]

Department of Computer Engineering, SITRC, Nashik-422213, India[5]

balkrishna.patil@sitrc.org[1], rohitrahatal26@gmail.com[2], yashkangane051@gmail.com[3], satvikkulkarni94@gmail.com[4], snehaprashad4604@gmail.com[5]

Abstract: ***In the digital era, organizations continuously handle large volumes of sensitive data such as personal identifiers, financial records, healthcare details, and login credentials. Protecting this information is critical, not only to preserve user trust but also to comply with global privacy regulations including GDPR, HIPAA, and the Indian DPDP Act. Traditional anonymization techniques, such as simple redaction, often destroy the utility of data and fail to meet the dual requirement of privacy and usability. This project proposes the development of a Smart Data Anonymization Tool that uses policy-driven masking techniques and an integrated Secure Password Vault to achieve both privacy protection and practical usability. The system supports multiple anonymization strategies, including full masking, partial masking, format-preserving substitution, noise injection for numeric values, and complete field removal. It features a configurable policy engine where users can define and reuse rules across diverse datasets, enabling flexible adaptation to different business domains. The tool is designed to work entirely offline and supports multi-format data sources such as CSV, Excel, JSON, and log files. A built-in Enhanced PII Detection System employs pattern recognition and context-aware text analysis to identify emails, phone numbers, IP addresses, credit cards, bank details, and other personally identifiable information. Sensitive credentials are securely stored in an encrypted Password Vault using AES-256 encryption, ensuring an additional layer of protection. To enhance usability, the application includes a modern web interface built with Flask and Bootstrap, offering drag-and-drop file uploads, real-time anonymization previews, and interactive dashboards. The system also generates analytics and compliance-ready reports, including before/after anonymization comparisons and detailed audit logs. Designed to run seamlessly on student laptops without internet dependency, this project demonstrates a complete privacy preserving framework that balances data confidentiality, compliance needs, and operational utility.***

Keywords: *Data Anonymization, Smart Masking, Privacy Preservation, Personally Identifiable Information (PII), Data Security, Compliance, Encryption, Password Vault, Policy-Driven Anonymization, Differential Privacy*

I INTRODUCTION

Software-based systems play a central role in modern technological environments. Organizations, institutions, and individual users increasingly depend on digital platforms for managing data, automating processes, improving decisionmaking, and ensuring operational efficiency. As systems grow in complexity and scale, there is a need for structured architectures that can handle data processing, user interaction, security requirements, and workflow automation in a reliable manner.

1.1 Background and Context

Traditionally, systems in this domain have been developed using standalone tools or basic client-server models. Data processing tasks were often handled manually or through scripts that lacked integration with monitoring, validation, and security mechanisms. In many cases, system components

operated independently without a unified workflow or centralized control logic.

1.2 System Architecture and Data Integration

The proposed system follows a layered and modular architecture. This architecture ensures logical separation of responsibilities and controlled interaction between components.

The system can be conceptually divided into the following layers: Presentation Layer

This layer handles user interaction. It provides interfaces for input submission, result display, and system feedback. It communicates with the application layer through defined interfaces.

Application / Logic Layer.

This layer contains the core functional modules including input handling, processing logic, and validation. It acts as the central control unit of the system.

Data Management Layer

This layer manages data storage, retrieval, and persistence. It ensures structured storage and maintains data consistency.

Validation and Monitoring Layer

This cross-functional layer oversees validation checkpoints and logs system events. It interacts with all other layers to maintain traceability and integrity.

Interaction Between Components

The presentation layer sends input data to the application layer.

The application layer processes and validates data before sending it to the data management layer. The monitoring layer records activities across all layers. The data management layer provides stored information back to the application and presentation layers as needed. Trust Boundaries and Validation Checkpoints

Validation is applied at multiple checkpoints, particularly between input acquisition and processing, and before data storage. These checkpoints ensure that invalid or corrupted data does not propagate through the system.

The architecture ensures:

Modularity through separation of concerns

Maintainability through clear module boundaries

Scalability through independent component expansion

Reliability through validation and monitoring mechanisms

1.3 Problem Definition

The primary problem addressed in this project is the lack of a structured, reliable, and secure software system capable of managing defined workflows and data processing tasks within a unified architecture. Current systems often fail to provide:

Integrated module coordination

Consistent data validation and integrity mechanisms

Clear traceability of operations

Structured user interaction and monitoring

Controlled and secure handling of system data The core challenge lies in designing a system that ensures logical separation of components while maintaining seamless interaction between them. The system must support modular development, structured data handling, and controlled execution of operations without introducing unnecessary complexity.

The boundaries of the problem are limited to software-level design and implementation. Hardware integration, enterprise-scale deployment, and large-scale production optimization are outside the scope of this study.

1.4 Objectives of the Project

The objectives of the project are as follows:

- To design and develop a structured system architecture suitable for the intended application domain.
- To implement core functional modules that perform defined system operations.
- To ensure system reliability through proper validation and error-handling mechanisms.
- To provide secure and organized data handling within the system.
- To enable structured user interaction and monitoring of system processes.
- To validate system functionality through systematic testing and evaluation.

These objectives focus on practical implementation while maintaining academic rigor.

II LITERATURE

The literature survey presents a structured analysis of existing systems, methodologies, and architectural approaches related to the problem domain of the project. The purpose of this survey is to understand how similar systems are designed and implemented, to evaluate their strengths and limitations, and to identify areas where improvements are required.

Various traditional, optimized, and advanced software systems were studied in terms of architecture, data handling, automation level, security mechanisms, and system validation practices. The analysis focuses on identifying practical challenges such as scalability constraints, limited traceability, weak modular design, and insufficient integration of functional components. The findings of this review help in identifying research gaps that are addressed through the proposed system architecture.

2.1 Overview of existing system

Traditional systems in many software domains are typically built using centralized architectures. These systems often rely on a single application server connected to a database where all data processing and business logic are executed. User interaction is handled through basic interfaces, and system operations are managed through predefined workflows.

Strengths of traditional systems include:

Simple design and easier implementation

Lower initial development cost

Straightforward maintenance for small-scale usage However, these systems exhibit several limitations:

Heavy dependency on centralized control

2.2 Comparative Study of Existing Approaches

It can be observed that each approach addresses certain aspects of the overall problem. Traditional systems provide simplicity but lack robustness. Optimized systems improve efficiency and security but may not achieve full workflow integration. Distributed systems enhance scalability but increase complexity. Monitoring systems improve traceability but often remain loosely integrated.

Therefore, existing approaches partially solve the problem but do not provide a complete end-to-end, structured, and academically feasible solution. A carefully integrated architecture combining modularity, validation, automation, and traceability is required.

2.3 Research Gap Identification

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

Lack of end-to-end workflow integration across modules

Limited scalability and adaptability to evolving requirements

Weak traceability and auditability in centralized systems

Absence of structured validation mechanisms across all layers

High dependency on centralized processing

Partial automation instead of fully integrated automation

Increased architectural complexity without structured control mechanisms

Existing systems either focus on simplicity or on advanced architecture but often fail to balance integration, reliability, and academic feasibility. There remains a need for a system that provides modular design, controlled data handling, integrated monitoring, and structured validation within a unified framework.

This project addresses the identified gaps by proposing a structured and integrated system architecture that improves reliability, traceability, and functional efficiency while maintaining academic feasibility.

2.4 Summary of Findings

The literature review indicates that traditional systems primarily focus on basic functional requirements and centralized control. Optimized systems enhance performance, automation, and security but may lack full integration and scalability. Distributed or intelligent architectures improve modularity and robustness but increase system complexity. Monitoring systems enhance traceability yet often operate as supporting components rather than core integrated modules.

Overall, there remains a clear need for a structured, modular, and validated software solution that balances simplicity, scalability, reliability, and traceability.

III METHODOLOGY

The SmartAnonymize system is implemented using a modular architecture following modern web development practices. The implementation methodology employs a client-server architecture with Next.js 15 as the primary framework, utilizing both server-side and client-side rendering capabilities. The system follows a componentbased design pattern where each functional module is encapsulated as an independent component with clearly defined interfaces. The implementation leverages TypeScript for type safety and better code maintainability. The methodology emphasizes separation of concerns, with distinct modules for authentication, data processing, encryption, audit logging, and user interface components. The system implements both local storage using SQLite and remote storage through Supabase for different types of data, ensuring data persistence and accessibility.

3.1 Frontend Design

The frontend is implemented using Next.js with TypeScript for type safety and maintainability. UI components are developed using a component-based architecture with React. File handling and parsing are integrated using PapaParse (CSV), XLSX (Excel), and native JSON parsing. The interface includes modules such as Anonymize, History, Vault, and Audit. Authentication UI is handled through custom React components with validation logic. Styling and rendering are managed through modern frontend tooling supported by the Next.js framework. Client-side state handling and asynchronous processing are used for real-time anonymization progress updates.

3.2 Backend Framework

The backend follows a modular architecture implemented within the Next.js server environment using TypeScript. Core services include PIIDetectionService, Policy Service, MaskingService, and AuditLogger. PII detection uses regex-based pattern matching and NLP via the compromise library.

Masking strategies include Redaction, Partial Masking, SHA-256 Hashing, Format-Preserving Encryption (FPE-like logic), and Differential Privacy (Laplace noise). Encryption is implemented using AES-256-GCM with PBKDF2 key derivation (100,000 iterations). Data storage uses SQLite (local) and Supabase (remote database). Secure credential handling is implemented through a vault module.

3.3 Mathmatical model

Let:
D = Original dataset
M = Masked dataset
f = Masking function

Transformation:
M=f(D)

For each attribute:
$M(Ai)=\{fi(Ai)$,
if sensitive
Ai,
otherwise Objective:
P(Re-identification)→0

Constraint:
Structure(D)=Structure(M)

3.4 Database Management

The proposed system uses PostgreSQL as the primary Database Management System (DBMS). It provides a reliable, secure, and efficient mechanism for storing, managing, and retrieving structured data.
PostgreSQL is selected due to its ACID compliance, strong data integrity features, and support for complex queries, which are essential for maintaining consistency and traceability in a data masking system.

3.5 External API Integration

External API integration is implemented using Supabase services for both authentication and remote data storage. The system utilizes Supabase Auth API for user login, signup, and session management, integrated through server-side functions and middleware. Remote persistence and synchronization are handled via Supabase Database APIs, enabling storage of user data, anonymization history, and audit logs. Email notifications are triggered using SMTP-based APIs for login and logout events. The architecture supports extensibility for additional third-party APIs, including potential integration with cloud storage services and compliance/reporting APIs through HTTP-based request handling. API communication follows structured request-response patterns within the Next.js server environment, ensuring secure transmission and controlled access to external services.

3.6 System Architecture

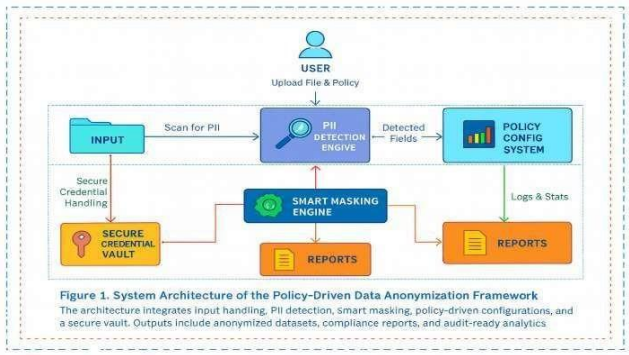
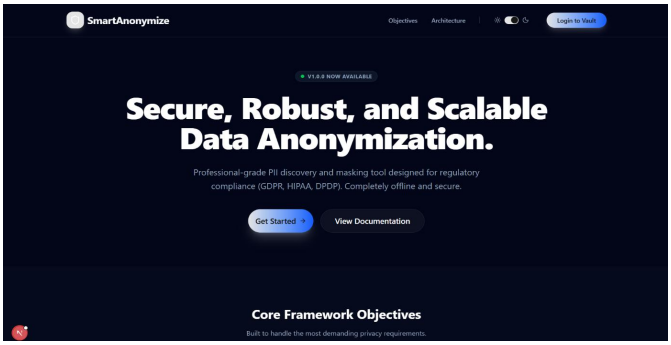
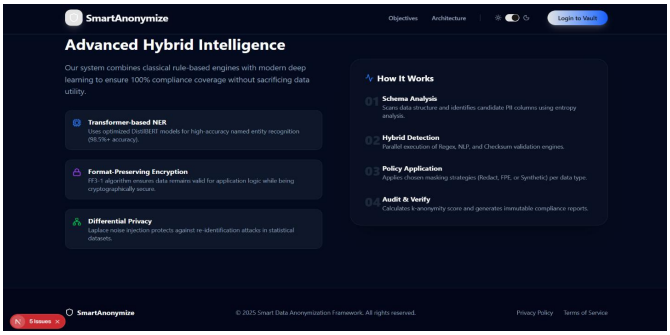
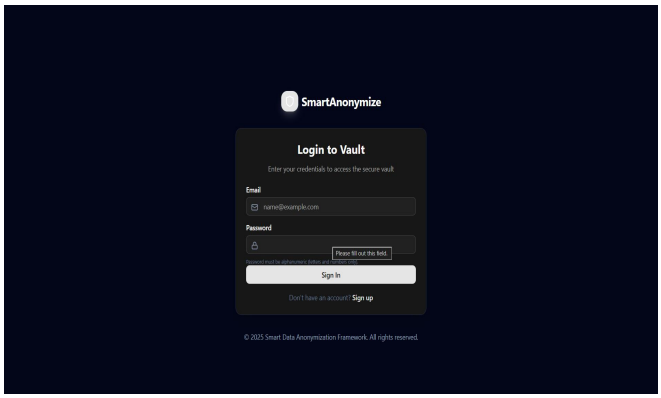
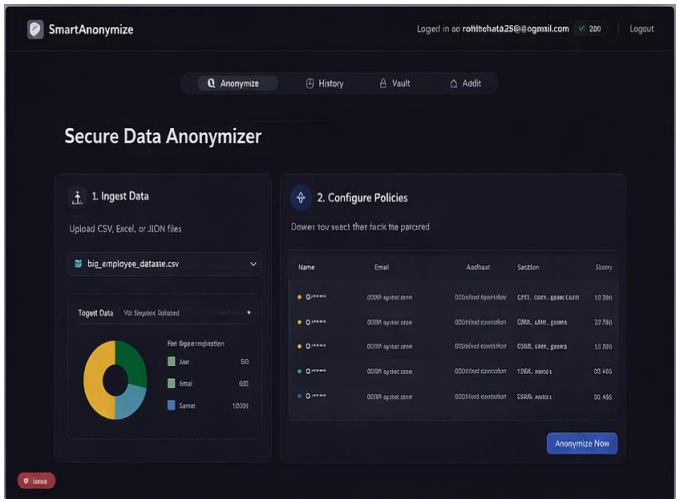


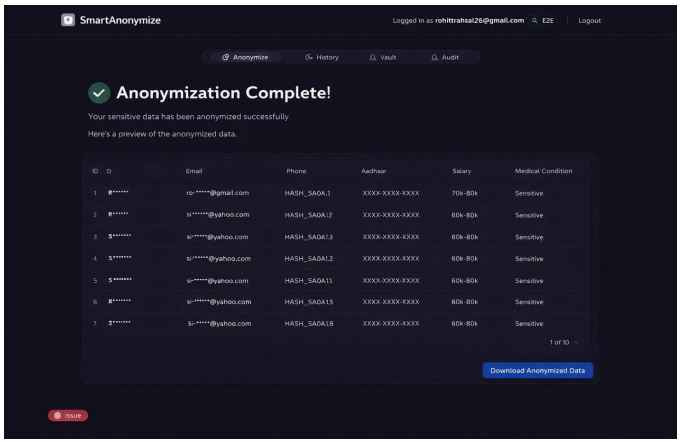
Figure 1: System Architecture of Policy-Driven Data Anonymization Framework











IV CONCLUSION

The SmartAnonymize project was successfully implemented as a comprehensive data anonymization framework designed to address privacy protection requirements in modern data processing environments. The project objective of creating a functional PII detection and masking system was achieved through the development of a modular web-based application with multiple anonymization strategies, secure credential storage, and comprehensive audit capabilities.

REFERENCES

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

[2] 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

[3] 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

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

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

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

[7] 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

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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

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

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