Product Specification

Anonymization as a Service

08.02.2019

Group 8

Members	Student Number	Email
Jeremiah Augie Salita Uy	s181369	jeremiah.uy@outlook.com
Lord André Groseth	s181365	andregroseth@outlook.com
Sondre Halvorsen	s305349	sondre.hal@gmail.com
Julian Sagen	s315584	julian.sagen@gmail.com
Viktor Vartdal Johansen	s315615	viktor.v.johansen@hotmail.com

Table of Content

Table of Content 1

1. Presentation 3

Client 3

About the Client 3

Person of Contact 3

Client Stakeholders 4

Supervisor from OsloMET 4

2. Project Background 4

3. Preface 5

4. Short System Description 5

Deliverables 6

System Diagram 7

5. Requirements 8

5.1 Functional requirements 8

5.2 Non-Functional Requirements 9

5.2.1 Software Requirements 9

5.2.2 Design Decisions 9

5.3 Stretch requirements 10

6. Actors and User Stories 11

6.1 Actors 11

6.1.2 Actor Characteristics 11

Data scientist 11

Data engineers 11

6.2 User Stories 12

7. System Restriction 15

7.1 Security 15

End-to-end encryption 15

7.2 Data Storage/Cache 15

7.3 Open Source 15

7.4 Accessible API 15

9. Additional Requirements for System Construction 16

9.1 Process Requirements 16

9.1.1 Continuous Integration/Continuous Delivery (CI/CD) 16

9.1.2 System Development Framework 17

9.2 Technical Requirements 17

9.2.1 System Packaging 17

10. Additional Requirements for Documentation 17

10.1 System Documentation 17

11. Dictionary 19

1. Presentation

Client

Arbeids- og velferdsdirektoratet
Sannergata 2
0557 Oslo, Norway

About the Client

The Norwegian Labour and Welfare Administration (NAV) is the backbone of the Norwegian welfare state, administering a third of the national budget and servicing almost 2.8 million people through a range of schemes such as unemployment benefit, work assessment allowance, sickness benefit, pension, child benefit and cash-for-care benefit.

In addition, NAV manages and retains stewardship of several important data sources containing information on its users and the services it provides. NAV IT is currently in the midst of a digital transformation, undergoing significant changes in team organization, work processes and harnessing new technologies.

Its use of data in the development of new and improved services is often contingent upon strict data privacy practices and its ability to innovate in a privacy-preserving manner.

Our client for this assignment is NAV IT Data og Innsikt. Data og Innsikt is a department within NAV IT. The department delivers the development of systems and operations of data storage, data processing, data access and analytics.

Person of Contact

Name	**Role**	**Email**
Gøran Berntsen	Tech Lead - Åpne Data	Goran.Berntsen@nav.no

Client Stakeholders

Robindra Prabhu	Data Scientist - AI Lab	Robindra.Prabhu@nav.no
Paul Bencze	Tech Lead - AI Lab	Paul.Bencze@nav.no

Supervisor from OsloMET

Name	**Role**	**Email**
Eva Hadler Vihovde	Associate Professor	evahadler.vihovde@oslomet.no

2. Project Background

AI Lab is a department in NAV IT Data og Innsikt, functioning as NAV ITs internal knowledge hub in machine learning and data science. One of the areas the team currently is handling is data anonymization. This area presents problems in both the legal and ethical domain, because of its close connection to personal data.

Data anonymization is a large field with many projects worldwide. There are well established models and algorithms for both anonymization and analytics of data. AI Lab is currently utilizing both internal and external tools for data anonymization. One such tool is ARX, which is widely regarded as top-class anonymization software. ARX is an open source (Apache License, Version 2.0) project distributed as a GUI application and as a Java JAR library. It is prominently used by large organizations to anonymize health and patient data.

ARX contains a large and powerful amount of functionality for data anonymization, but the interaction with said functionality is limited to either interaction with the GUI application or programmatically with the Java API provided by the JAR. Neither option is well suited to modern data science applications.

Data science today is typically conducted within programming languages like Python and R. The data scientist develops scripts, notebooks and larger programs for extracting and analysing data. Early stage tasks typically include data cleaning and data transformation. Anonymisation may be viewed as such a data transformation task, but ARX currently does not integrate seamlessly into the typical Python/R-based data science workflow.

Moreover, while ARX provides functionality and flexibility for a skilled user, the front-end arguably requires knowledge of technical concepts and methods in anonymisation above and beyond that of the typical developer and which requires a non-trivial investment of time to learn and understand.

The AI-lab has therefore requested the group to:

- Provide access to ARX functionality from modern data science toolsets and workflows
- Provide an extendable framework for making state-of-the-art anonymisation methods accessible to a wider audience in NAV IT by lowering the barriers to use.

3. Preface

Sommerville (2010). “The software requirements document [...] is an official statement of what the system developers should implement.” “[...] the requirements document has to be a compromise between communicating the requirements to customers, defining the requirements in precise detail for software developers and testers, and including information about possible system evolution.” *Software Engineering* (91-92)

This is a technical document meant for the product stakeholders, with the purpose of providing clarification and guidance to the project. It contains both the technical and non-technical requirements and they are written in close cooperation with our client.

The terms *data* and *dataset* are used continuously throughout the documentation. Unless anything else is specified, the term refers to tabular data/datasets. The problem space is data anonymization, as such, when talking about data/datasets we are generally referring to population data.

The team employs an Agile work process. The product specification also serves as our basis for the backlog of user stories to be prioritised and implemented during the project lifetime.

4. Short System Description

The system will provide access to anonymization tools for data scientists at NAV IT. A data scientist should be able to anonymize tabular dataset based on user-specific configurations. Configurability includes privacy models, column attribute types and transformation models that determine how much data will be lost in the resulting anonymized dataset.

A common use case would be in a workflow where the data scientist is manipulating a dataset, and requires dynamic analysis of the data’s anonymity metrics. Another use case could involve integrating the system in a data pipeline to provide data analytics and anonymization capabilities.

Deliverables

- **Python Package - working title: PyAaaS**

  Python package wrapper providing abstracted access to the backend service.

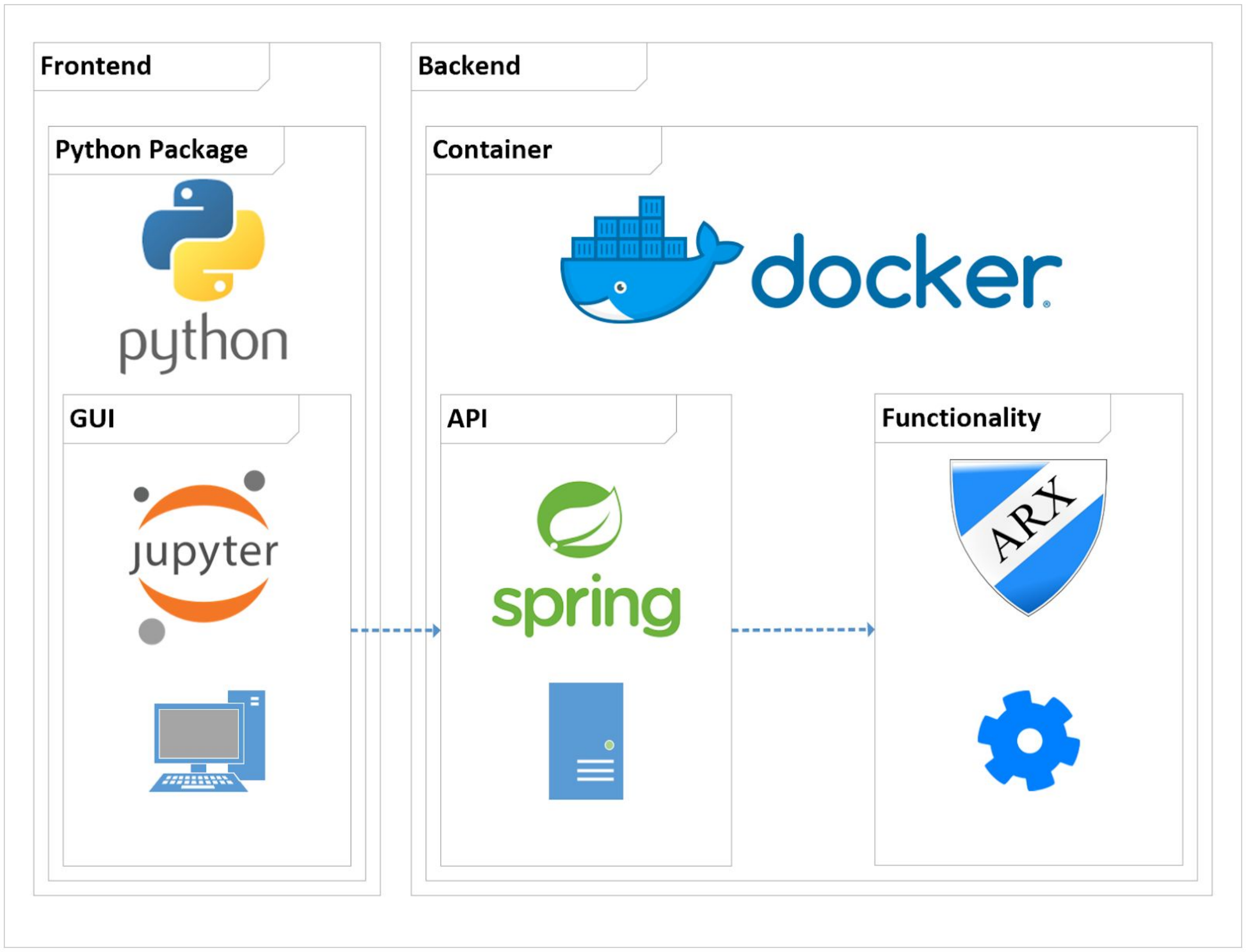
- **Web Service - working title: AaaS**

  Java Spring web service.

System Diagram

Preliminary draft of the system to be implemented

Jupyter notebook is a common user interface among data scientists, and will be a important platform for the system to support. In a Jupyter Notebook a data scientist that wishes to anonymize or analyze a dataset will import a Python package which wraps and abstracts the backend service. The backend service utilizes the ARX library and Spring framework to deliver the anonymization and analytics functionality as a web service. The service is packaged as a Docker container.



5. Requirements

The collection of system requirements defined in collaboration with the client(NAV IT - AI lab)

5.1 Functional requirements

- The system will provide the ability to complete data anonymization with the provided user configurations on tabular datasets.
- The system will provide the ability to analyze re-identification risks on tabular datasets.
- The system will provide the ability to configure the Privacy Models to use in the anonymization.
- The system will provide the ability to configure data Attribute Type to use in the anonymization.
- The system will provide the ability to configure the Transformation Models to use in the anonymization.
- The system will provide the ability to produce a visual presentation of data anonymity metrics.
- The system will provide the ability to compare data from before and after data anonymization.
- The system will be able to consume data in a variety of formats including (pandas.DataFrame, path to csv file, url to data resource, csv string, JSON).
- The system will be able to deliver the anonymized dataset in a variety of formats including (pandas.DataFrame, csv file, JSON).
- The system will be able to deliver metrics about the anonymization in a variety of formats including (pandas.DataFrame, csv file, JSON, Data Package).
- The system will provide the ability to produce data package metadata regarding the anonymization process that has been completed on the dataset and the relevant metrics.

5.2 Non-Functional Requirements

5.2.1 Software Requirements

- The client has requested that the team uses the ARX anonymizer library to implement anonymization functionality.
- The client has required that resulting anonymization process has to be more efficient than the previous and reduce the hours that are spent doing this manually.
- The client has required that the project is published as an open source project with an MIT licence.
- The client has required that the system backend will be packaged as a docker image so the service can be deployed to the Nais/Kubernetes platforms.
- The client has required that the team develop a Python package to "wrap" the web service, it will provide easy integration and interaction between the web service and data scientist tools and processes.
- The client has requested that the Python package has to be designed for use in a Jupyter notebook.
- The client has required that the system will utilize end to end encryption for data in transit, to and from the web service backend.

5.2.2 Design Decisions

Design decisions made by the team in collaboration with the customer to achieve the stated goal of the system.

- English will be the main language used for both the documentation and programming to make it easier for the team to deliver on the open source requirement from the client.
- The team has decided to utilize Java as its runtime environment for the backend service. The ARX library that the client has requested to be used is packaged as a Java JAR file. Using Java was a logical choice.
- The team has decided to use a service architecture to decouple the different logical components of the project. A service architecture will also deliver on the clients wish to be able to scale the system dynamically according to use.

- The team has decided to utilize Spring as its backend framework to deliver a web service in accordance with the service architecture. Spring is the defacto standard for Java web applications and has great libraries for development of secure, scalable web applications.

5.3 Stretch requirements

Stretch requirements are wishes from the customer that the development team will try to achieve if there is time left after the main requirements.

- The system will be able to **auto generate a hierarchy** level based on the column attribute type.
- Provide specific Transformation Model hierarchies for **NAV specific use cases** (eg. Norwegian geographical hierarchies, Norwegian zip code hierarchies).
- Provide an alternative **web frontend** that provides a lower barrier to entry, and a more user friendly interaction.
- Graphana dashboard for surveillance of the anonymization service.

6. Actors and User Stories

Description of Actors, their characteristics and user stories that defines their interest.

6.1 Actors

An actor is an entity who interacts with the system from the outside. Primary actors are those who require assistance from the system. While secondary actors are those who the system needs assistance from.

Primary Actors	**Description**
Data Scientist	A data scientist working at NAV.

Secondary Actors	**Description**
Data Engineer	A data engineer working at NAV.

6.1.2 Actor Characteristics

Data scientist

Data scientist are super users with good programming and statistical knowledge. Data scientists are the primary user of the system.

Data engineers

Data engineers are system engineers specialised in data driven applications. They support the data scientists by building and deploying data pipelines and other data infrastructure.

6.2 User Stories

User stories are one of the primary development artifacts for Scrum project teams. A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it.

Actor	Story	Priority
Data Scientist	As a data-scientist, I would like to easily **visualize the anonymization metrics** for anonymized datasets Reasoning: Visualization is a powerful tool to get an understanding of complex data.	High
Data Scientist	As a data-scientist, I would like to **analyze the anonymization metrics (re-identification risks)** of my dataset Reasoning: Getting metrics of the anonymization level of a dataset is necessary to judge how safe the dataset is to use in production, and/or if the dataset should be anonymized further.	High
Data Scientist	As a data scientist, I would like to have a single source where to lookup the **documentation for the PyAaaS package** (AaaS Python wrapper package). Reasoning: To facilitate efficient use of the Python package it is critical to make available up-to-date documentation of both the package API and tutorials for common use cases.	Medium
Data Scientist	As a data scientist, I would like to be able to **configure the Privacy Models** to be used when anonymizing my dataset	High
Data Scientist	As a data scientist, I would like to be able to **configure the Transformation Models** to be used when anonymizing my dataset.	High
Data Scientist	As a data scientist, I would like to be able to use **K-Anonymization as a Privacy Model** for my dataset.	High
Data Scientist	As a data scientist, I would like to be able to **set a global transformation scheme for all record**s in a column/field.	High

Data Scientist	As a data scientist, I would like to be able to **set a local transformation scheme for a column/field**. Meaning a unique transformation scheme for each individual row or subset of rows in a column/field.	Low
Data Scientist	As a data scientist, I would like to be able to use a **Value Generalization hierarchy as a Transformation Model** for a column/field.	High
Data Scientist	As a data scientist, I would like to be able to use **random sampling as a Transformation Model** for a column/field	Medium
Data Scientist	As a data scientist, I would like to be able to use **attribute suppression as a Transformation Model** for a column/field	Medium
Data Scientist	As a data scientist, I would like to use **microaggregation as a Transformation Model**.	Medium
Data Scientist	As a data scientist, I would like to use **Top- and bottom-coding as a Transformation Model.**	Medium
Data Scientist	As a data scientist, I would like to use **Categorization as Transform Model for a column/field.**	Medium
Data Scientist	As a data scientist, I would like to **identify rows affected by lowest risk** in a dataset.	Low
Data Scientist	As a data scientist, I would like to determine the **Lowest prosecutor re-identification risk.**	Low
Data Scientist	As a data scientist, I would like to determine **highest prosecutor re-identification risk.**	High
Data Scientist	As a data scientist, I would like to **identify rows affected by highest risk.**	High
Data Scientist	As a data scientist, I would like to **determine average prosecutor re-identification risk.**	High
Data Scientist	As a data scientist, I would like to **determine fraction of unique records.**	High
Data Scientist	As a data scientist, I would like to be able to **use K-map as a Privacy Model** for my dataset.	Low

Data Scientist	As a data scientist, I would like to be able to use **Average risk as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **Population uniqueness as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **Sample uniqueness as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **δ-Disclosure privacy as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **β-Likeness privacy as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **δ-Presence privacy as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **Profitability privacy as a Privacy Model** for my dataset.	Low
Data Scientist	As a data scientist, I would like to be able to use **Differential privacy as a Privacy Model** for my dataset.	Medium
Data Scientist	As a data scientist, I would like to be able to use **ℓ-Diversity as a Privacy Model** for my dataset	High
Data Scientist	As a data scientist, I would like to be able to **set presets for anonymization** e.g loss percentage, min/max risk of prosecution	Low
Data Scientist	As a data scientist, I would like to be able to **verify whether an anonymized dataset** has been anonymized from an original dataset	Low

Actor	**Story**	**Priority**
Data Engineer	As a data engineer supporting the AaaS service in NAV, I would like to be able to **deploy the service to a docker container** environment.	high
Data Engineer	As a data engineer supporting the AaaS service in NAV, I would like **continuous information about the build status of the AaaS web service** source code	medium

Data Engineer	As a data engineer supporting the AaaS service in NAV, I would like a **single source/location for documentation,** for setup and deployment of the AaaS service.	medium
Data Engineer	As a data engineer supporting the AaaS service in NAV, I would like a single **source/location for the AaaS projects JavaDoc**	low
Data Engineer	As a data engineer supporting the AaaS service in NAV,I would like to have **logging available from the AaaS service.**	high

7. System Restriction

This section explains the system and client defined restriction. In this section the team defines the limitations of the system to be developed.

7.1 Security

End-to-end encryption

The client has requested that the system use end-to-end encryption between the backend service and consumers of the service (Python package, Third-party applications). The team has decided to use TLS/SSL provided by the Spring framework.

7.2 Data Storage/Cache

The developed system cannot store or implement caching due to the sensitive nature of the data used.

7.3 Open Source

In accordance with the clients request the project with all source code and documentation will be released under an open source licence. Currently the team working under the MIT Licence.

7.4 Accessible API

Our API must follow RESTful guidelines and strive to provide endpoints that allow for seamless interaction with the ARX library.

9. Additional Requirements for System Construction

Additional non-functional requirements defined by the development team in cooperation with the client. These requirements are meant to improve the quality and maintainability of the end product.

9.1 Process Requirements

9.1.1 Continuous Integration/Continuous Delivery (CI/CD)

CI platform

The system uses Travis CI and Code Climate as Continuous Integration tools. Travis ensures that the codes are tested, while code climate checks the code quality and test coverage before being pushed to the repository. Along with Travis and Code Climate, the team uses GitHub with merge rules to ensure that the master build stays stable.

Each new iteration of the master build is packaged into a jar file, which would be packaged again into a Docker image. This way we will have different stable builds that can be deployed easily as a Docker container.

Version Control System (VCS)

Travis CI along with GitHub Merge rules maintains the stability of each version before a release.

Github Merge rules does not allow directly pushing to the master build, along with not allowing to push to the repository unless all test class passes. Each time there is a new implementation a new branch needs to be made. This new branch is then tested before being pushed to the repository, which is then finally merged to the master build.

Travis CI instantiates a docker container that runs the build being pushed along with all the test classes. If a build passes Travis will then allow the build to pushed.

Static Code Analysis

Code climate is used to ensure the maintainability and test coverage of the codes written. Travis generates a test report using Jacoco, which is then forwarded to Code

Climate by using a unique ID. Code climate reads through the report and generates a grade for test coverage of our system. Code climate is directly link to the systems GitHub repository, granting access to check the quality of the codes written. Based on the quality of the code a grade will be generated for the maintainability of our system.

9.1.2 System Development Framework

The team is developing the system under the SCRUM framework.

9.2 Technical Requirements

9.2.1 System Packaging

Backend Service

- Docker Image

Python Wrapper Package

- Python package wheel and source distribution

10. Additional Requirements for Documentation

Additional non-functional requirements defined by the team in cooperation with the client. These requirements are meant to improve the quality of the documentation.

10.1 System Documentation

The system backend service and python package will be delivered with documentation for the corresponding to the intended usage.

Backend Service Documentation

- Setup and deployment tutorial
- System JavaDoc

Python Wrapper Package Documentation

- Installation
- Common usage tutorial
- Examples (notebooks)
- API docs

11. Dictionary

Attribute Type - the disclosure risks from which a dataset is to be protected.

ARX - is a comprehensive open source software for anonymizing sensitive personal data.

Code Climate - engineering process insights and automated code review for GitHub and GitHub Enterprise help you ship better software, faster.

Continuous Integration - the practice of merging all developer working copies to a shared mainline several times a day.

Data Package - Data Package is a simple container format used to describe and package a collection of data. link: https://frictionlessdata.io/data-packages/

Data pipeline - a data pipeline is a set of actions that extract data (or directly analytics and visualization) from various sources.

Docker - the fastest growing cloud-enabling technology and driving a new era of computing and application architecture with their lightweight approach to bundle applications and dependencies into isolated, yet highly portable application packages.

GitHub - a git repository hosting platform. Commonly used for open-source software.

Jacoco - is a free Java code coverage library distributed under the Eclipse Public License.

Java - is a high-power, stable and highly trusted programing language. Commonly used in backend application with a requirement for stability.

Javascript - an object-oriented computer programming language commonly used to create interactive effects within web browsers.

JSON - is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types.

Jupyter Notebook - Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

Kubernetes - is an open-source container orchestration system for automating application deployment, scaling, and management.

MIT Licence - The MIT License is a permissive free software license originating at the Massachusetts Institute of Technology (MIT). link: https://opensource.org/licenses/MIT

Nais - is an application platform built to increase development speed by providing our developers at NAV with the best possible tools to develop and run their applications.

Pandas - is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Pandas Dataframe - a one-dimensional labeled array capable of holding any data type with axis labels or index.

Privacy Model - Privacy models are specific algorithms applied to a dataset to protect the dataset from an identification risk vector.

Product Stakeholders - NAV IT, OsloMET, the development team

Python - Python is a powerful, high-level language. Used in everything from web apps to data-science/machine learning.

Spring - is an open-source, high-power, Java application framework for business applications.

Scrum - is an agile framework for managing knowledge work, with an emphasis on software development. It is designed for teams of three to nine members, who break their work into actions that can be completed within timeboxed iterations, called "sprints", no longer than one month and most commonly two weeks, then track progress and re-plan in 15-minute stand-up meetings, called daily scrums.

Tabular dataset - a data consisting of or presented in columns or tables.

Transformation Model - Transformation Models are strategies that describes how a specific column in the dataset should lose data as the anonymization tries to achieve the required anonymization level specified by the Privacy Model.

Travis CI - is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub.