Department of Computer Science
University of Pretoria

Software Requirements Specification for GND

April Four

June 2024

Version	Version History	Author
1.0	Initial release	AprilFour

Table 1: Version History

Table of Contents

1 Introduction

1.1 Document Overview

This document serves to outline the Software Requirements Specifications of the GDPR Non-Compliance Detector (GND). It is a blueprint that will detail the what GDPR is, the purpose of the project, a guide about the development process as well as other details such as functional and quality requirements, use cases, user stories and the technological stack that will be used to develop the GND.

1.2 Background

The General Data Protection Regulation (GDPR) was a data protection law that was passed in 2018 by the European Union in attempt to give citizens greater control control over their personal data and hold corporations accountable for user data mismanagement. The regulation states that users must give their explicit consent before their data may be processed, they may choose to withdraw this consent at any time and use data must be securely stored with any data breaches to be reported immediately. The following are the data protection principles laid out as per GDPR.eu:

- Lawfulness, fairness, and transparency — Processing must be lawful, fair, and transparent to the data subject.
- Purpose limitation — You must process data for the legitimate purposes specified explicitly to the data subject when you collected it.
- Data minimization — You should collect and process only as much data as absolutely necessary for the purposes specified.
- Accuracy — You must keep personal data accurate and up to date.
- Storage limitation — You may only store personally identifying data for as long as necessary for the specified purpose.
- Integrity and confidentiality — Processing must be done in such a way as to ensure appropriate security, integrity, and confidentiality.
- Accountability — The data controller is responsible for being able to demonstrate GDPR compliance with all these principles.

1.3 Objectives

The objective of AprilFour in developing the GND is to create system that may be used to detect data within documents that may be considered confidential by GDPR standards and alert the user. The client, Africa Talent by Deloitte, requires an application that will run as background process and regularly scan incoming documents for GDPR violations as well as allow users to upload documents themselves. The application should be with a single system architecture, be lightweight and simple to use and navigate by users. Security is our most important priority with special care to be taken to ensure that documents may not be leaked, intercepted and mishandled.

2 User Characteristics

The following details the types of users who will utilise the GND:

1. Deloitte employees located within the EU who will submit and transfer documents.
2. Deloitte employees who will monitor the number of violations within the company's network.

Deloitte employees located within the EU need to ensure that every document they intend on sending outside of the Deloitte network needs to be free of any potential GDPR violations. Employees should be able to upload documents and receive a rating of how compliant the document is, in order for them to minimize the chances of transmitting non-compliant data.

Deloitte employees who will monitor the number of violations within the company's network will need a system which logs the current number of potentially dangerous documents. These users will be given an overview of the network's content with regard to non-compliant documents. Accessing this kind of information warrants authentication and user logging.

3 User Stories

Deloitte employees located within the EU who will submit and transfer documents:

1. I can access the system through a Windows executable.
2. I can connect my email attachment path to the system.
3. I can connect my Microsoft Teams attachment path to the system.
4. I can remove or update the email attachment path.
5. I can remove or update the Microsoft Teams attachment path.
6. I can terminate monitoring of the email attachment path.
7. I can terminate monitoring of the Microsoft Teams attachment path.
8. I can select a file to upload to the system, either Microsoft Word, Excel, or a PDF
9. I can submit the file I have chosen to be processed by the system.
10. I view the output of the sequence carried out by the system.
    - I can view the report.
    - I can view a visualization of the potential violations.
    - I can view a document which highlights the potential violations.
11. I can forward the generated report.
12. I can access the 'Help' or FAQ section.

Deloitte employees who will monitor the number of violations within the company's network:

1. I can access the logs through dedicated website or program logs.
2. I can view the number of violations detected within a specified time frame.
3. I can specify the time frame.
4. I can view the date a single violation was detected.
5. I can view the report of a single violation flag.
6. I can view the visualization of a single violation flag.
7. I can delete the record of the violation.

4 Functional Requirements

4.1 Manual Interface

The GND will allow users to manually upload files to analyse for possible GDPR violations.

1. Allow users to upload a file of any type
2. Allow users to remove the file
3. Allow the user to submit the file for analysis
4. Allow the user to view and download the report generated

4.2 Email Monitor

The GND will automatically monitor the attachments folder of a user's email for possible GDPR violations

1. Specify path of email attachments
2. Initiate path monitoring
3. Submit file to Document Parser
4. Terminate monitoring of file path

4.3 Document Parser

The Document Parser within the GND is where documents that are submitted for scanning are converted, sanitised and secured before processing.

1. Accept any document type
   - 1.1. The document parser should be able to verify any document type (PDF, Word, Excel)
   - 1.2. Route the document to the appropriate processor
2. Handle document
   - 2.1. Extract Text
   - 2.2. Store output
   - 2.3. Parse Output
3. Secure and sanitise text
   - 3.1. Sanitize the text to remove any potential threats
   - 3.2. Parse Output

4. Preprocess text for analysis
5. Create transaction on database
6. Submit text to GDPR Detection Engine

4.4 GDPR Violation Detection Engine

The GDPR Violation Detection Engine is where the text is processed and analysed for possible GDPR violations

1. Send text through text classification model
    - Package and submit output
2. Send text through metadata extraction mode
    - Package and submit output
3. Send text through language detection model
    - Package and submit output
4. Capture output from models
    - Package and submit output
5. Perform quantitative analysis on data received from models
    - Package and submit output
6. Generate report based on quantitative analysis
    - Package and submit output
7. Generate visualisations based on quantitative analysis
8. Generate marked up text for display on the User Interface
9. Package the report, quantitative analysis, and visualisations to be stored on the database as output of the transaction
10. Encrypt data in transit
11. Submit output

4.5 Data Storage Module

This where reports generated by the engine will be sent to be analysed, stored and data visualisations generated.

1. Connect to Azure Server
2. Schedule data crawling
3. Discover personal data
4. Extract and store data automatically
5. Data Storage Management
    5.1. Organize and manage stored data
    5.2. Optimize storage for performance
    5.3. Handle large volumes of data
    5.4. Backup and recovery mechanisms
6. Data Protection and Security
    6.1. Encrypt data at rest
    6.2. Encrypt data in transit
    6.3. Log access and modification events
    6.4. Maintain audit logs securely
    6.5. Provide log access for audits

5 Subsystems

1. Manual Interface
2. Email Monitor
3. Document Parser
   - 3.1. File Validator
   - 3.2. Text Extractor
   - 3.3. Sanitizer
   - 3.4. Analysis Preparation Module
   - 3.5. Storage and Submission
4. GDPR Violation Detection Engine
   - 4.1. Language Detector
   - 4.2. Metadata Extractor
   - 4.3. Text Classifier
   - 4.4. Model Data Interpreter
   - 4.5. Data Presentation Component
5. Data Storage Module

6 Use Case Diagrams

6.1 Manual Interface

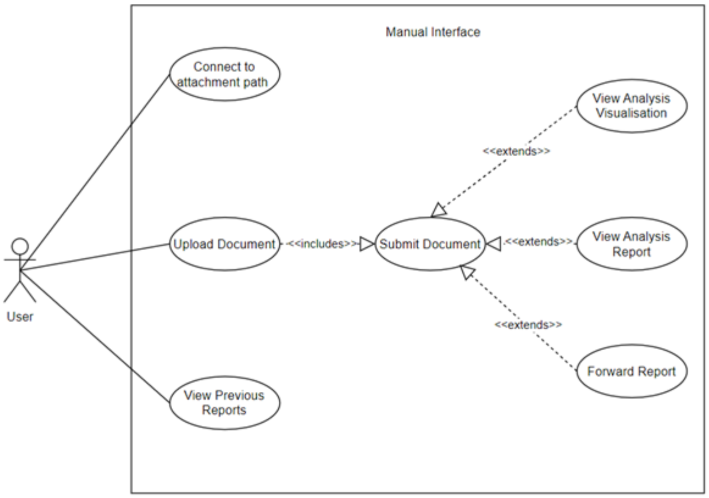

Figure 1: Manual Interface Diagram

6.2 Email Monitor

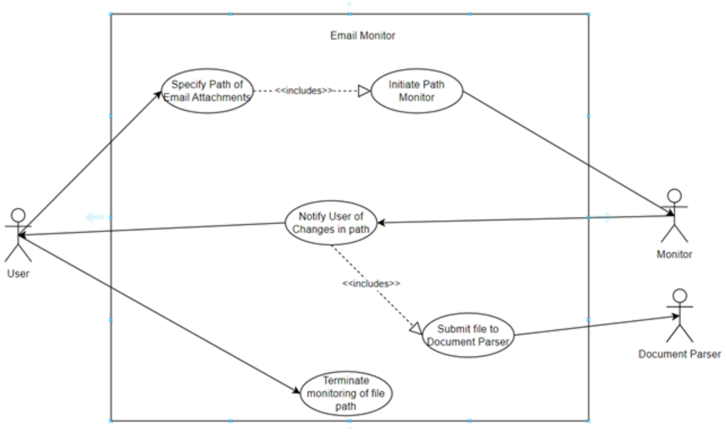
Email Monitor
Specify Path of Email Attachments
<<includes>>
Initiate Path Monitor
Notify User of Changes in path
Monitor
User
<<includes>>
Submit file to Document Parser
Document Parser
Terminate monitoring of file path

Figure 2: Email Monitor Diagram

6.3 Document Parser

6.3.1 Validator

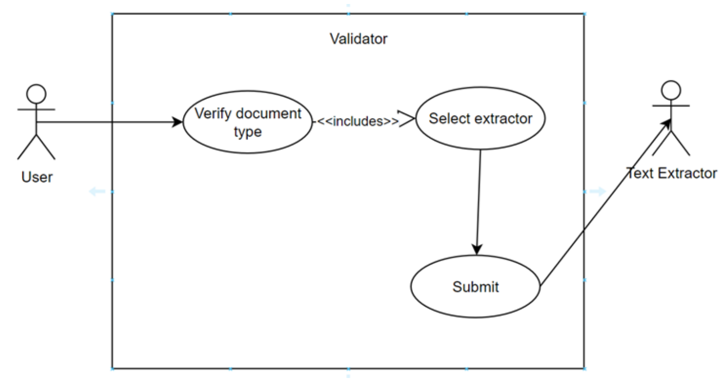

Figure 3: Validator Diagram

6.3.2 Text Extractor

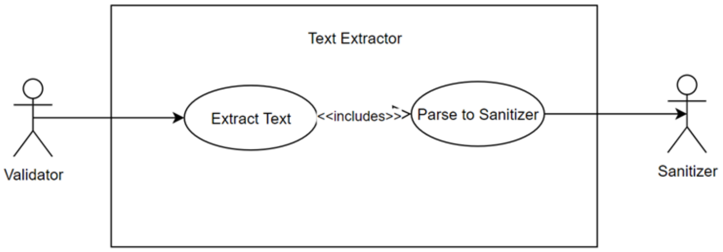

Figure 4: Text Extractor Diagram

6.3.3 Sanitizer

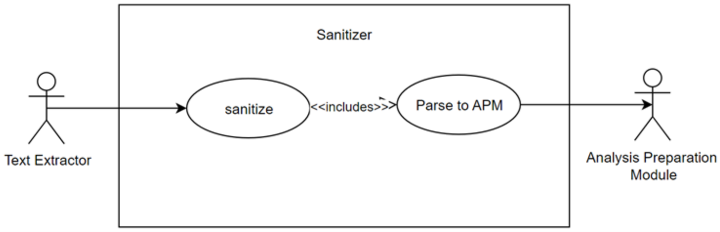

Figure 5: Sanitizer Diagram

6.3.4 Storage and Submission

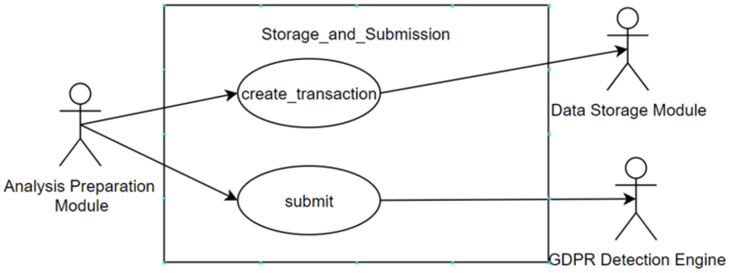

Figure 6: Storage and Submission Diagram

6.4 GDPR Violation Detection Engine

6.4.1 Language Detector

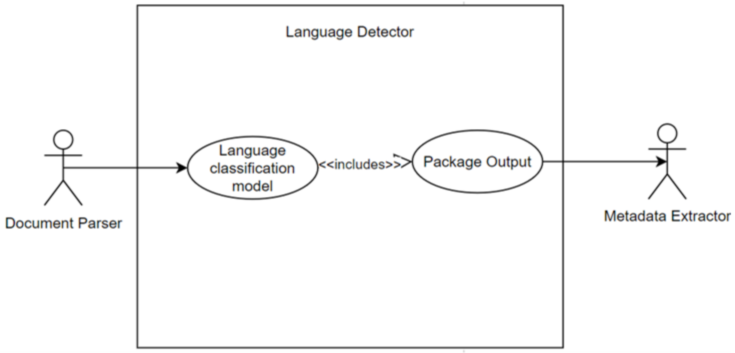

Figure 7: Language Detector Diagram

6.4.2 Metadata Extractor

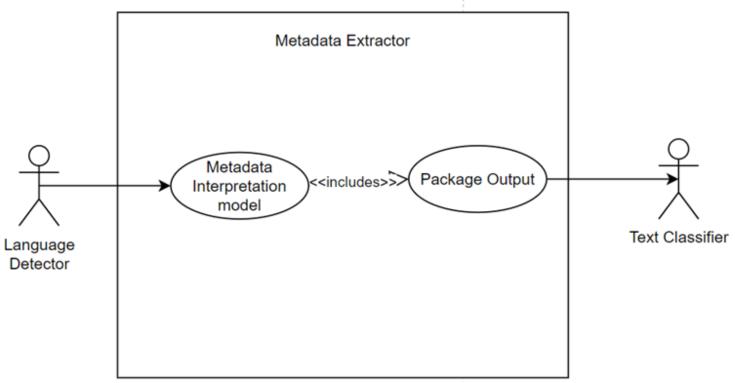

Figure 8: Metadata Extractor Diagram

6.4.3 Text Classifier

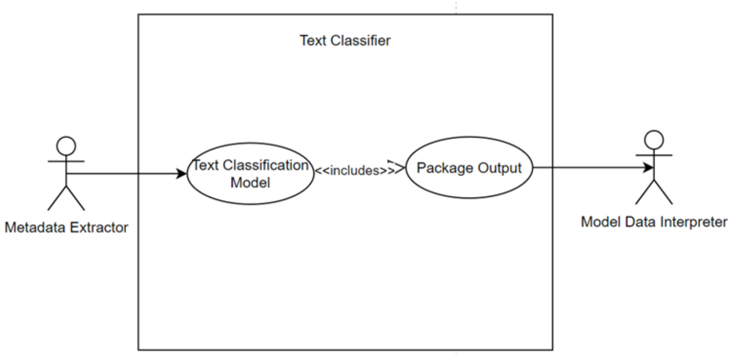

Figure 9: Text Classifier Diagram

6.4.4 Model Data Interpreter

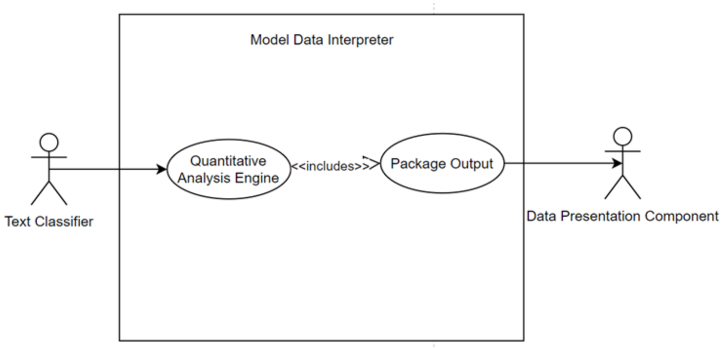

Figure 10: Model Data Interpreter Diagram

6.4.5 Data Presentation

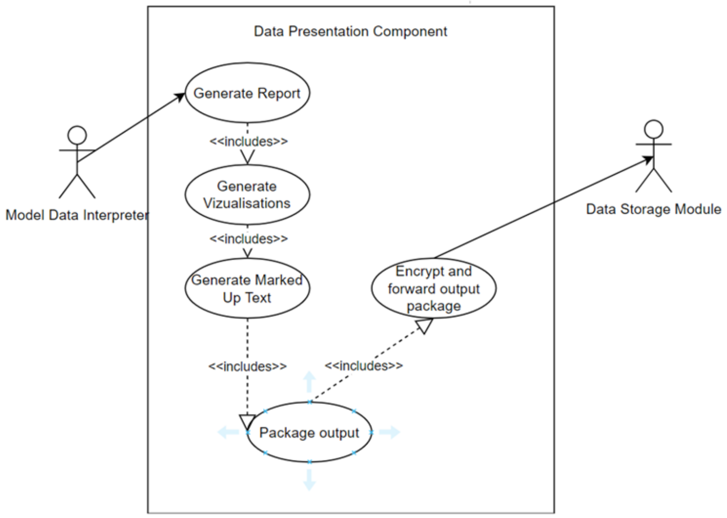

Figure 11: Data Presentation Diagram

6.5 Data Storage Module

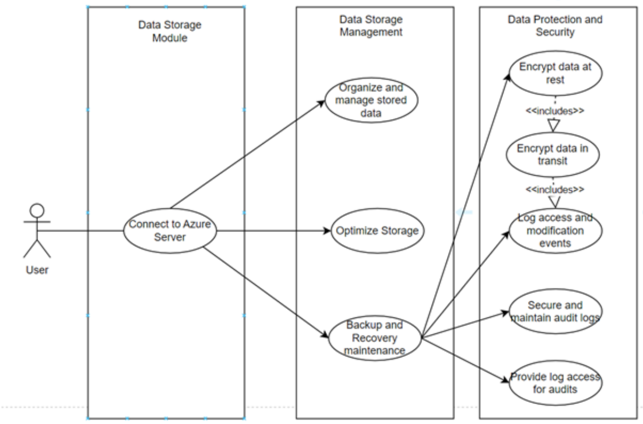
Data Storage Module
Data Storage Management
Data Protection and Security
User
Connect to Azure Server
Organize and manage stored data
Optimize Storage
Backup and Recovery maintenance
Encrypt data at rest
<<includes>>
Encrypt data in transit
<<includes>>
Log access and modification events
Secure and maintain audit logs
Provide log access for audits

Figure 12: Data Storage Diagram

7 Service Contract

7.1 Deliverable

As per the project proposal and further contact with the project owner, Africa Talent by Deloitte, the team, AprilFour, will be required to develop and deliver a Windows executable that is able to run on all Deloitte machines within the project time frame. The application, GND, should be abe to detect possible GDPR violations and alert users of these violations.

7.2 Project Management

The team will follow the Agile methodology to develop the system. The team is familiar with the workings of it and are comfortable working within it. We will make use of Scrum, daily team stand up meetings, sprint planning and weekly reviews. Feature Driven Development will enable us to keep on top of the project while ensuring proper forward planning. Bi-weekly meetings with the client will make sure that we are on track to deliver what is required and assist in forward planning.

7.3 Timeline

The project is broken down in 4 sprints, with a demo to the University of Pretoria COS 301 team at the end of each sprint. This is to detail the current status of the project, assist in any issues and help grade the team's performance.

7.4 User and Data Privacy

Due to the nature of the project, data security and management is our number one priority. We are bound by not only the Protection of Personal Information Act (POPI) of South Africa, but the General Data Protection Regulation (GDPR) of the EU as well. Any data processed must be handled within the laws of these acts.

8 GND Class Diagram

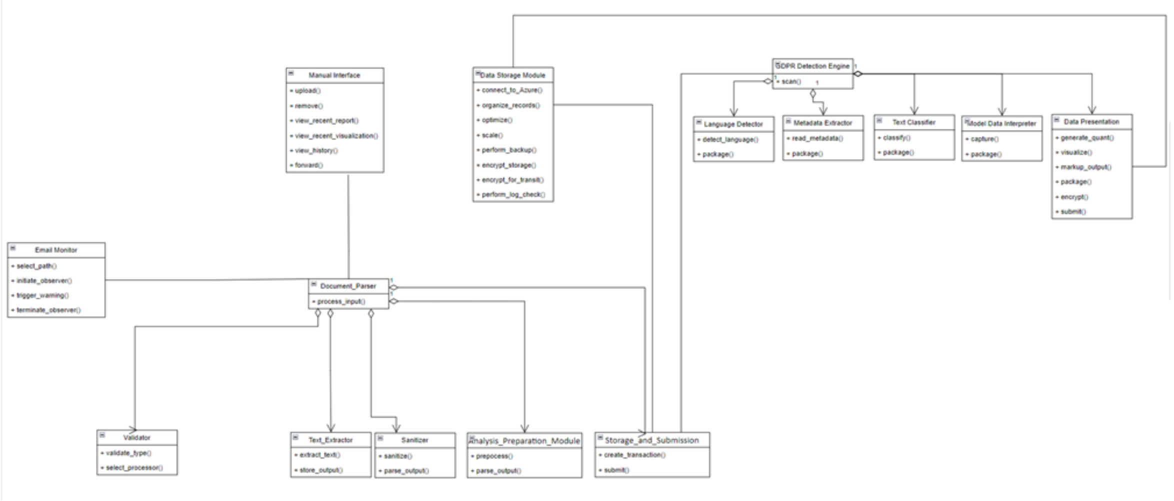

Figure 13: Class Diagram

9 Quality Requirements

The following quality requirements are those deemed most important in the context of the purpose and function of the GDPR Non-Compliance Detector (GND). Due to the nature of the system and agile methodology employed, more quality requirements may be added at a later stage.

9.1 Security

Due to sensitive data accessed and processed, security is the most important quality requirement for the GND application. Since the system will have to access and process sensitive data it is important that it handles any data it uses securely. Documents should be sanitised before processing to remove any identifying features such as metadata, geolocation data and data embedded within. After processing, the document should not be stored and immediately deleted. Any information collected during processing that may be needed for training purposes should also be sanitised and stored anonymously.

9.2 Compliance

Due to the nature of the system and its purpose of detecting potential GDPR violations, it is imperative that it comply with the laws and regulations laid out by the GDPR act. Any data that the system needs, should be sanitised, anonymised, and promptly deleted after processing. Failure to adhere to GDPR regulations may lead to legal action taken against the owners of the product.

9.3 Performance

The client (Africa Talent by Deloitte) requires that the application be able to run in the background and be able to access any documents received via email. The system will have to access the user's attachments folder and automatically process incoming documents for possible GDPR violations. The system should be lightweight and consume as little resources as possible. Due to the need to regularly access a user's inbox and inform the user of possible GDPR violations in a document they may have just received, the system must have fast response times, be able to handle multiple documents at the same time, different document types and process large files as well.

9.4 Reliability

Due to the system needing to run in the background and monitor a user's inbox or be able to process documents uploaded manually, it is important the system be designed and implemented in a robust manner. A failure of the system may result in documents not being scanned for potential GDPR violations could result in legal and financial issues.

9.5 Usability

The primary users of the software will be people that may not have technical knowledge of the system and how it works. The program should therefore be easy to understand and navigate ensuring that the intended users can utilise the application with minimal knowledge of its implementation and any additional training.

9.6 Scalability

The application should have the capability to process multiple documents at the same time. This could be via manual file inputs or retrieving multiple email attachments. It should also be able to accommodate new filters that will be used to detect violations in future. Therefore, the system should be designed with these factors mind and multiple file processing should not affect performance of the app.

10 Architectural Patterns

The following describes the architectural patterns that will be used in the design of the GND to achieve the required functionality of the system and how they can be used to enforce the quality requirements previously described.

10.1 Monolithic Pattern

As specified by the client, we are required to make use of a single system architecture. The Monolithic architecture aligns with this constraint, ensuring that all subsystems are encapsulated within one major system. The use of Monolithic architecture also aids in meeting system requirements, such as tight coupling, which can improve efficiency—a key concern—and the deployment of a single executable.

10.1.1 Quality Requirements Addressed

- **Security** - The single entity characteristic of the Monolithic pattern it is easy to enforce consistent security measures across all subsystems and the single point of entry makes it difficult for bad actors to access the system.
- **Performance** - Due to the tight coupling enforced by the Monolithic pattern, interaction between components is fast, efficient and overhead is reduced.

10.2 Model-View-Controller Pattern

MVC facilitates user interaction with the program, as the user will be able to specify input which will be processed by the system (Controller) through an interface (Model) and be able to view the resulting analysis (View) of the process carried out on the input. The use of MVC allows us to separate concerns and facilitates concurrent design, development, and testing of the frontend and backend features.

10.2.1 Quality Requirements Addressed

- **Performance** - By separating concerns of the different components, we can easily delegate the different types of interactions that the system will experience leading to modular, robust and efficient code.
- **Usability** - By separating concerns of the different components, we can easily delegate the different types of interactions that the system will experience leading to modular, robust and efficient code.
- **Reliability** - The different responsibilities of the different components allow easy error handling, debugging and different thresholds of fault tolerance.

10.3 Event Driven Pattern

The Event Driven pattern will be utilised primarily for the implementation of email inbox monitoring. Once a new email with an attachment is received, the program (which monitors the attachments folder of the email client) will access, process, and detail a report on whether the document is GDPR compliant and possible violations if it is not. The pattern can also be used to address certain events that may happen within the system itself.

10.3.1 Quality Requirements Addressed

- **Security** - When a document is uploaded manually or automatically accessed from an email inbox, event handlers can be used to sanitise, encrypt and anonymise the document before processing. Data can automatically be deleted after processing as well.
- **Reliability** - Events can be logged by the system to ensure that all possible violation checks are performed by the system and provide strong error handling assistance should a sub-system break down.

10.4 Pipe and Filter Pattern

The Pipe and Filter pattern is an ideal fit for the detection engine that will be used to detect possible GDPR violations within a document. The pattern allows for modular code where each data violation check is implemented as its own filter with a "pipe" and operates independent of each other. This promotes loose coupling within the engine and allows for more filters to be added later on without redesigning the entire system.

10.4.1 Quality Requirements Addressed

- **Scalability** - The pipe and filter pattern allows new checks/filters to be added to the detection engine without needing to change the design of the entire system.
- **Performance** - Each filter within the pipe can be run concurrently meaning that multiple documents can be processed at the same time, allowing near instant feedback for users on the status of documents.
- **Reliability** - Due to the loose coupling nature of the pipe and filter pattern debugging and error handling can be done independently and the failure of a single filter within the system will not affect the others.

11 Architectural Constraints

11.1 Client Defined Constraints

- The system must be file agnostic. This means that files of different types should be able to be handled and processed by the program without any input from the user.
- The system must be developed with a single system architecture.
- The deliverable must be a Windows executable and be integrated with Windows for automated scanning

11.2 Hardware and Operating System Constraints

- The program must be able to run on the Windows operating system. This is standard the operating system used across Deloitte computers.

12 Technologies Used

12.1 Frontend Development

The frontend of the system will be developed in C# with the .NET MAUI framework. C# is well integrated into the Microsoft environment and provides us with the freedom to design and implement UI components in accordance with Deloitte's preferred theme and styling, handle user interactions such as file input and viewing, and create a visually appealing and user-friendly interface for users to interact with.

12.2 Backend Development

The backend of the system will be developed with Python. Python is well equipped for data processing and analysis that will be required for processing large amounts of documents. It also provides a variety of libraries which can be used to develop reliable and performance driven modules such as a GDPR compliance detection engine, Document Parser and Data Storage Module. It provides solid cross-platform compatibility and a stable and consistent base for these modules as it can be used for integration of the system components.

12.3 Version Control

For version control, we will utilise GitHub. GitHub provides a highly efficient platform for managing and tracking changes to our codebase and provides a convenient way of handling each iteration of the system design. All group members are well versed in the use of GitHub and are also familiar with a GitHub development strategy which has proven effective for us and assists with the overall system development life cycle.

12.4 AI Frameworks

The integration of PyTorch, TensorFlow, NLTK, and Scikit-Learn will be the driving for behind the GDPR compliance engine. NLTK will provide the engine with the ability to scan text inputs to identify patterns, keywords, and contextual cues indicative of personal data, and collect details regarding the data subject. PyTorch and TensorFlow will be used to implement techniques needed to conduct likelihood analysis of GDPR violations through machine learning models trained on text samples which may or may not contain GDPR violations. By utilizing these frameworks, the engine can parse text inputs and generate compliance metrics, offering insights into the adherence to GDPR regulations. Additionally, Scikit-Learn enables metadata evaluation, allowing the engine to assess file paths, location tags, and email headers for patterns or information relevant to the data's origin. The use of these frameworks allows us to provide the engine with unique set of techniques for singling out GDPR violations and breaking the data down into understandable representations.

12.5 Data Visualisation

Data visualisation will be done using PowerBI. It is a powerful visualisation tool that will aid in report generation and group members have experience with them. It also integrates well with the Microsoft Azure ecosystem. As previously mentioned, we are aiming for consistency and stability within our system and the use of these technologies can help us do this.

12.6 Testing

Each testing tool serves a specific purpose in ensuring the quality and reliability of software. unittest is a built-in Python framework for performing unit tests. PyTest offers features such as automatic test discovery and parameterization. xUnit will be used for frontend UI testing as it works well with the .NET framework.

12.7 Team Organisation

Monday.com has been chosen because it allows for management of multiple resources in different streams, such as client projects, task management, resource management. This will separate our activities and simplify the process of assigning tasks along with providing and receiving updates. GitHub Project Boards will allow us to keep track of development progress. We are familiar with the technology and found it very useful when it comes to assigning tasks, indicating urgency, and accessing new developments in a timely manner.

12.8 Security

Pycryptodome can provide the system with encryption capabilities, such as confidentiality of data stored within the system through encryption. Operating system permissions provide an additional layer of defence, controlled by the device itself, allowing low level control over access to sensitive files and local resources, which can aid in limiting potential security breaches and ensuring data integrity within the system. Given that sensitive information will be handled within a single system architecture, the implementation of these technologies and measures is crucial for maintaining robust security.

12.9 CI-CD

We are considering the use of GitHub for our CI-CD activities. GitHub will serve as the central repository for version control and code sharing, and GitHub Actions can be used for automating the build, testing and integrating processes.