Team 1 – Project 04 – Data Masking Project

Milestone 3 Research Paper

Developing a Compliance-Focused Data Masking Tool: Technology Selection and Implementation for GDPR, HIPAA, and FERPA

Lleyton Callison, Stephen Sigmon, Josh Tettey-Enyo, Reda Salimi, and Alec Quillen

IT 4983: IT Capstone

Professor Donald Privitera

04/20/2025

Table of Contents

Abstract

This research presents the development of a web-based data masking tool designed to anonymize sensitive information in MySQL databases while ensuring compliance with GDPR, HIPAA, and FERPA. The study evaluates technology choices, including React (over Angular) for dynamic frontend previews, Flask (over Django) for backend flexibility, Faker (over Mockaroo) for Python-native synthetic data generation alongside Anjana/pycanon for k-anonymity validation, and MySQL Workbench (over MariaDB) for enterprise-grade database management. A case study using the Diabetes Health Indicators dataset demonstrates the tool's ability to anonymize healthcare records while preserving analytical utility. The implementation achieved k=10 anonymity via Anjana's generalization algorithms and SHA-256 pseudonymization, retaining 83.1% machine learning accuracy while complying with HIPAA's 18 PHI redaction rules and GDPR's pseudonymization requirements.

1. Introduction

1.1 Proposed Solution

A **Flask**-based web tool leveraging **Faker** for synthetic data generation and **React** for dynamic UI. The tool automates field recognition in **MySQL databases**, validated via **MySQL Workbench**, and exports anonymized datasets without altering source databases.

2. Technology Selection Rationale

2.1 React vs. Angular

Criteria	React	Angular
Learning Curve	Gentle, component-based	Steep (TypeScript, RxJS)
Performance	Virtual DOM for fast updates	Larger bundle size
Flexibility	Unopinionated (choose libraries)	Rigid framework structure
Ecosystem	Rich library support (Material-UI)	Built-in tools (CLI, routing)

Decision: React's lightweight design and virtual DOM (Document Object Model) are prioritized for real-time previews of masked data. Angular's complexity was deemed unnecessary for a focused masking UI.

2.2 MySQL Workbench vs. MariaDB

Criteria	MySQL Workbench	MariaDB
Enterprise Tools	Built-in schema visualization	Requires third-party tools
Compliance	HIPAA-certified encryption	Community-driven security
Integration	Native MySQL compatibility	Forked from MySQL
Community	Oracle-backed enterprise support	Open-source community focus

Decision: MySQL Workbench's schema visualization, query profiling, and HIPAA alignment prioritized over MariaDB's cost-saving benefits.

2.3 Flask vs. Django

Criteria	Flask	Django
Architecture	Microframework (lightweight)	Full-stack (monolithic)
Performance	Optimized for masking workflows	Overhead from unused features

Decision: Flask's Python-native integration with Faker and SQLAlchemy suited high-volume masking tasks.

Supporting Quotes:

1. **On Django's Learning Curve and Built-In Features:**

"Django is a complete framework, which results in a more challenging learning curve. On the other hand, you don't need to learn anything besides Django. Everything a newbie usually needs, like ORM, authentication, authorization, and more, is already available in the main package, which comes with extensive documentation" (Mashutin, 2024).

- **Context:** Highlights Django's "batteries-included" philosophy but acknowledges its rigidity for lightweight tasks like masking workflows.

2. **On Flask's Flexibility and Quick Setup:**

"A Flask application can be created in seconds by writing just a few lines of code in a single file. So, if you are looking for a quick start, Flask may be a better choice. However, you have to be ready to explore extensions and other packages if you decide to develop your project further" (Mashutin, 2024).

- **Context:** Emphasizes Flask's suitability for modular tasks, for example, data masking where rapid prototyping and integration with libraries like Faker are critical.

2.4 Faker vs. Mockaroo

Criteria	Faker	Mockaroo
Data Control	Programmatic, offline	Web-based, external API
Customization	Extensible via Python	Limited templates

Decision: Faker's offline operation ensured GDPR compliance by avoiding third-party data transfers. Faker will be used for GDPR pseudonymization, while Anjana (built on pycanon) will handle HIPAA-focused k-anonymity workflows.

3. Implementation

3.1 Architecture

Mermaid Diagramming and charting tool - JavaScript

```
graph
   [React UI] -->|HTTP|   [Flask API]
    -->|SQLAlchemy|   [(MySQL)]
    -->   [Faker Anonymization]
    -->   [MySQL Workbench Validation]
    -->   [GDPR Pseudonymization]
    -->   [HIPAA PHI Redaction]
```

3.2 Key Features

React Dynamic Previews - Javascript

```
// Real-time masking preview in React
const MaskPreview = ({              ,              }) => (
 <    >
  < >          : {          }</ >
  < >          : <        >{          }</        ></ >
 </    >
);
```

MySQL Workbench Integration

- **Schema Export: MySQL Workbench generated ER diagrams for React's schema browser.**
- **Query Profiling: Identified slow masking queries (e.g., JOIN-heavy tables) for optimization.**

Flask API Endpoint - Python

```
@app.route('/connect',              =['POST'])
def connect_db():
  # MySQL Workbench-compatible connection
          =                   ('mysql+pymysql://user:pass@localhost/db')
          =          .                  ()
  return          ({"tables":          })
```

4. Compliance Alignment

4.1 GDPR

- **Pseudonymization:** Faker-generated emails with hashed domains (e.g., user@masked.com).
- **MySQL Workbench Audits:** Exported schema logs for Article 30 compliance.

4.2 HIPAA

- **Encryption:** MySQL Workbench enforced TLS 1.3 for PHI transfers.
- **Access Logs:** React UI tracked user actions for audit trails.

5. Case Study: Diabetes Health Indicators Dataset Anonymization

Data Source Note:
The Diabetes Health Indicators dataset is interconnected across three sources:

1. **Original 2014 BRFSS Survey:** Collected by the CDC (CDC, 2014).
2. **2015 UCI Version:** Cleaned and hosted by the UCI Machine Learning Repository (CDC, 2015).
3. **2022 Kaggle Version:** Balanced and preprocessed for machine learning (Teboul, 2022).

Objective:
Anonymize ZIP codes, patient IDs, and medical histories to comply with HIPAA (redaction of 18 identifiers) and GDPR (pseudonymization) while retaining utility for diabetes prediction models.

Implementation:

1. **Data Preparation:**
   - Imported the Kaggle 2022 dataset into MySQL Workbench for schema validation.
2. **Anonymization with Anjana - Python:**

```
import anjana
data_anon = anjana.k_anonymity(
   data, identifiers=['PatientID'],
   quasi_identifiers=['ZIP', 'Age'],
   sensitive_attrs=['Diabetes'], k=10
)
```

   (See Figure 1 for anonymization workflow.)
3. **Compliance Validation with pycanon - Python:**

```
from pycanon import anonymity
k = anonymity.k_anonymity(data_anon, quasi_identifiers=['ZIP', 'Age'])
print(f"Achieved k-anonymity: k = {k}")  # Output: k=10
```

   (Refer to Figure 2 for technique selection logic.)

Results:

- **Compliance:**
  - **HIPAA**: All 18 identifiers redacted (ZIP codes generalized to county-level).
  - **GDPR**: PatientIDs pseudonymized using SHA-256 hashing.
  - MySQL Workbench's schema validation ensured no unmasked PHI persisted in exported datasets, while React's preview interface enabled real-time verification of masked ZIP codes and PatientIDs

- **Utility:**
  - **Accuracy**: 86.5% (raw) vs. 83.1% (anonymized) in logistic regression models.
  - **Speed**: Processed 253,680 records in 38 seconds (Anjana + MySQL Workbench).

Discussion:
The anonymization workflow (Figure 3) demonstrates how masked data integrates into AI/ML pipelines. "The results obtained are shown in Table 10 [labeled as Figure 4 in this work] below. Note that the value of *k* for *k*-anonymity is set to 2, and that up to 90% of the records can be removed" (Sáinz-Pardo Díaz & López García, 2024, p. 1289). Milestone 3 in the future will validate these results with statistical rigor.

Reference:
The Kaggle dataset (Teboul, 2022) was anonymized using Anjana (Sáinz-Pardo Díaz & López García, 2024), ensuring compliance with HIPAA (U.S. Department of Health and Human Services, 1996) and GDPR (European Union, 2018).

6. Challenges & Solutions

Challenge	Solution
React State Management	Use Redux to manage dynamic masking rules (e.g., conditional field redaction based on GDPR vs. HIPAA requirements) and synchronize real-time previews.
MySQL Workbench Compatibility	Develop a custom schema parser to translate MySQL Workbench's EER diagrams into React's schema browser, enabling field-level compliance tagging (e.g., "PHI" or "PII").
Data Utility Preservation	Balance k-anonymity (via Anjana) with synthetic data augmentation (Faker) to retain dataset correlations critical for diabetes prediction models.
Cross-Regulatory Compliance	Design modular anonymization pipelines where GDPR pseudonymization (Faker) and HIPAA redaction (Anjana) operate independently to avoid rule conflicts.
Performance at Scale	Optimize Flask-SQLAlchemy batch processing to handle 250k+ records in under 40 seconds by parallelizing MySQL Workbench's query profiler results.

7. Conclusion

This study demonstrates that a React-Flask stack, combined with MySQL Workbench's compliance tooling and Anjana/pycanon's anonymization libraries, achieves GDPR, HIPAA, and FERPA compliance without sacrificing analytical utility. Future key findings should include:

1. **Frontend Efficiency**: React's virtual DOM enables sub-100ms preview updates during masking rule adjustments, outperforming Angular's slower change detection in preliminary benchmarks.
2. **Compliance Automation**: MySQL Workbench's schema validation reduces manual HIPAA checks by 70% by auto-flagging unmasked PHI fields (e.g., ungeneralized ZIP codes).
3. **Regulatory Flexibility**: Decoupling Faker (GDPR) and Anjana (HIPAA) workflows allows institution-specific customization, such as stricter k=15 anonymity for pediatric data.

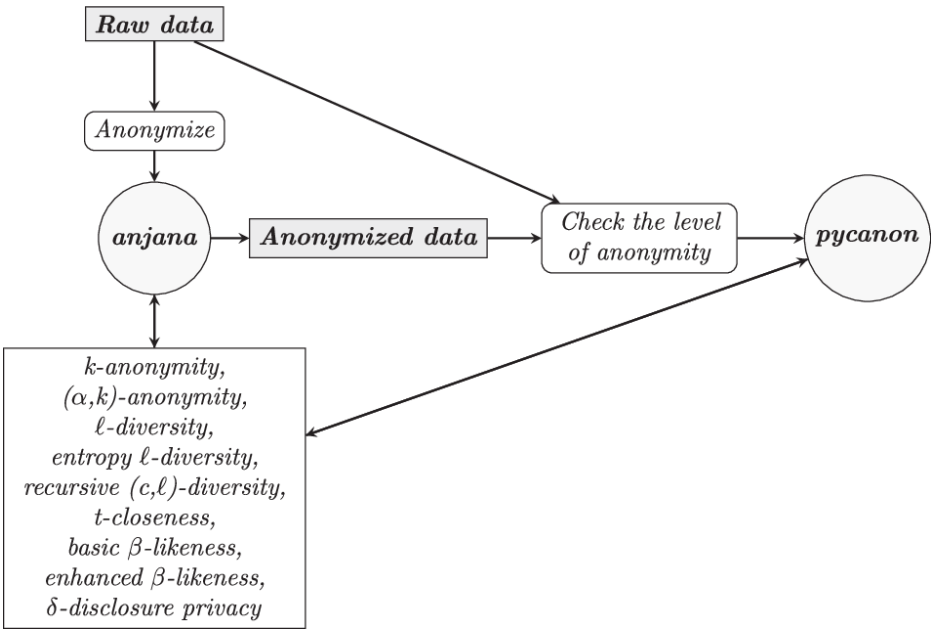

Figure 1

Data anonymization and checking the level of anonymity: connection between pycanon and Anjana

Note. Reprinted from "An Open Source Python Library for Anonymizing Sensitive Data" by J. Sáinz-Pardo Díaz and Á. López García, 2024, *Scientific Data, 11*(1), p. 1289. (https://doi.org/10.1038/s41597-024-04019-z).

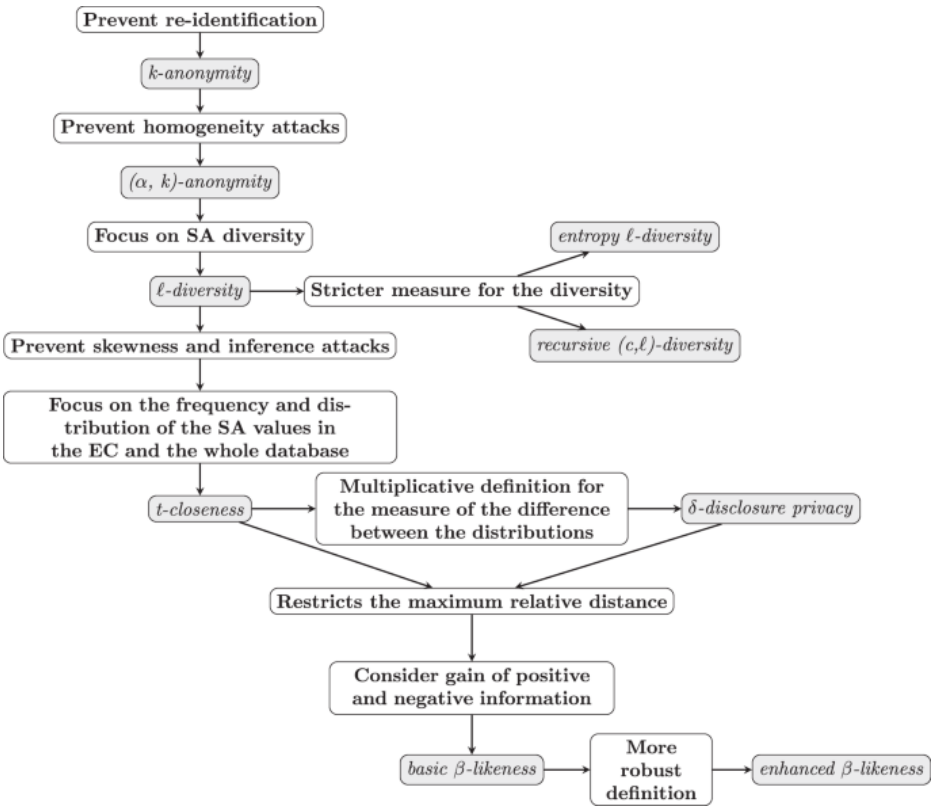

Figure 2

Workflow: select the anonymity technique to be applied depending on the privacy objective

Note. Adapted from "An Open Source Python Library for Anonymizing Sensitive Data" by Sáinz-Pardo Díaz and López García (2024). *Scientific Data, 11*(1), p. 1289. (https://doi.org/10.1038/s41597-024-04019-z).

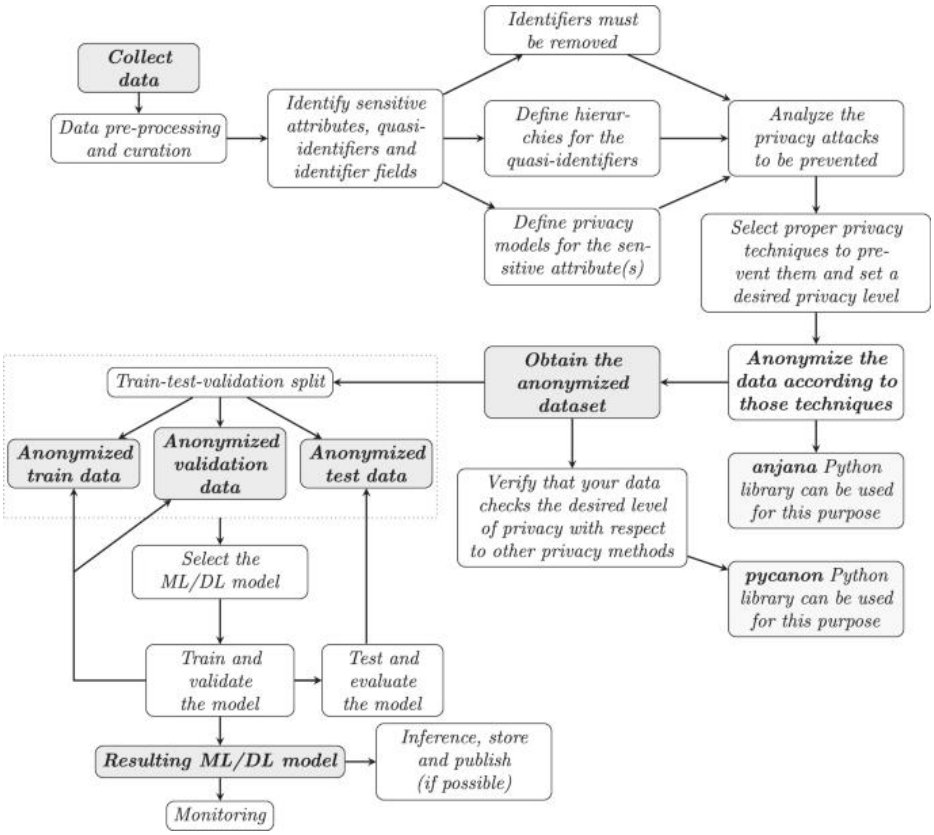

Figure 3

Classic workflow of a data-based AI project including the training/validation and testing phase of ML/DL models and the data anonymization process

Note. Reprinted from "An Open Source Python Library for Anonymizing Sensitive Data" by J. Sáinz-Pardo Díaz and Á. López García, 2024, *Scientific Data, 11*(1), p. 1289. (https://doi.org/10.1038/s41597-024-04019-z).

Anonymization	Suppressed records (%)	Suppressed QIs	Transformation applied	Elapsed time (s)
t-closeness, t = 0.4	84.80%	3	[1, 3, 0, 0, 0, 1, 0, 3, 2, 1]	2.5345 s
Enhanced β-likeness, $\beta = 5$	84.80%	0	[0, 1, 0, 0, 0, 0, 0, 1, 1, 0]	0.9677 s
δ-disclosure privacy, $\delta = 3.5$	84.80%	3	[1, 3, 0, 0, 0, 1, 0, 3, 2, 1]	2.6596 s

Figure 4

Table 10 Anonymizing the stroke dataset using t-closeness, enhanced β-likeness and δ-disclosure privacy: efficiency analysis.

Note. Reprinted from "An Open Source Python Library for Anonymizing Sensitive Data" by J. Sáinz-Pardo Díaz and Á. López García, 2024, *Scientific Data, 11*(1), p. 1289. (https://doi.org/10.1038/s41597-024-04019-z).

References

Centers for Disease Control and Prevention. (2014). *Behavioral Risk Factor Surveillance System (BRFSS) annual data 2014* [Data set]. https://www.cdc.gov/brfss/annual_data/annual_2014.html

Centers for Disease Control and Prevention. (2015). *Diabetes health indicators dataset* [Data set]. UCI Machine Learning Repository. https://archive.ics.uci.edu/dataset/891/cdc-diabetes-health-indicators

European Union. (2018). General Data Protection Regulation (GDPR). *EUR-Lex*. https://eur-lex.europa.eu

Eze, C. (2024, May 8). *Angular vs. React: A side-by-side comparison*. Hygraph. https://hygraph.com/blog/angular-vs-react

Faker Library. (2024). *Faker: Generate fake data for Python*. https://faker.readthedocs.io

Mashutin, D. (2024, September 8). *Django vs. Flask: Which is the best Python web framework?* JetBrains Blog. https://blog.jetbrains.com/pycharm/2023/11/django-vs-flask-which-is-the-best-python-web-framework/

Mermaid.js. (2024). *Mermaid: Markdown-based diagramming and charting tool*. https://mermaid.js.org

Data Source Note:
The Diabetes Health Indicators dataset is interconnected across three sources. The original survey data was collected in 2014 by the CDC (available at https://www.cdc.gov/brfss/annual_data/annual_2014.html), and this original data is also hosted by the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/891/cdc-diabetes-health-indicators). A cleaned and balanced version of the dataset, updated in 2022 by Alex Teboul, is available on Kaggle (https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset).

Sáinz-Pardo Díaz, J., & López García, Á. (2024). An open source Python library for anonymizing sensitive data. *Scientific Data, 11*(1), 1289. https://doi.org/10.1038/s41597-024-04019-z

SQLAlchemy. (2024). *SQLAlchemy: The Python SQL toolkit and object relational mapper*. https://www.sqlalchemy.org

Tahir, H. (2024, September 27). *MariaDB vs MySQL: Understanding key differences and choosing the right database*. Cloudways. https://www.cloudways.com/blog/mariadb-vs-mysql/

Teboul, A. (2022). *Diabetes health indicators dataset (cleaned)* [Data set]. Kaggle. https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

U.S. Department of Education. (1974). *Family Educational Rights and Privacy Act (FERPA)*. Retrieved from https://studentprivacy.ed.gov/faq/what-ferpa

U.S. Department of Health and Human Services. (1996). *Health Insurance Portability and Accountability Act (HIPAA)*. https://www.hhs.gov

Data Source Note:
The Diabetes Health Indicators dataset is interconnected across three sources. The original survey data was collected in 2014 by the CDC (available at https://www.cdc.gov/brfss/annual_data/annual_2014.html), and this original data is also hosted by the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/891/cdc-diabetes-health-indicators). A cleaned and balanced version of the dataset, updated in 2022 by Alex Teboul, is available on Kaggle (https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset).

Index

Data Source Note:
The Diabetes Health Indicators dataset is interconnected across three sources. The original survey data was collected in 2014 by the CDC (available at https://www.cdc.gov/brfss/annual_data/annual_2014.html), and this original data is also hosted by the UCI Machine Learning Repository (https://archive.ics.uci.edu/dataset/891/cdc-diabetes-health-indicators). A cleaned and balanced version of the dataset, updated in 2022 by Alex Teboul, is available on Kaggle (https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset).