Department of Computer Science
University of Pretoria

Testing Specification for GND

April Four

September 2024

Version	**Version History**	**Author**
1.0	Initial release	AprilFour

Table 1: Version History

Table of Contents

1 Introduction **3**

2 Python Testing - Backend **4**

3 Angular and Electron Testing - Frontend **4**

4 Automated Testing **5**

5 Non-Functional Testing **6**
5.1 Security 6
5.2 Compliance 6
5.3 Usability 6

1 Introduction

This document serves to outline the Testing Specification of the GDPR Non-Compliance Detector (GND) application. It is a blueprint that will detail the technologies used by the AprilFour team to create and run tests, the reasons for running these tests and the outcomes of the testing specification.

2 Python Testing - Backend

Python was the language chosen by the team to write the business logic of the system, create and run the API which is required to interact the system and build the machine learning models on which GDPR is based on. `PyTest` and `unittest` were the testing frameworks chosen to test the robustness and fault tolerance of the application. `unittest` was used to write individual unit tests for functions and Python modules, while `PyTest` was the runner used to execute all backend integration tests. Both testing tools gave us the ability to mock data, API calls and results that assisted in developing robust code and preventing security leaks.

3 Angular and Electron Testing - Frontend

Angular was the Javascript framework chosen by the team to design and create the user interface that one can use to interact with application. `Karma` and `Jasmine` were the testing frameworks chosen to test the frontend user iterface. `Karma` was used to write tests for the all the components, services and Javascript files, while `Jasmine` was the runner used to execute all frontend integration tests. Similarly to `PyTest` and `unittest`, both testing tools gave us the ability to mock data, API calls and results that assisted in developing robust code and help to handle errors that may be thrown.

4 Automated Testing

In order to automate our tests when pushing to GitHub, a CI/CD pipeline was created that runs via GitHub Actions. All tests are run when pushing to GitHub as well whenever new pull requests. The team's policy was to ensure that all pull requests into the `develop` and `main` branches were free of errors and all tests passed. This includes deployments and linting. Linting was performed with SuperLinter and Flake8. This CI/CD pipeline helped ensure that any code pushed onto our repository was thoroughly tested and that the standard of code was high.

After each push, a report was detailing our testing is sent to CodeCov. This assisted us in pinpointing areas in our code that required more testing, error handling and similarly to our GitHub Action workflows helped keep the quality of code in our repository high.

5 Non-Functional Testing

The three primary non-functional requirements identified by the team is **Security**, Compliance and Usability.

5.1 Security

Due to the sensitive nature of the documents that may be passed through the system, files are encrypted during transit. Though the file is not sent over a public network, it is important that data still be secured. Before the file is sent to the system it is encrypted and the contents can only be seen when the file enters the GND system and is decrypted so that it can be parsed correctly.

5.2 Compliance

In a similar vein to security, compliance is one of our most important non-functional requirements. Any data that passes through the system must secured, not identifying or personal data may be saved and the document itself must be deleted from the system immediately after processing. This is achieved by removing the file from the system after it has been parsed and a report generated as well as clearing the apps storage before it closes.

5.3 Usability

The primary users of the GND application will be those with little technical experience in the field of machine learning, artificial intelligence and computer systems. Therefore the system must be easy to understand, use and not too overwhelming. Usability tests were performed on those on the corporate environment that were identified as the primary users. Feedback on the app was mostly positive, with emphasis on its simple design, ease of use and informative results.