Politecnico di Torino

Master's Degree in Cybersecurity
LM32 - Computer Engineering

Master's Degree Thesis

Towards Privacy-Aware Data Sharing: A Proactive Approach to Risk Analysis

Supervisors

Prof. Fulvio VALENZA
Prof. Daniele BRINGHENTI
Prof. Riccardo SISTO

Candidate

Valeria POLIZZI

Academic Year 2024–2025

Summary

Data sharing across digital services introduces significant privacy and security risks, especially when sensitive information can be correlated across multiple contexts. Traditional techniques of threat modeling approaches often remain manual, fragmented, and lack automated mechanisms to detect such risks in real time.
To address these limitations, this thesis applies the LINDDUN Privacy Threat Modeling framework as a methodological foundation for systematically identifying and categorizing privacy threats throughout the system architecture. Building on this systematic order model, this thesis proposes a framework for data sharing that preserves privacy and automated threat analysis, through the integration of Natural Language Processing techniques with risk evaluation based on rules.
The solution combines a BERT-based Named Entity Recognition (NER) model with a configurable risk engine and a Chrome browser Extension to perform real time analysis of user input in web applications. The platform detects sensitive entities such as names, email addresses, and financial identifiers automatically, evaluates their risk based on the sharing context (e.g., social media, e-commerce, private messaging), and identifies cross-site correlation threats that can lead to profiling, doxing, or identity theft.
The approach has been validated through case studies involving widely used services, including Google Calendar and Facebook. Experiments indicate that automated NLP-driven analysis, combined with context-aware modeling, provides an effective and scalable methodology to analyze and mitigate data sharing vulnerabilities, thereby supporting users in protecting personal information.

Acknowledgements

I would like to thank the Politecnico di Torino with all my heart for welcoming me and for giving me the chance to grow academically and personally. Politecnico has been an institution that has given me opportunities I never thought possible and allowed me to rediscover my value.

I would like to express my sincere gratitude to Professor Valenza for having faith in me and supporting my ideas. Professor Valenza has not only been an inspirational mentor but also a constant and priceless presence throughout the duration of these two years at the Politecnico.

I would also like to thank Professor Bringhenti for the guidance and availability provided throughout the process of preparing the thesis, which was essential during the entire writing process.

My thanks also extends to Professor Sisto, for the opportunity he gave me and for his trust in my capabilities.

My heartfelt thanks go to my parents, for giving me the opportunity to study as well as for pursue my dreams. You are the most precious part of my life, and I will be grateful to you forever.

To my sister, thanks for being the one who understands me and for always being there for me. You are my home and my biggest support. You are my source of inspiration.

This thesis is the result of perseverance and dedication. I hope it can be helpful and contribute, even in a small way, to future research.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The rapid expansion of digital ecosystems has changed the way personal data is produced, shared, and processed in everyday life. Social media, AI tools, and web services have become an important part of communication, work, and education.
While these systems enable great levels of connectivity and personalization, they also create new and subtle forms of privacy exposure. Users continuously share fragments of information such as names, addresses, identifiers, or behavioral traces, which, when combined, can reveal sensitive aspects of their identity. The widespread diffusion of artificial intelligence, together with the lack of transparency in data handling, has further intensified these risks, making privacy protection not only a technical challenge but a societal necessity.
In this context, privacy awareness must evolve from static policy enforcement toward dynamic and intelligent protection mechanisms capable of operating directly where data is generated: the user's browser.
Traditional privacy-preserving techniques, such as encryption or anonymization, often act after the data has already been transmitted, providing limited defense against contextual misuse. What is increasingly required is a proactive system that can detect and assess the sensitivity of information *before* it is shared, interpreting both the semantic meaning of the text and the context in which it appears.
This thesis introduces a privacy-aware monitoring system that integrates artificial intelligence with browser analysis in real time.
The proposed solution consists of two main components: a **Chrome extension** that operates as a lightweight monitoring agent on the device, and a **Flask-based backend** that performs deep linguistic analysis and contextual risk evaluation.
The system employs a hybrid detection pipeline that combines deterministic pattern matching, through regular expressions, with a transformer-based Named Entity Recognition (NER) model capable of identifying entities that depend on the context such as names, organizations, or locations. Each detected entity is then associated with a contextual risk level, determined by a hierarchical set of rules that consider both the *macro-context* (the category of website, e.g., social, professional, commercial) and the *micro-context* (the specific user action, such as posting publicly or sending a private message).
Beyond analysis in real time, the backend introduces a persistent, anonymized memory layer where detected entities are securely stored in a local database using

hashed identifiers. This enables the inference of **cross-site correlation risks**, where combinations of data shared across different domains might collectively reveal sensitive user patterns. For instance, the same name appearing on multiple platforms or a name combined with a birth date that could indicate identity theft. By correlating these findings, the system transforms from a reactive anonymization tool into an intelligent privacy assistant capable of identifying complex, multiple context threats.
Therefore it is important to design, implement, and validate a privacy oriented monitoring architecture that enhances users' awareness of the information they disclose online.
The work aims to demonstrate that privacy protection can be achieved not only by concealing data but by understanding its meaning and relevance within its specific digital environment. Through this approach, the system aspires to bridge the gap between artificial intelligence, human behavior, and privacy engineering, providing a concrete step toward responsible, sensitive to context data sharing in modern digital ecosystems.

1.1 Thesis description

This thesis is structured to guide the reader from conceptual motivation to technical implementation and evaluation of the system.
Chapter 1 represents the introduction with a general description of the problem and motivations behind the thesis. Enriched with a description of each chapter.
Chapter 2 introduces the theoretical foundation for understanding how personal data is shared online. It explores concepts such as data privacy, GDPR principles, pseudonymization techniques and privacy by design basics. This chapter also presents Threat Modeling methodologies, focusing on the LINDDUN framework used to address privacy risks rather than purely security oriented threats.
Chapter 3 focuses on the technological core of the analysis with the use of Natural Language Processing (NLP) for detecting sensitive entities in text. This chapter explains the functioning of Named Entity Recognition (NER) and details how transformer based models, in particular BERT, are used to improve detection accuracy.
Chapter 4 refines and formalizes the thesis goals, defining the research questions and measurable objectives that guide the development and evaluation of the proposed system.
Chapter 5 describes the methodological approach adopted in this thesis, including the motivation behind the chosen architecture, the incremental development process and the criteria used to assess effectiveness.
Chapter 6 provides a detailed description of the technical implementation and system design. It explains the technology used and describes both backend and frontend components. The chapter also explores the correlation risk inference mechanism and the design of the Chrome Extension's user interface.
Chapter 7 presents the experimental results, including detection accuracy, computational performance and a qualitative analysis of how the system responds to different types of inputs and contexts.
Chapter 8 summarizes the achieved objectives, discusses the broader impact of the proposed system on privacy protection, and suggests future directions.

The overall goal of this work is to demonstrate that privacy protection should evolve from a reactive to a proactive paradigm. Instead of limiting itself to hiding sensitive data, a new system should understand the *who*, *where*, *why* behind data sharing, allowing users to make conscious and informed decisions in an increasingly data driven society.

Chapter 2

Data Sharing and Threat Modeling

2.1 Introduction to Data Sharing

Data sharing can be defined as the process of making data resources available across different applications, organizations, or users. It depends not only on technical infrastructures, but also on organizational practices, governance mechanisms, and legal frameworks that regulate how information is exchanged.
Nowadays, data sharing serves as a foundation for data driven innovation and lies at the center of the experience in contexts like *social media*. Platforms like Facebook, Instagram or Google Calendar allow people to exchange personal information, including their photos, private messages, preferences with a network of friends, followers or even publicly. This form of sharing *democratizes access to information*, allowing the creation of communities, helping the dissemination of news, and supporting global collaboration.
At the same time, this process is not immune from risks, especially regarding privacy and transparency. All the huge amounts of data gathered from those social media can be used for malicious purposes like **hacking, scraping or data leakage**. An example was the scandal of Cambridge Analytica [2], where the data collected from Facebook was used without consent to influence political processes and behavioral models, highlighting the consequences of data manipulation. Moreover, the emerging phenomena, such as *sharenting* [3], which is the habit of parents to share content about their children online, can involuntarily pose a risk for children's privacy, exposing them to identity theft or any other potential damage.
To better understand how to properly approach data sharing, it is important to understand how data sharing is performed, and which are the risks.
ENISA (European Union Agency for Cybersecurity), in the "Handbook on Security of Personal Data Processing" [4], proposes a set of guiding questions that can be adapted to data sharing scenarios. In particular, when data crosses organizational or contextual boundaries, risk assessment must consider not only the nature of the data but also the actors involved, the communication channels, the possibility of cross-context correlations, and the consequences of misuse. In this case, there are several key questions:

1. **What type of data is being shared?** (identifiers, contact details, sensitive data)
2. **With whom is the data being shared?** (internal department, external partner, third-party platform)
3. **What channel or technology is the data being shared through?** (API, email, cloud storage, social media post)
4. **Is the data combined with other datasets?** (risk of re-identification, profiling, correlation)
5. **How much control does the original data subject have?** (consent, withdrawal, portability)
6. **What protects the data while it is being transferred?** (encryption, access control, contractual protection)
7. **What are some of the potential consequences of unauthorized disclosure?** (identity theft, financial fraud, reputational damage)

Furthermore, **European Data Protection Board**, in "Guidelines 07/2020 on the concepts of controller and processor in the GDPR" [5], identifies three key roles when performing data sharing:

- **Data Subject**: the individual whose personal data is being collected, stored or processed. For instance, a user creates a profile on Facebook where they share their photos or information.
- **Data Controller**: the entity (organization or person) that decides why and how personal data is processed. For instance, Facebook is the controller and decides why and how to gather the data that is shared in the platforms.
- **Data Processor**: the entity (organization or person) that processes the data on behalf of the controller. The data processor is for instance a cloud society that analyzes the data for Facebook and handles data according to what Facebook instructs.

Data sharing has different forms, depending on the actors involved and the scope of the sharing. At an **intra-organizational level**, it allows different departments of the same entity to collaborate by accessing the shared resources.
At the **inter-organizational level**, data may be exchanged between separate companies, institutions, or partners. In other cases, data is released more widely through **public or open data** initiatives.
Finally, in the context of **user-generated content**, which characterizes social media platforms, individuals become the primary source of data sharing, by posting personal information, preferences, or media content.
The motivations behind data sharing are different. It supports **innovation and economic value creation**, since it allows combining heterogeneous datasets and the development of smarter and reliable services. It is also an enabler for **artificial intelligence and machine learning**, which require access to large and diverse datasets to achieve higher accuracy and performance. Data sharing promotes scientific collaboration, allowing for faster discovery and ensuring reproducibility. At

the same time, it contributes to the public good through open government data, smart cities, and health research.
Despite its benefits, data sharing also poses several challenges. On a technical level, organizations must cope with **data quality, interoperability, and standardization issues**. On an organizational level, questions of **trust, governance, and accountability** are central to ensure that shared data is managed responsibly. From a legal and ethical standpoint, compliance with diverse frameworks such as the **GDPR** in Europe or **HIPAA** in the United States requires careful consideration of principles such as data minimization and purpose limitation. In addition, the economic value of data raises problems about its **commodification**, since information is increasingly treated as a tradable asset, a concept emphasized in the well-known metaphor of data as "the new oil", which highlights its potential to fuel innovation and economic growth, but also the risks of concentration and exploitation.
In the context of social media, where personal information can be misused or exploited, it is important to pay particular attention to the evaluation of the impact and the **risk**. Risk evaluation allows organizations to identify potential threats and establish their probability and potential impact on data subjects. Rather than relying on generic security controls, a solution based on risk enables proportionate protection aligned with actual threats.
In fact, under regulatory frameworks like the GDPR, risk assessment is mandatory by law rather than an option. Art. 32 of the GDPR mandates that data controllers and processors put security measures in accordance with the risks that individuals are subject to. As a result, risk assessment becomes an underlying process for ensuring compliance with the law and accountability.
The ENISA, in the same book mentioned above [4], resumes the four levels of impact on a table:

LEVEL OF IMPACT	DESCRIPTION
Low	Individuals may encounter a few minor inconveniences, which they will overcome without any problem (time spent re-entering information, annoyances, irritations, etc.).
Medium	Individuals may encounter significant inconveniences, which they will be able to overcome despite a few difficulties (extra costs, denial of access to business services, fear, lack of understanding, stress, minor physical ailments, etc.).
High	Individuals may encounter significant consequences, which they should be able to overcome albeit with serious difficulties (misappropriation of funds, blacklisting by financial institutions, property damage, loss of employment, subpoena, worsening of health, etc.).
Very high	Individuals which may encounter significant, or even irreversible consequences, which they may not overcome (inability to work, long-term psychological or physical ailments, death, etc.).

Figure 2.1: ENISA Risk Impact Levels

These categories help organizations understand and evaluate the level of risks and apply stronger protection based on the level identified. In addition, a risk matrix is proposed:

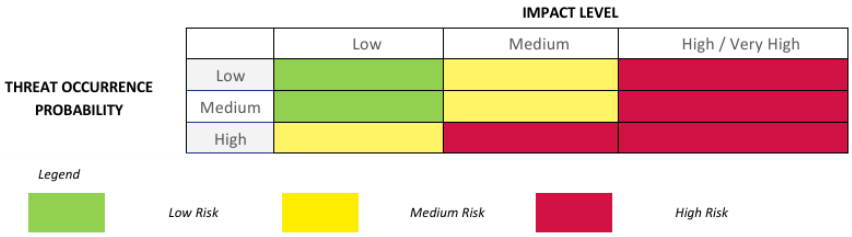

Figure 2.2: ENISA Risk Assessment Matrix

On the **vertical axis** there is the Threat Occurrence Probability:

- **Low**: the threat is unlikely to occur.
- **Medium**: the threat may occur under certain conditions.
- **High**: the threat is likely or expected to occur.

On the **horizontal axis** there is the impact level:

- **Low**: the consequences are low, even if the threat occurs.
- **Medium**: consequences that are significant but manageable.
- **High/Very High**: critical consequences for the system or organization.

Moreover, risks are represented through colors:

- **Green (Low risk)**: threats that are unlikely to occur and have limited consequences. For instance, a minor software bug in a non-critical system.
- **Yellow (Medium Risk)**: events that have moderate probability or impact, such as occasional data leakage of non-sensitive information.
- **Red (High Risk)**: scenarios that are very likely to occur or that would have severe consequences if they did. Such as several data breaches involving personal identifiers.

When the risk matrix is used within data sharing contexts, the probability of a threat and the consequences must be assessed together.
In a **low-risk scenario**, for instance, a user might post a neutral status such as "Good morning everyone!" with no sensitive or personal data. The risk that this information could be exploited is minimal, while the impact on the user's privacy is negligible.
A **medium-risk scenario** could be represented by a person that shares holiday pictures on Instagram through the geotag option. The risk of this data being used to profile or estimate whether a person is not at home is medium, while the impact is high because it exposes patterns of behavior and places.
In the **high-risk category**, the consequence of disclosure becomes severe. For example, posting a phone number or an email address publicly makes it highly likely that the information might be collected by bots or spammers, leading to spam, phishing attempts, or even harassment. An even more critical situation arises when multiple identifiers are shared in a single message, such as a full name, workplace, and current location (e.g., "At a business meeting in Milan with my company"). In

such cases, the probability of misuse is high and the potential impact significant, as the data can facilitate doxing, stalking, or professional exploitation.
Finally, in data sharing the risk derives from:

- **Oversharing**: the user discloses more than intended. For instance, a user shares sensitive data that can be combined and used for malicious purposes.
- **Inferences**: pieces of information that appear harmless on their own but, when aggregated, can reveal sensitive insights or lead to unintended conclusions.
- **Contextual risk**: the level of risk associated with the same piece of information may vary significantly depending on the context in which it is shared.

Ultimately, beyond rules, evaluation methods, and matrices, the purpose of risk assessment is to protect the fundamental rights and freedoms of data subjects. This perspective introduces the concept of data privacy, which encompasses both the legal and ethical dimensions of handling personal data.

2.1.1 Data Privacy and GDPR

Data is considered an important resource that leads to an enhancement in social and economic sectors. Data and how it is handled have huge importance. In fact, data requires careful handling of privacy, ethics and governance.
It is important to consider that there is an increasing and wide amount of risks for users or organizations that decide to share data in a platform. That information can be accessed by everyone and can be used for specific purposes, even with malicious aim.
There lie the importance and the concept of *privacy* which has evolved through time. Traditionally, privacy was intended as physical protection, like the walls of a house. The walls of a house provided not only shelter, but also a form of protection that ensured daily activities could be carried out without unwanted observation. In many societies, privacy was therefore closely associated with property and wealth, meaning that only those who could afford a house, or spaces with physical barriers, could enjoy a tangible level of personal protection.
During the Cold War, the use of digital technology for surveillance became crucial as it allowed a lot of personal data of citizens to be collected. People feared being classified according to their political preferences or religious beliefs. This allowed the emergence of the first generation of data protection.
The first generation of data protection addressed issues like government surveillance, digitization and data collection. However, the major concern was the misuse of this data. Those laws focus mainly on **transparency**, as people wanted to know who was collecting their data and how it was processed. In this context it was created a **data protection authority** (DPA), that was an independent body responsible for protecting and monitoring how the personal data of individuals were handled. The system had an **authorization model**, meaning that anyone wanting to create a database to collect data needed for the approval of the DPA, which would approve it only if the data were being collected in a legitimate way. Another aspect was the **procedural approach** where, instead of waiting for a violation, the laws aimed to prevent risk by setting rules for how data should be collected,

processed and protected from the beginning. This approach led to what became the **GDPR**.
There was also the second generation of data protection that moved from the authorization model to a **notification model**, where organizations simply informed authorities about data processing. During this phase, data protection was formally recognized as a fundamental right, distinct from yet closely connected to the right to privacy.
The third generation faced fragmentation across member states, leading to the adoption of international frameworks such as the **OECD** guidelines and the **Council of Europe's Convention 108**. With the rise of the internet and commercialization of personal data in the 1990s, the focus shifted to private companies as major data controllers. In this context, **consent** emerged as a central principle, giving individuals greater control over their data and reinforcing protection as a matter of self-determination. This principle would later be codified and expanded in the GDPR, where in the Art. 4 [6] defines *consent* as *any informed, free and unequivocal indication of the data subject's will, provided with full awareness and manifested through a clear statement or affirmative action authorizing the processing of personal data.*
This evolution is part of a wider transformation of the concept of privacy itself, which gradually detached from its purely physical dimension and came to be understood in terms of control over personal information, a perspective that underlies the GDPR framework. The main idea is that individuals should have the right to control what others know about them. The concept of privacy, with the advent of modernity, no longer relies on 'hiding behind the walls of the house', but on managing the flow of personal data across networks, platforms and organizations. This transition marks the shift from privacy as *secrecy* (meaning something that should be kept hidden from the others) to privacy as **data governance and autonomy over the information**.
The concept of **personal data** is used in the **General Data Protection Regulation** (GDPR), which was issued by the European Union (EU) and came into force in May 2018. The GDPR in [6] defines personal data as *any kind of information that can identify a person directly or indirectly.* This includes name, ID, number, location or characteristics linked to their physical, genetic, economic or social identity. The GDPR represents the most recent evolution of data protection regulation in Europe. It builds upon principles such as consent and procedural fairness, but distinguishes itself through a strong emphasis on risk management.
This shift was necessary due to the social and technological changes that occurred between the adoption of previous directives and the introduction of the GDPR. For example, in the 1990s, using a loyalty card at a supermarket only gave benefits. Today, the same card can be used to profile, track purchases and influence advertising decisions. This change means that consent is no longer enough to protect individuals. If people do not fully understand what they're consenting to, it becomes difficult to ensure meaningful protection. For this reason, the GDPR emphasizes **risk management**.
The key elements of the GDPR are:

- **Data centric framework**: the main actors are the data controller, data processor and data subject. It's important to understand their responsibilities

to ensure compliance.

- **Individual rights and a rights-based model**: the regulation enforces the individuals' dimension, allowing people to have a set of fundamental rights such as the right to be informed, the right of rectification of inaccurate data and several others that enforce how their personal data is being processed.
- **Legal grounds for data processing**: data processing is lawful only if based on a valid legal ground. The GDPR specifies several bases, including consent, contractual necessity, compliance with legal obligations, protection of vital interests, performance of tasks in the public interest, and legitimate interests pursued by the controller.
- **Role of risk management**: Data Protection Authorities (DPAs) are responsible for assessing risks and implementing measures to mitigate them. This focus on risk assessment helps ensure that personal data is handled safely and responsibly.

In the European framework, the GDPR adopts the broad notion of personal data, which includes not only direct identifiers but also any information that may reasonably lead to the identification of a natural person. In the United States, standards, regulations, and laws make this concept more specific by using the term **Personally Identifiable Information** (PII). The information that relates to an individual can be used to discover or infer their identity. For instance, the name, email, location, IBAN or telephone number.
NIST SP 800-122 (Guide to Protecting the Confidentiality of Personally Identifiable Information) [7] defined PII as any information that is linked to an individual, such as medical, educational, financial or employment data, and can be used to distinguish or trace their identity. In particular:

- To **identify or distinguish** an individual means being able to refer to a natural person directly or indirectly. Identifiers under the GDPR, including names, ID or passport number, numbers, biometric data or any other attribute capable of directly identifying a human being.
- To **trace** an individual refers to the act of collecting and processing enough information to reconstruct or monitor a person's actions, behavior, or condition. An example is represented by user action audit logs, which allows to trace an individual. Under the GDPR, such tracing qualifies as processing of personal data and for this reason require a valid legal basis and appropriate safeguards.
- **Linked information** refers to data that is already logically associated with other information about an individual within the same dataset or system. On the other hand, **linkable information** refers to information that even if not directly connected can be combined with other sources to reveal or infer the identity of a person. Under the GDPR, both categories are considered personal data: Recital 26 clarifies that identifiability also includes cases where identification is reasonably possible through means that are likely to be used.

Examples of PII are [7]:

- **Contact information**: e.g., email address, physical address, and telephone numbers.
- **Online information**: e.g., Facebook and other social media identifiers, passwords, and PINs (personal identification numbers).
- **Geolocation data**: from smartphones, GPS devices, and cameras.
- **Device address**: such as an IP address of a device connected to the Internet or the media access control (MAC) address of a device connected to a local area network.
- **Medical records information**: such as prescriptions, medical records, exams, and medical images.
- **Biometric and genetic information**: such as fingerprints, retinal scans, and DNA.
- Information about an individual that is **linked** or **linkable** to one of the above, like date of birth, place of birth, race, religion, weight, activities, geographical indicators, employment information, medical information, education information, financial information.

2.1.2 Pseudonymization

Another important concept is given by Art. 4 of the GDPR relates to pseudonymization [6], that means altering personal data so it can not be linked to a specific person, unless extra information is given. The extra information is kept separate and protected with proper technical measures.
The GDPR definition reflects the principle of separating direct identifiers from the rest of the dataset. While it does not offer full anonymity, it can be used as an important safeguard that lowers the risk associated with personal data processing. The pseudonymization applies entity recognition, context-aware risk evaluation, and substitution of sensitive information with generic placeholders such as **[PERSON], [EMAIL], [LOCATION]**. This allows giving a solution to ensure that personal data is not immediately accessible and at the same time allowing the text to be used for communication or analysis. In this context, organizations should select and implement technical controls, including:

- **Data masking** which involves concealing parts of information when being stored or transmitted. This can be performed through pseudonymization, data obfuscation, data de-identification, or data scrambling.
- **Encryption** is the process of transforming readable information into an unreadable format using mathematical algorithms and cryptographic keys. Through this technique, only someone with the correct key can decrypt the information and turn it back into readable form.
- **Protecting privacy-related metadata**, such as document attributes or descriptive information that may contain personal details such as the name of the person who last updated a file.

2.1.3 Privacy by Design

The principle of **Privacy by Design**, introduced by **Art. 25 of the GDPR** ([8]), requires that *the protection of personal data should be integrated from the very beginning of the system design.*
The goal is to make privacy a structural and proactive component of every process, rather than a reactive or additional layer applied after development.
The concept of Privacy by Design was first formulated in the 1990s by **Ann Cavoukian** ([9]) and was later incorporated into the GDPR as a requirement that is legally binding.
It is based on the idea that privacy and data protection mechanisms must be embedded into the design and architecture of IT systems and business practices, rather than being added as external safeguards. This paradigm shift transforms privacy from compliance obligation into a *core engineering principle.*
The fundamental principles are structured into seven functional principles:

1. **Proactivity and Prevention**: anticipate and prevent privacy risks before they occur, rather than reacting to them after the fact.
2. **Privacy as the Default Setting**: only the data strictly necessary for the specified purpose should be collected and processed.
3. **Privacy Embedded into the Design**: data protection should be an integral part of the system architecture and not dependent on optional configurations.
4. **Full Functionality**: privacy and usability must coexist without unnecessary tradeoffs.
5. **End to End Security** data must be protected throughout its entire lifecycle from the collection to deletion.
6. **Visibility and Transparency**: data processing activities must be clear and verifiable by users.
7. **Respect for the User Privacy**: individuals should retain full control over their personal information.

This principle represents the conceptual foundation upon which the entire system has been built.
This is translated into an architecture that is **proactive, minimal and transparent**, where data protection is an inherent characteristic of the technological design itself.

2.2 Threat Modeling

After analyzing the risks related to data sharing, privacy, and security, it becomes clear that a structured process is needed to evaluate such risks step by step. This is the role of threat modeling, a discipline that originated in software and system security but has since expanded to broader contexts.
Threat modeling provides a *systematic way to identify possible threats, assess their likelihood and impact, and determine suitable countermeasures.* Instead of

relying on ad hoc practices, it helps organizations anticipate how vulnerabilities could be exploited and prepared accordingly. In this way, threat modeling supports not only technical security but also the protection of personal data, aligning with regulatory requirements such as the GDPR.
To apply threat modeling effectively, it is necessary to introduce the fundamental concepts it relies on. A fundamental concept is the notion of **vulnerability** which is *a weakness or a flaw in a system, application or process that can be exploited by a threat to compromise confidentiality integrity or availability.* Where:

- **Confidentiality**: prevents unauthorized access, disclosure or theft of information through the usage of techniques such as encryption or access controls.
- **Integrity**: ensures data is accurate, complete and unaltered unless changed in an authorized way through checksums, hashes or digital signatures.
- **Availability**: ensures the data and the systems are accessible and usable when needed by authorized users through redundant systems, backups or disaster recovery plans.

In this context, a *threat* [10] is defined as *some potential (something that may cause damage to the system), unwanted (natural disasters, inadvertent human error, malfunctioning) or intentional (information stealing, denial of service, phishing) source of harm that are capable of compromising one or more pillars of cybersecurity (**CIA Triad**: Confidentiality, Integrity and Availability).*
The damage can involve data, systems, reputation, operation or functionalities. *A threat implies that a vulnerability exists that can be exploited.*
Threat modeling aims to identify, assess threats and mitigation to ensure the protection of valuable resources. A workflow typical of threat modeling is indicated by OWASP (the Open Source Foundation for Application Security) in [11]:

- **Define** the environment under analysis.
- **Establish** initial assumptions that may need revision or rejection as the risk landscape evolves.
- **Identify** possible **risks** or dangers that could impact the system.
- **Determine** and **apply countermeasures** for each identified risk.
- **Test** both the model and the identified risks.
- **Evaluate** the effectiveness of the implemented protections.

The main objective of threat modeling is to improve security and to mitigate the effect of potential threats within a system. Evaluating all the possible things that may go wrong in a system or vulnerabilities is a waste of resources and could be difficult.
In this context, it is important to understand which possible threats to focus on. OWASP proposes the **Four Question Framework** [11]:

- **Assess Scope**: It is important to identify what we are working on (social media platforms), what are the principal assets (such as sensitive data like name, phone, birth date, location), boundaries (it is analyzed the risk related to the sharing and combination of personal data).

- **Identify What Can Go Wrong**: This phase corresponds to the identification of threats. This can be developed in a creative way (brainstorming) or in a structured way through **Threat Modeling Frameworks**, such as **STRIDE** or **LINDDUN**.
  In the context of data sharing there are some categories that may be identified:
  - **Oversharing**: the user shares more information than necessary and will expose himself to risky scenarios such as data theft.
  - **Inference**: data that may seem harmless, but, once is correlated, may reveal sensitive data.
  - **Risk contextualization**: the same information is exposed to different risks relating it to a specific context.
  - **Cross-site Profiling**: the repetition of identical data across multiple platforms facilitate the creation of detailed profiles.
- **Define countermeasures**: For every threat that has been identified, it is needed to establish a way to handle it: acceptance, transfer or recovery. **Acceptance** refers to the decision to acknowledge a risk and take no further actions to reduce it. **Transfer** involves shifting the potential impact of a risk to a third party. **Recovery** refers to the process of responding to and restoring operations after a risk has occurred. In the previously identified categories:
  - **Oversharing**: mitigate with preventive warnings and increased user awareness.
  - **Inference**: mitigate through anonymization techniques and reduction of the granularity of the data that is shared.
  - **Contextualize**: manage with rules that assess the same data differently depending on the platform or context.
  - **Cross-site profiling**: can be mitigated through monitoring of recurring patterns and correlation alerts.
- **Assess Your Work**: finally, it is important to understand what coverage is provided:
  - Have all the critical assets been identified?
  - Are the countermeasures proportionate to the level of risk and correct identified given the context?
  - Do they cover realistic attack vectors?

  This process is fundamental and must be reviewed **periodically**.

2.2.1 Application of LINDDUN

In order to make systematic analysis of privacy threats, it is possible to adopt a reference framework that guides the identification and classification of risks. In this

context the **LINDDUN** model [12] has been chosen, and it is specifically developed for **privacy threat modeling**. LINDDUN is an acronym that identifies seven categories of threats regarding privacy:

- **Linking**: that refers to the possibility to link two or more actions to the same entity.
- **Identifying**: that is the possibility to identify an individual from the available data. The data subject can be identified through leaks or can be easily deducted or inferred.
- **Non-repudiation**: impossibility for the user to deny specific action performed.
- **Detecting**: possibility of determining whether a certain data point or event exists by analyzing the existence of relevant information.
- **Data Disclosure**: that refers to the unauthorized share of sensitive information. This can be explicit or implicit disclosure of personal data.
- **Unawareness**: lack of user awareness regarding the risks arising from data sharing.
- **Non-compliance**: violations of data protection regulations or legal requirements.

The adoption of this model allows analyzing more in depth the attack scenarios linked to data sharing, allowing to have a coherent picture and to make it easily mappable to the concrete cases.

2.2.2 DFD Model

The **DFD (Data Flow Diagram)** offers a high level overview of the system, the processes that compose it, the external entities it interacts with, the data involved and how the data moves among the components. The image below shows the principal elements that compose a DFD diagram:

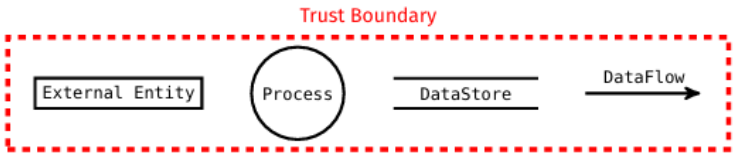

Figure 2.3: Data Flow Diagram (DFD): Elements Overview [1]

It is composed of several elements such as:

- **External Entity (EE)**: that is something outside the system it's communicating with, that can be an end-user or third-party service. Either humans or other systems may be such entities.
- **Process (P)**: that refers to any computation or operation carried out in the system.

- **Data store (DS)**: that is any collection where data is held, such as temporary files, local storage, in-memory data structures, or databases.
- **Dataflow (DF)**: which is used to depict the information flow between the various elements.
- **Trust Boundary (TB)**: it may be defined in many ways. It may define points of interaction between entities at different privilege levels, network or deployment divisions, or the areas within which security elements are applied. They are not required to be included, but if included, their purpose should be explained.

In the context of data sharing and this thesis, an example of DFD could be:

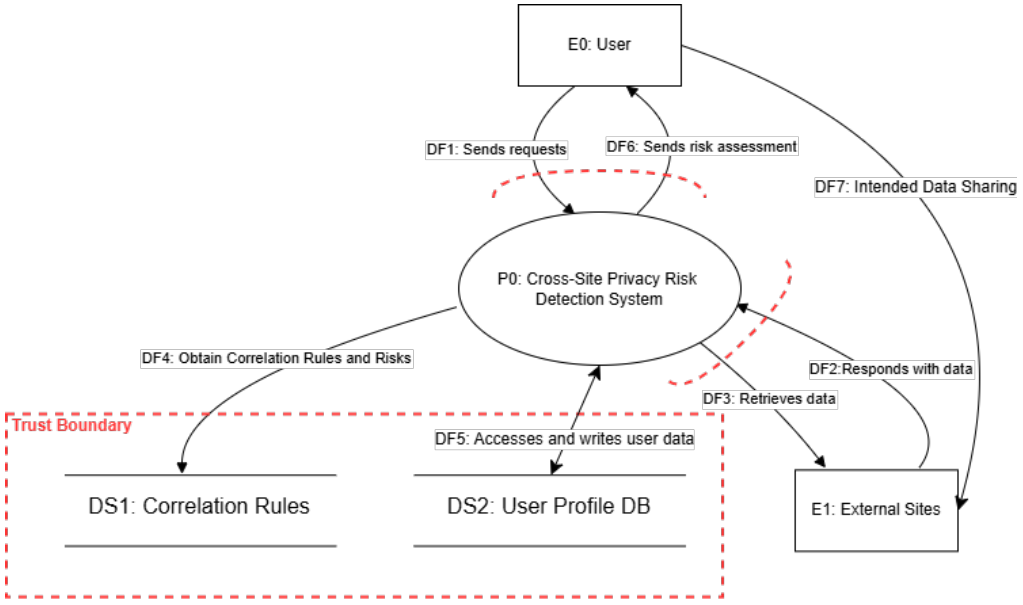

Figure 2.4: Data Flow Diagram (DFD)

The **EE** are represented by:

- **User (E0)**: that is the individual that interacts with the system. It writes a text in the browser's field (messages, posts or notes) and receives a risk report that indicates that the user is sharing sensitive or risky data.
- **External Sites (E1)**: represent the platforms or websites (such as Facebook, LinkedIn, Google Calendar or Instagram) where the user writes. The platforms are external because they do not belong to the system, and they are used to generate the context in which the text of the user is written.

There's a unique process **P0** that is the **Cross-Context Privacy Risk Detection System** which is the brain of the system. In this block there are all the principal functions of the system, such as text analysis, evaluation of the risk based on the context, and correlation between the data. In level 1, even if the system is complex, it is represented in a unique engine that receives inputs and returns outputs. The level 1 breaks down the main processes into sub-processes and provides an overview

of the system's functionality.
There are two Data Store (DS):

- **DS1 - Correlation Rules**: where all the rules are saved (in JSON format) that defines the criteria to understand the risks, the entities and the correlation. This is consulted by P0, but never modified by the user.

- **DS2 - User Profile DB**: is the SQLite DB, where all the entities are extracted by the text are saved and associated with a hashed profile. This allows the system to have a historic memory and apply the correlation rules cross site.

Lastly, the **Trust Boundaries** are three:

- **TB-1 and TB-2**: which are represented by the dashed line that separates the entities from the process P0. These allow distinguishing what is inside the system and what lies outside it. This represents the principal trust boundary.

- **TB-3**: separates the two DS components from the other elements. This indicates that the components are persistent and sensitive data that needs to be protected with specific measures (such as access control, hashing and secure backups).

The DataFlow (DF) of the DFD describes the main interactions between the external entities, the process P0 and data stores.
The flow follows this idea, from E0 to P0: the user writes a text inside the browser's field. This input is captured and sent to the P0, together with minimal contextual metadata such as the timestamp and the current page URL (provided by E1).
The flow from E1 to P0 is used to provide information about the context in which the text is written or whether it belongs to a public post or private message. These contextual elements are essential for evaluating the exposure level of the user's data.
Moreover, the flow that goes from P0 to E1, allows the process to retrieve contextual data through browser interfaces or APIs. The external sites respond with the requested data, which is used to enrich the contextual analysis and improve risk detection accuracy.
Then, the flow from P0 to DS1 reads from the Correlation Rules data store all the necessary rules written in JSON format that define how to evaluate the risks, detect entities and correlations. The dataflow is read only since DS1 is a static reference resource.
The flow from P0 to DS2, indicates that the process P0 both reads and writes from the User Profile Database (SQLite). When new entities are detected, they are stored and linked to a hashed profile identifier, preserving the user's anonymity. The DB is also queried to retrieve previously stored entities and perform cross-site correlation.
Lastly, after processing the text, the system returns an analysis report to the user. The report includes the anonymized version of the text, the list of detected entities, and the corresponding risk levels based on the context. After receiving the report, the user makes an informed decision and shares information on the external sites.

2.2.3 Elicitation of Privacy Threats

After establishing the DFD of the proposed system, the next step as indicated [1] is the **elicitation of privacy threats**. This part allows mapping all potential risk scenarios starting from the system model (in this case a DFD). This process involves systematically iterating over every interaction in the model, that is, each combination of source, dataflow and destination to determine which LINDDUN threats may apply to each element.
Each element of the system is mapped to the seven categories identified in 2.2.1. The principle is that each element indicated in the DFD can be exposed to different categories of threats and it is analyzed individually to assess whether threats may arise at three distinct levels:

- **Source**: where the data originates or is shared.
- **Data Flow**: where the data is transmitted.
- **Destination**: where the data is received, processed, or stored.

In the case of the system proposed, this corresponds to iterating over the defined data flows (DF1-DF7) illustrated in the figure 2.4. For each flow, possible threats are examined in the context of the system's components: **external entities (E0, E1)**, the main **process (P0)** and **data stores (DS1, DS2)**.
In general:

- **Processes (P)** can introduce risk of **Unawareness**, if users are not informed of the operation that are done with their data or **Non-compliance** if the legal constraints are not respected.
- **Data Flow (DF)**: are susceptible to **Linkability** and **Detectability**, since the data transmitted may be correlated or intercepted.
- **Data Store (DS)**: are typically associated with **Disclosure of information**, since those elements contain sensitive information that may be disclosed in case of compromise.
- **External Entities (EE)**: can be associated with risks such as **Identifiability** or **Non-repudiation**, in the moment in which the interaction with the system allows to link data to a specific subject or denying/attributing actions taken.

The following list summarizes the privacy threats identified for each data flow of the system, considering the source, dataflow and destination elements.

- **DF1: E0 → P0 (Sends Requests)**
  Source: the user could unintentionally share sensitive personal data in the input text.
  Data Flow: repeated inputs could be linked over time, allowing profiling or re-identification of the user across sessions.
  Destination: the system (P0) processes the input, and any lack of minimization or improper handling could lead to privacy leakage.
- **DF2: E1 → P0 (Responds with data)**
  Source: external platforms could expose more information that indirectly

identifies the user.
Data Flow: contextual data exchanged might contain linkable identifiers that allow profiling across multiple domains.
Destination: improper sanitization of contextual data within P0 could reveal user activity across sites.

- **DF3: P0 → E1 (Retrieves Data)**
  Source: P0 initializes requests to obtain contextual information than strictly needed, leading to excessive data collection.
  Data Flow: the retrieval request could contain unique identifiers or cookies that link the system to a specific user.
  Destination: external sites could log these requests and reconstruct behavioral patterns.

- **DF4: P0 → DS1 (Obtain Correlation Rules and Risks)**
  Source: assessing correlation rules does not involve personal data but may influence how personal data is later interpreted or categorized.
  Data Flow: not transparent or outdated rules could lead to biased or unfair classification of user data.
  Destination: inappropriate rules may result in a non compliance with privacy by design principles.

- **DF5: P0 ↔ DS2 (Accesses and write user data)**
  Source: P0 writes detected entities into DS2, weak hashing or pseudonymization could enable re-identification.
  Data Flow: user data in transit between P0 and DS2 could be exposed without data anonymity.
  Destination: DS2 accumulates personal information over time, increasing the risk of inference or secondary use.

- **DF6: P0 → E0 (Sends risk assessment)**
  Source: the system outputs risk reports that might contain partially deanonymized or traceable information.
  Data Flow: if transmitted without protections, the report could be intercepted and reused.
  Destination: users may not fully understand the implications of the risk scores or how their data contributed to them, leading to unawareness.

- **DF7: E0 → External Sites (Intended data sharing)**
  Source: the user shares the written text with the external platform.
  Data Flow: this represents the intended communication that would occur if the user confirms the publication. The text may contain personal or sensitive data.
  Destination: external platforms receive the data once the user decides to proceed, which could result in public disclosure or profiling. The system (P0) aims to prevent this by alerting the user before this flow takes place.

The purpose of this phase is not yet mitigation, but rather the creation of a list of potential threats to the system according to the guiding questions and **threat trees** provided by LINDDUN. Such systematic design steers clear of missing some threats and provides a solid foundation for the subsequent phases of risk assessment

and determination of countermeasures.
After the construction of the DFD, the LINDDUN elicitation phase was carried out to identify potential privacy threats associated with each data flow.
This phase is important because it highlights areas where the privacy risk could theoretically emerge. This assessment is crucial to support the implementation of countermeasures and ensures that the final system aligns with the principles of Privacy by Design and Data Minimization.
Relating to the Level 1 DFD in 2.4, it is possible to map the threats to the elements identified previously in a table:

Table 2.1: Application of LINDDUN to DFD

DFD Element	**LINDDUN Threat**	**Description**
User (EE)	Unawareness	The user may not be fully aware of the type and extent of data processing performed by the system.
External Sites (EE)	Identifiability	The information exchanged with external platforms can contribute to re-identification of the user when it is combined with other data.
Process P0 (Privacy Analyzer)	Unawareness / Non-compliance	The automatic analysis and the profiling may not be transparent to the user, and it could risk non-compliance with legal frameworks such as the GDPR if not handled properly.
Data Flow: User → P0	Linkability	Multiple text inputs from the same user could be correlated, allowing reconstruction of a digital profile.
Data Flow: External Sites → P0	Detectability	An observer may detect that a user is active on a specific external platform, even without knowing the content of the message.
User Profile DB (DS2)	Disclosure of Information	Stores sensitive entities (names, emails, identifiers) that could be exposed in case of unauthorized access or leakage.
Rules (DS1)	Non-compliance	The not correctly defined or outdated rules may lead to processing that does not comply with privacy regulations or exceeds the intended purpose.
Data Flow: P0 ↔ DS2	Linkability / Disclosure	Cross-site correlation queries may expose relationships between data points, increasing the risk of profile reconstruction.
Data Flow: P0 ↔ DS1	Non-compliance	If the risk rules are not updated or aligned with privacy-by-design principles, the system may not meet regulatory requirements.
P0 → User (report)	Unawareness	The report may not communicate risks in a clear way, leaving the user unaware of the real consequences of sharing data.

Documenting and prioritizing threats

After the elicitation phase, in which we have a list of the potential privacy threats, the LINDDUNN methodology specifies a phase of documentation and prioritization of threats:

1. **Documenting Threats**: every threat identified should be written in a clean and complete way including:
    - **ID or unique code**: to refer easily to the threat in the subsequent analyses.

- **Element of DFD involved**: it can be a process, a dataflow, a data store.
- **Corresponding LINDDUN category**: the categories belonging to the seven (Linkability, Identifiability, etc.).
- **Threat description**: a clear explanation of the threat.
- **Privacy Consequences**: such as exposure of sensitive data, loss of anonymity or risk of profiling.

2. **Prioritizing Threats**: Once identified and documented, threats must be prioritized according to criteria derived from the risk analysis:
   - **Probability of occurrence**: how probable it is that the threat will occur, based on the technical and organizational context.
   - **Impact** on the user's privacy: from low severity to high implication.

   A risk matrix can be created combining likelihood and impact:

Impact \ Likelihood	Low	Medium	High
Low	Low	Medium	Medium
Medium	Medium	High	High
High	High	High	Critical

Table 2.2: Risk matrix: Impact and Likelihood for Threat Prioritization

   The combination of these two factors results in a risk matrix that distinguishes between threats requiring immediate action and those that are either acceptable or can be addressed at a later stage.

3. **Expected Outcome**: the outcome of this step is a documented and ordered list of threats, which serves as the basis for the subsequent definition of appropriate countermeasures.
   In this way, LINDDUN not only ensures a wide coverage of potential threats, but also supports a realistic prioritization, directing attention to the most critical threats affecting the system under analysis.

Table 2.3: Example of Privacy Threats Based on LINDDUN Framework

ID	DFD Element	LINDDUN Category	Description	Impact	Likelihood	Priority
T1	User Profile DB (DS2)	Disclosure of Information	Sensitive entities stored in the DB could be leaked in case of unauthorized access.	High	Medium	High
T2	Data Flow: User → P0	Linkability	Multiple text inputs from the same user could be linked together, enabling the reconstruction of a digital profile.	Medium	High	High
T3	Process P0 (Privacy Analyzer)	Unawareness	Users may not be fully aware of the automatic profiling and correlation performed on their data.	Medium	Medium	Medium

Chapter 3

Natural Language Processing

The **AI System**, according to the Art.3 of EU Artificial Intelligence Act [13], is a *machine-based system designed to operate with varying levels of autonomy, that may exhibit adaptiveness after deployment and that, for explicit or implicit objectives, infers, from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments.*
Natural Language Processing (NLP) is a subset of Artificial Intelligence (AI) that uses **Machine Learning** (ML) to allow computers to *understand and emulate human language.*
NLP allows computers to generate, recognize and understand, both text and speech, by combining computational linguistics (the rule based modeling of language) with statistical modeling, machine learning and deep learning. NLP is already part of everyday life, as it is used in search engines (such as Google or Bing), customer service chatbots, voice commands and digital assistants.
The benefits of NLP include:

- **Automation of routine processes**: it can handle repetitive tasks that involve language, saving time and reducing human intervention in basic operations.
- **Enhanced data analysis and insights**: organizations, with the use of NLP, can analyze massive quantities of unstructured text data to uncover patterns, sentiments or trends to make better decisions.
- **Advanced information retrieval**: it makes searching more intelligent and it has context aware search capabilities.
- **Content Creation**: it allows much faster content production while maintaining a good amount of coherence and relevance.

Through the application of **text mining**, that is the process of analyzing and extracting meaningful information from large collections of unstructured text, NLP can identify patterns, trends and sentiments that would not be immediately apparent in a large dataset. NLP can be used for different tasks, including:

- **Sentiment Analysis**: classifies the emotional intent of a text. Typically, it is

based on **hand-crafted features** (lexicon-based polarity scores, presence of emoticons, punctuation counts) or **automatically learned features (word n-grams, or deep learning (DL) models)** that capture both long and short term dependencies in the text. It can be used to classify customer reviews or detect signs of mental illness in online comments.

- **Toxicity classification**: a specialization of sentiment analysis that aims to reveal not only the hostile intent, but also specific categories such as threats, insults or hate speech towards group of people. This model receives in input a text and will return the probability of belonging to each class.
- **Machine Translation**: it allows to automate the translation from one language to another one. Starting with a specific language, the model will generate the equivalent text in the target language (as in systems like Google Translate).
- **Named Entity Recognition (NER)**: identifies and classifies textual elements into predefined categories such as name (PER), organization (ORG) or location (LOC). The input is generally raw text, and the output consists of the detected entities along with their start and end positions.

In the context of this thesis, the task of NLP is not merely to understand general language, but to fulfill a specific purpose: identifying and anonymizing sensitive information that is present in user provided text using a computer.
In greater detail, NLP techniques are employed to perform Named Entity Recognition (NER), that is, the identification of names, places, organizations, email addresses, phone numbers or bank accounts. Once extracted, entities can be anonymized and, more importantly, assigned a risk level in accordance with the context in which data is shared.
This situational approach illustrates how NLP can enforce privacy and security by design, by detecting potential threats such as identity theft, doxing or profiling through cross-correlation of seemingly harmless data.
In this sense, NLP serves as a methodological bridge between threat modeling analysis and raw textual information, because it transforms unstructured text into structured representation that can be examined rigorously for privacy threats.

3.1 Named Entity Recognition

Named Entity Recognition (NER) is a fundamental subtask of Natural Language Processing (NLP). NER involves the identification and classification of named entities, such as names of individuals, organization and location.
The task of automatically detecting and mitigating privacy and security risks in user generated content is often a **Sisyphean struggle**. As in the myth of Sisyphus, where the rock inevitably rolls back despite relentless effort, NLP applications must face a constantly evolving landscape of malicious behaviors, linguistic nuances, and emerging forms of sensitive information disclosure. This implies that, while progress is both possible and desirable, the work is inherently ongoing and requires continuous adaptation of models, regulatory frameworks, and contextual awareness.
Traditionally, NER relied on:

- **Rule-based Methods**: systems that relied on manually designed patterns and linguistic heuristics to detect entities.
- **Feature-Based Machine Learning**: algorithms such as Conditional Random Fields (CRFs) or Support Vector Machines (SVMs) and Decision Trees were trained on engineered linguistic features, such as word morphology or pre-computed word embeddings.
- **BiLSTM-CRF**: Bidirectional Long Short Term Memory combined with CRFs became the state of the art. These models were highly effective at capturing sequential dependencies and contextual information within text. The problem was the fact that those models were sequential (and therefore slow) and struggled to capture long range dependencies across sentences or documents.

Those models were the base and allowed the introduction of the **Transformer** models, and in particular of **BERT** which has then become the standard.

3.1.1 Working Principles

To comprehend better how the pipeline of NER works, it is useful to understand some key concepts.
The first central component is the **Tokenizer**, which is the component responsible for splitting text into smallest elements called **tokens**. Contrarily to a standard approach, which considers entire words, modern language models prefer a more subtle segmentation, outputting sub-tokens. This means that rare or compound words are split down. In this manner, the model has a better chance of handling non-vocabulary words, skipping the Out-Of-Vocabulary (OOV) phenomenon (that refers to words that are not present in the vocabulary or training data).
Another important step is **Labeling** where each token is assigned to a label according to whether it is part of an entity.
Another essential concept is that of the model used for training and prediction, in general there are two possible approaches.
On the one hand, one could **train a model from scratch**, starting with random parameters and training it on a large and annotated dataset until it learns how to recognize entities. Although this approach offers maximum flexibility, it is extremely costly in terms of time, computational resources, and data requirements, as it demands a very large number of manually labeled examples.
On the other hand, it is possible to rely on a **pre-trained model**, that can be a deep learning model that has already been trained on massive amounts of unlabeled text to capture general linguistic patterns such as syntax, semantics, and contextual relationships between words. This prior knowledge can then be transferred and fine-tuned for specific downstream tasks like NER.
In this thesis, the chosen model is **dslim/bert-base-NER**, a model derived from the Transformer architecture, which was fine-tuned to recognize entities such as persons, organizations, or locations. The presence of a pre-trained model makes it possible to achieve good performance without having to endure the costly process of training from scratch.
Another step is the **inference** that occurs after training and is used to predict entity labels on new and unseen text.

Lastly, we have **evaluation**, where the model performance is evaluated with different metrics such as **Precision**, **Recall**, **F1 Score**. **Accuracy**, in this case, is not used because the classes are imbalanced and this measure would not give faithful results.
After analyzing the individual components, the overall workflow of NER can be summarized as shown below:

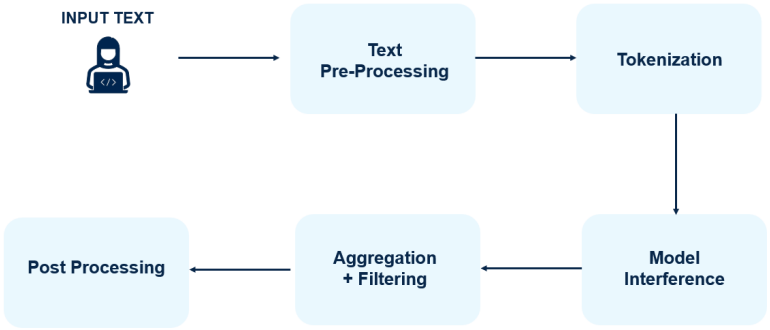

Figure 3.1: Named Entity Recognition Workflow

The process begins with an **unstructured input** or input text. The input could be, for instance, "Mario Rossi works in Milan for a firm called FinTech Spa". The system preserves the integrity of the original text for an accurate offset calculation (start and end indices) which is useful to handle spaces, punctuation such as commas, periods, colons or typographic apostrophes.
The first stage is the **preprocessing** phase, in which the text is divided into sentences, and then tokenized into words based on spaces or symbols. After that, it breaks rare or compound words into **sub-tokens** (such as Fin, ##Tech) and reduces the Out Of Vocabulary (OOV).
For each token/sub-token, the **tokenizer's offset mapping** is used to understand which character indices it corresponds to in the original text. This alignment is important, since any transformation that changes the length of the text makes the offset alignment more difficult. For this reason, it is important to normalize the text, preserving index consistency.
Once we have clean tokens and offsets, the next step is the **model inference**. At inference, the tokens of the tokenizer are passed through the model, and it computes for each of them the probability of belonging to a particular class (e.g., PER for person, LOC for location, ORG for organization). These predictions are typically expressed in the **BIO tagging scheme** (**B**, **I**, **O**), which indicates whether a token marks the **beginning**, **inside**, or **outside** of a named entity.
The result is a list of tokens with their corresponding predicted labels and confidence scores. To make such predictions more readable and consistent, an aggregation method is used. This is done through the aggregation of contiguous sub-tokens of the same entity and generating one and unambiguous span. As an example, if the model is given "Fin" and "##Tech" and both are tagged ORG, the **aggregation** module reconstructs them into the entity "FinTech."
After inference, we have a list of recognized entities with:

- the extracted text span (obtained via offset mapping),

- the predicted label (person, location, organization, etc.),
- the associated confidence score.

The following stage is **filtering**, a process that discards inconsistent or low confidence predictions. There is usually a threshold of confidence applied, below which an entity is not considered valid. Filtering also resolves conflicts: for example, if a string has already been recognized as an email address by one rule based on regular expressions, it should not also be labeled as a person (PER).
Finally, the **post-processing** phase, where the output is adapted to the aims of the application. For this task, post-processing not only replaces entities with anonymized tags (such as [PERSON] or [EMAIL]), but also appends each data type with a specific risk class (e.g., personal names are "Very High", locations are "Medium"). The results can be utilized to construct a risk overview or spot dangerous correlations when identical information is presented on multiple web sites.
Overall, these multiple stages pipeline ensures accuracy, with the combination of regex for precise patterns and ML for context dependent entities, and usability, since the offset mapping enables precise anonymization and detailed reporting.

3.1.2 Evaluation Metrics

The performance of the model usually are measured with:

- **Precision**: the fraction of identified entities that are correct (considering the errors).

$$P = \frac{TP}{TP + FP}$$

- **Recall**: the fraction of actual entities that the model successfully recognized (ignoring errors).

$$R = \frac{TP}{TP + FN}$$

- **F1-score**: the harmonic mean of Precision and Recall, balancing the two measures.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Where:

- **True Positives (TP)**: are the entities that are present in the text and have been correctly identified and classified by the model. For instance, the model correctly tags "Milan" as a location (LOC).
- **False Positives (FP)**: are the entities predicted by the model that are not actually entities or are assigned to a wrong class. For example, the model wrongly tags "firm" as a location (LOC).
- **False Negatives (FN)**: entities are present in the text, but the model fails to detect them. For instance, the model does not recognize "FinTech Spa" as an organization.

3.1.3 Limitations

Despite significant progress, there are still significant challenges:

- **Semantic Ambiguity**: the same word belongs to different categories depending on the context. For example, "Amazon" may refer to either a company (ORG) or a rainforest (LOC).
- **Domain-specific constraints**: generic pre-trained models struggle with highly specialized domains such as medicine, finance or law. In this context, using a dedicated dataset is crucial to achieve acceptable performance.
- **Out of Vocabulary Words**: new entities may render the tokenization more difficult, even though the sub-tokenization technique aims to minimize the problem it does not eliminate it completely.

3.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) was introduced by **Google** in 2018, and it is based on **Transformer architecture**. The main innovation that brought BERT was the **bidirectionality**, that allows for every word to be read both looking at the left and right context. This allows the model to build a more accurate representation of the language compared to the previous approaches that processed data only left to right and right to left. This property is important for the Named Entity Recognition (NER). The majority of entities are inherently ambiguous and need to be interpreted only by context. For example, the word "Apple" can refer to either the fruit or the company depending on words of context. BERT's bidirectional attention enables the model to disambiguate and improve entity classification.
BERT was released as a pre-trained model, which was pre-trained on large generic databases such as **Wikipedia** and **BookCorpus**. Through this pre-training, BERT achieves a generic statistical knowledge of natural language. However, it also has the capability to be fine-tuned for specific applications such as NER by further training of the model over annotated datasets. **Fine-tuning** allows the general linguistic understanding acquired during pre-training to be redirected toward a more specific purpose.
The model used is **dslim/bert-base-NER**, which is already a pre-trained BERT fine-tuned for Named Entity Recognition. The model was trained on the **CoNLL-2003** (Conference on Natural Language Learning 2003) dataset, which is an **English Language Corpus**. The initial part of the name *dslim* is the username of *David S. Lim*, a researcher and natural language engineer who fine-tuned and published several BERT based models on **Hugging Face**, focused on NER tasks ([14]). His main contribution was adapting *Google*'s bert-base-cased model and fine-tuning it on the CoNLL-2003 dataset for the entity recognition task. The model can identify common entity types such as PER (person), LOC (location), ORG (organization), and MISC (miscellaneous).
The usage of a pre-trained model has several practical advantages: it avoids the manual execution of the computationally costly fine-tuning process, offers strong

performance on general datasets, and integrates well into the HuggingFace Transformers framework used.

3.2.1 Model Characteristics

The model `dslim/bert-base-NER`, which is based on the BERT architecture, has several positive characteristics over traditional approaches:

- **Contextual understanding**: Since BERT is bidirectional in the sense that it uses both left and right context for decoding tokens, the model can better disambiguate words with multiple senses.
- **Sub-word tokenization**: employing WordPiece tokenization, the model can process out-of-vocabulary or complex words easily, escaping the Out-of-Vocabulary (OOV) problem.
- **Pre-training on large corpora**: since BERT is pre-trained on massive generic text, it possesses a strong prior language knowledge and therefore needs less labeled data for fine-tuning.
- **Fine-tuning for NER:** the `dslim/bert-base-NER` model was specifically fine-tuned on NER benchmark dataset (**CoNLL-2003**) and achieves state-of-the-art performance on general entity types like *person*, *organization*, and *location*.
- **High Performance**: it achieves high performance when evaluating the model.

However, BERT does not entirely eliminate the shortcomings of NER. Certain limitations continue to exist:

- **Domain adaptation**: although good on general-purpose text, performance is significantly weakened in specialized domains.
- **Semantic ambiguity**: although reduced, entity classification remains ambiguous in the presence of many different possible meanings per context for a word.
- **Computational cost**: BERT models are resource-intensive, making their deployment in real-time applications or low-resource environments challenging.
- **Bias**: there may be some biases introduced if they are already present in the pre-trained model.
- **Tokenization Issues**: there may be some problems with dividing the words in sub-words.

BERT does not solve all issues native to NER but provides a solid state-of-the-art basis for generic texts. In practice, achieving optimal performance requires additional steps, such as fine-tuning on domain-specific datasets, integrating rule-based components (e.g., regular expressions for highly structured entities), and applying post-processing strategies to handle ambiguous or nested entities. This layered approach highlights that while BERT establishes a strong baseline, effective NER

solutions typically emerge from a combination of machine learning and complementary techniques.

Chapter 4

Thesis Objective

Over the last decade, the growing reliance on digital platforms has radically changed the way people share information on the internet. Social networks, e-commerce websites, cloud computing platforms, and group collaboration tools require users to share large volumes of personal data. While these platforms offer convenience of access and connectivity, they also raise serious concerns regarding data security and privacy. Information that might look harmless in isolation, such as a name, a date, or an address, can become highly sensitive when combined, aggregated, or exposed in inappropriate context. The risk is amplified by the fact that users lack awareness of who can access and view their data or how that information can be correlated and used. This leaves the door open to profiling, identity theft, or even more severe threats such as doxing and stalking.

Existing technologies that preserve privacy are usually implemented for specific actions, such as alerting on established keywords or enforcing generic security policies. These approaches are typically insufficient: they are not aware of data-sharing context and do not consider the cumulative effect of subsequent disclosures on many sites and apps. As a result, the gap between increasingly sophisticated exploitation techniques and the simplicity of protection tools continue to grow.

This thesis bridges this gap by proposing the design and development of an intelligent proactive privacy risk detection and mitigation system. The system combines rule-based methods (Regular Expressions (regex) for structured identifiers) with state-of-the-art Natural Language Processing models based on Transformers (BERT) for high-precision detection of a wide variety of sensitive entities. More importantly, the analysis goes beyond entity recognition to incorporate a hierarchical risk model that considers the context in which the information is shared. This framework considers not only what type of data is being shared, but also where and how it is disclosed, and how data from multiple platforms can be correlated to reveal deeper insights.

A second critical objective of the project is to provide users with real-time anonymization and feedback mechanisms. Sensitive information is de-identified through visible placeholders, such as [PERSON], [LOCATION] and [ORG], that preserve text legibility but prevent exposure. At the same time, risk levels are communicated through a clear reporting interface, making privacy protection actionable and meaningful in everyday browsing.

Another novel contribution is the addition of long-term memory and cross-site correlation analysis. By storing anonymized entities in a local database in a secure fashion, the system builds a digital profile of the user that evolves over time. This makes it possible to detect privacy threats that are not visible in a single instance, such as the same personal data appearing on multiple domains or unsafe combinations of data (e.g., a name with a date of birth or a location). These cross-site threats are particularly important, as they can reveal personal habits, relationships, or even physical whereabouts.
Finally, for usability and accessibility, the system has been implemented as a Chrome browser extension with a Flask-backed analysis engine. This implementation allows the system to be able to operate in the background unobtrusively, analyzing text as the user types and providing immediate feedback through a non-intrusive interface. In doing so, this thesis contributes not only to the advancement of privacy risk analysis methods but also delivers a practical tool that empowers individuals to better safeguard their personal data in real-world scenarios.

Chapter 5

Methodology and System Design

This chapter presents the methodological approach that guided the development of the system. After having explained the main risks involved in data sharing and defining the objectives of the thesis, it becomes necessary to adopt a structured process to move from theoretical considerations to the practical implementation of a functioning prototype. The approach chosen combines threat modeling techniques with iterative system design and testing, ensuring that what was seen theoretically can be applied to real-world scenarios.

5.1 Methodological Approach

The methodology used for developing the system can be resumed with the following schema:

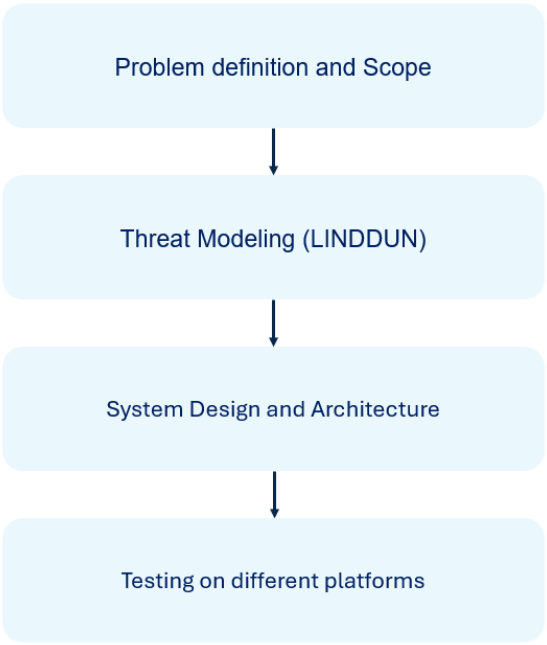

Figure 5.1: Proposed Methodological Framework

The approach taken in this thesis was structured in a series of incremental phases, from problem definition to system design and implementation, and then to its iterative refinement. The overall objective was to understand how sensitive information is shared online and create a system capable of detecting, anonymizing, and quantifying privacy threats across different environments.
The research began with the delimitation of the problem and research site. Particular emphasis was laid on the way users share personal information on web platforms such as Facebook and Google Calendar. Posts, comments, private messages, and calendar events are all forms of interaction, and each one unleashes personal information at various levels of risk. By doing this analysis, the most relevant threats were identified, **identity theft, doxing, and profiling**, thus guiding the needs of the system to be constructed.
After having established the scope, the next step was to select an appropriate modeling framework for privacy threats. Different methodologies are established and used for threat modeling, **LINDDUN** was chosen out of the ones available because is highly adaptable to privacy related contexts. This technique allows a logical approach to consider how data passes through a system and how such movements might be misused. By specifying potential dangers to specific interaction and environment, it was then feasible to construct the design upon a sound methodology foundation and not simply ad hoc assumptions.
On top of this foundation, the work went on to system design and architecture. The system was implemented as a two parts solution. The **backend engine** constitutes the first part and it is employed to recognize entities from text, anonymize them, and determine the risks posed by them. The second part is a **browser extension** that constantly observes user input and provides immediate feedback in the form of contextual alerts and extensive reports.
This allows a more **user-friendly threat analysis** that can make users aware of the inherent risk of sharing personal information in different contexts.
To enable the system to go beyond single page analysis, a **database** was added as well. This allowed anonymized entities to be stored and added to user profiles in order that correlations across different websites and overtime could be found.
The implementation phase relied on a combination of complementary techniques, each selected according to its suitability for a specific task. The core system was designed to balance general-purpose text analysis with structured pattern recognition, thereby ensuring that both unstructured and structured data could be effectively handled. To guarantee flexibility, the logic for assessing risks and correlations was externalized into configuration files rather than being hard-coded, allowing the system to remain adaptable to future changes. On the user side, a lightweight client component was developed to provide real-time feedback and interactive reports, ensuring usability and accessibility.
The project evolved through repeated cycles of testing and refinement. The earliest prototypes focused solely on text anonymization. Later iterations integrated contextual risk assessment, and finally, long-term correlation capabilities were introduced, enabling the detection of privacy risks emerging across platforms and over time. Each cycle contributed to improving the precision of detection, refining the risk rules, and enhancing the user interface.
The outcome is a system that not only analyses sensitive information in isolation but also accounts for contextual and temporal relations, thus providing a broader

perspective on privacy risk.
The following sections describe the rationale behind the adopted methodology, the alternative approaches considered, the incremental refinement of the system, and the criteria used to assess its effectiveness.

5.1.1 Threat Modeling Method Selection

In Threat Modeling exists various options, each with their own scope and focus.
An example is **STRIDE**, one of the most used and adopted frameworks in security engineering. It is particularly effective in identifying security threats such as **spoofing**, **tampering**, or **denial of service**. However, the focus of this method is not suitable to privacy-specific risks such as **identifiability**, **linkability** or **information disclosure**.
Since the objective of the thesis is to build a model focused on how personal data may expose users to different risks such as **identity theft** or **profiling**, a methodology limited to security concerns would not been sufficient.
Another methodology is **PASTA** (Process for Attack Simulation and Threat Analysis), that focuses on **attacker-centric perspective**. This helps simulate potential adversarial strategies against the system. This requires a detailed system specification and attacks scenarios, which were not aligned with the more exploratory and user-focused goals of this research.
For those reasons, **LINDDUN** was adopted. Unlike STRIDE and PASTA, this methodology was specifically built to address privacy threats, organizing them into categories such as linkability, identifiability, non-repudiation, detectability, information disclosure and policy non-compliance. This allowed having a structured reasoning framework, directly tied with the privacy domain.
The main advantages of LINDDUN are:

- **Privacy-oriented taxonomy**: which can be directly linked to the purpose of the research.
- **Flexibility**: it can be applied to several scenarios, in particular social networks, messaging services and calendar platforms.
- **Systematic Process**: avoid ad hoc assumptions and provides a reproducible approach.

On the other hand, it also has some limitations. This methodology requires significant effort in mapping the data flows and can be time-consuming in situations that rapidly change. Furthermore, while it is good for qualitative reasoning, it provides less quantitative reasoning compared to the other methodologies.
Despite the aforementioned weak points, the benefits overcame the drawbacks since LINDDUN provided the most appropriate compromise between methodological thoroughness and usability in the privacy domain.
In the table below, the main characteristics of the most common threat modeling frameworks are summarized in Table 5.1, which provides a comparative overview of their primary focus, applicability and limitations.

Table 5.1: Comparative Overview of Threat Modeling Frameworks

Framework	Primary Focus	Applicability	Limitations
STRIDE	Security threat taxonomy (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege).	Useful for mapping classic security threats at the system or component level.	Security-centric; limited coverage of privacy-specific risks; may overlook data correlation across contexts.
PASTA	Attacker-centric, business-driven threat modeling.	Suitable for detailed system specifications and red-team analyses.	Heavyweight; requires comprehensive attacker models; less suited for exploratory or user-focused privacy analysis.
LINDDUN	Privacy-oriented threat modeling (Linkability, Identifiability, etc.).	Directly targets privacy; fits goals such as profiling, doxing, and identity theft; adaptable to multiple contexts.	Mapping data flows can be time-consuming; offers qualitative rather than quantitative guidance.

5.1.2 Incremental Development and Refinement

The system was created through a series of iterative incremental prototypes. The first prototype worked only on anonymization, in this way sensitive information would be appropriately masked and replaced by neutral placeholders. This provided a foundation for what was created in later improvements by demonstrating that the system could handle personal data in a structured way.
The second prototype added contextual risk assessment, which allowed the system to go beyond anonymization and consider the sensitivity of data in relation to the context in which it was shared. This aspect was a dramatic shift from a purely functional protective tool to an analytical system that can quantify privacy risks.
The final deployment included further scope by introducing a persistent database and correlation functions. This feature made it possible to detect risks that exist only over time or across platforms. Testing and iteration followed each deployment, allowing a refined system.
The system in this way was transformed into a more robust and user-friendly solution.

5.1.3 Criteria for Assessing Effectiveness

To ensure that the system fulfilled its intended objectives, its effectiveness was assessed according to both technical and user-oriented criteria.
From a **technical perspective**, the following aspects were considered:

- **Detection and anonymization accuracy**: the system has to correctly identify and mask sensitive entities with low error rates.
- **Robustness**: the methodology should remain consistent with different forms of data as well as in different online environments.
- **Coverage of critical risks**: the framework was evaluated on its ability to capture high-impact scenarios.

From **user perspective**, the assessment focused on:

- **Report readability**: the results had to be understandable for non-expert users, highlighting the nature of the risks in an accessible way.
- **Timeliness and usability**: feedback was required to be given in a timely and non-intrusive manner to facilitate users to always be aware of potential risks without disrupting their usage of digital platforms.
- **Flexibility**: the system should remain adaptable, with rules and thresholds modifiable to respond to new privacy challenges.

These common criteria ensured that analysis of the system was not limited to its technical efficiency but also considered its usability and applicability in real-world scenarios.

Chapter 6

Implementation

This chapter is dedicated to the development and technical part of the implemented system. The aim is to illustrate how the theoretical concepts seen in previous chapters, together with the system design in chapter 5, were translated into a functional tool.
The system operates as a full-stack application composed of two interacting components: a *Flask backend* serving as the central engine and a *Chrome extension* acting as the user interacting interface. A brief overview of the system is shown in the figure below:

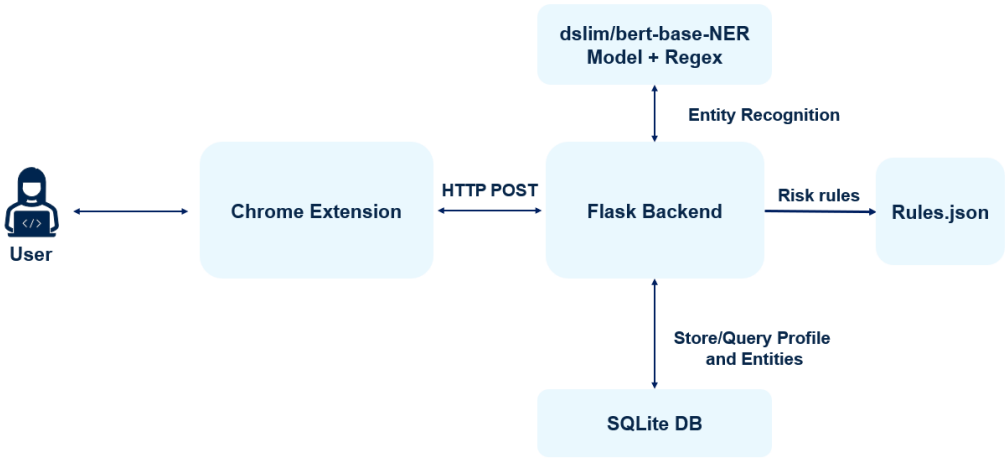

Figure 6.1: System Architecture and Implementation Overview

6.1 Tools and Technologies Used

This section will present the tools and technologies that have been used in this thesis to conduct the experiments and to implement the proposed system.

6.1.1 Facebook

Facebook represents one of most widely used social media when it comes to sharing personal information. The platform allows users to interact through public posts, comments or private messages.
Public posts allow users to share and communicate content that can be visible to a broad audience, that can also be beyond the immediate social circle. When a user shares personal information such as emails, date of birth or phone numbers, this data can be used by an attacker to perform identity theft or profiling.
Comments are used to interact with the contents that are shared by friends, pages or other figures and are linked to a specific post. The comments can also be used to reveal sensitive data.
Facebook also offers a more confidential space to allow direct communication with others through direct messages. Those spaces may remain subject to potential exposure when shared with external or untrusted recipients.

6.1.2 Google Calendar

Google Calendar is not a social network in the strict sense of the term, but it deploys social and collaborative features that can be still relevant in the context of information sharing. Google Calendar allows users to create, manage and share events, often including personal details such as event titles and descriptions, locations, invitation and participants.
Even if designed primarily as a tool for productivity, Google Calendar implies sharing data among private or semi-private communities. From a privacy perspective, this argues that data shared in such circumstances, even if not being publicly available, can still contribute towards building an enriched digital footprint and potentially expose individuals' safety if aggregated with data from other sources.

6.1.3 Other platforms

The extension and backend are designed to operate universally across web platforms. Beyond Facebook and Google Calendar, the system automatically detects and analyzes user interactions on other domains, classified under broader contextual categories.
LinkedIn, that was categorized as *professional_networking*, representing a *semi-public* context where personal identifiers and personal data can still expose users to profiling or doxing risks.
Ecommerce Platforms (Amazon, eBay, etc.), categorized as *e_commerce_checkout*, are considered as high risk environments for financial and identity data exposure, having more strict anonymization and risk rules.
ChatGPT was classified as a *generic_site* but of particular interest due to the frequent sharing of personal information during writing or email rephrasing activities. The system detects such patterns before the text is transmitted.
Evaluating the risks of sharing sensitive data on different platforms is important, since the levels of risk change depending on the domain and context in which it is evaluated.

6.1.4 Python

Python is a high-level general-purpose programming language widely used in context like Artificial Intelligence (AI), data science and web development. It consists of an ecosystem of libraries that can be used for **Natural Language Processing** (NLP) and **Machine Learning** (ML) tasks.
In this thesis, Python was used to implement the backend through a **Flask** framework. It manages the **Named Entity Recognition** (NER) pipeline using the **Transformers** library, executes the anonymization logic, applies contextual risk rules and communicates with the database.
Flask is used to expose the core analysis engine as a set of **RESTful API endpoints**. With use of the Flask server, the chrome extension communicates with the server through **HTTP POST** requests, sending data and receiving structured responses. The flask server acts as a bridge between the frontend and the backend.
A central role is played by the **Hugging Face Transformers** [14] library, which provides access to state of art pre-trained models. Specifically, the system integrates the **dslim/bert-base-NER** model which is a fine-tuned version of BERT. The model uses the **CoNLL-2003** (Conference of Natural Language Learning 2003) dataset for training. This dataset is a standard benchmark for Named Entity Recognition (NER) in the English language. The model is specialized in detecting standard entities such as **persons (PER)**, **organizations (ORG)** and **locations (LOC)**. Within the implementation, it is used with entity detection based on **regular expressions** (regex), making the risk analysis more robust.
Behind Hugging Face, the model is powered by **PyTorch**, one of the most used Deep Learning (DL) frameworks. PyTorch allows an efficient tensor computation, GPU support, and an agile architecture for Transformer models. Hugging Face leverages PyTorch to load, fine-tune, and run inference on huge language models with minimal setup.
The application of Python together with Hugging Face and PyTorch provides several benefits that make the pairing particularly adapt for the requirements of this work.
With its **readability and ease of use**, Python enables rapid prototyping, and it is feasible to develop and fine-tune the backend within a period which is brief. At the same time, Python has access to an extremely rich NLP library base including Transformers and PyTorch which can quickly be integrated with advanced ML models without needing to code complex algorithms directly.
Hugging Face also leverages this potential to the fullest with pre-trained, state-of-the-art models such as **dslim/bert-base-NER**. The use of such models eliminates the process of training across large datasets, which is a process that otherwise would be costly in terms of computation and time. In addition, taking advantage of BERT's bidirectionality, implemented efficiently in PyTorch, allows to improve entity detection's accuracy by enabling more disambiguation of words.
A second critical advantage of this approach is its **flexibility**: Hugging Face's modular structure makes it easier to include new models or fine-tuned versions, enabling the system to keep up with the most recent research and technology breakthroughs.
Practically, these characteristics result in a strong and efficient processing pipeline. Nicely formatted information such as email addresses, IBANs, or telephone numbers are recognized with precision by rules based on regex, while the Hugging Face

model recognizes entities dependent on the context that are difficult to identify. The usage of these two approaches enhances the credibility of the anonymization process and provides depth to the global efficiency of the risk assessment system.
The use of a Flask server allows to have more **simplicity and readability** making it easier to maintain the API routes. In addition, adding new endpoints is quite easy allowing a minimal effort addition to the system. It also allows seamless use of Hugging Face, Pytorch and SQLite in a single service.

6.1.5 JavaScript

JavaScript (JS) is a programming language deployed from the web, and it is essential for developing interactive and dynamic applications in the browser. During the implementation, JS was used to implement the **Chrome Extension**, which represents an important component to allow the user to interact with the system.
JavaScript is effective since it can communicate directly with the **Document Object Model** (DOM), such that the system can provide responses in real time as the user types. This implies that the extension can provide feedback instantly, such as warning symbols whenever a possible privacy risk is found, without requesting any additional effort from the user. Therefore, JavaScript allows for integration of backend intelligence and frontend user experience in order that privacy analysis can be made accessible and not obtrusive.

6.1.6 JSON Configuration Files

JSON (JavaScript Object Notation) is a lightweight, human-friendly data interchange format, and it is widely used to represent structured data. In the project, JSON files play a key role in externalizing the **risk rule definition**. The JSON files contain the mapping of entity types, contextual overrides, and correlation conditions that must be applied at the time of analysis.
The biggest advantage of this approach is **flexibility**. It allows to detach the rules from the code, in this way the system can be extended or modified just by modifying the JSON files, without the need to modify the Python logic. This architecture makes the risk engine very flexible and configurable: new types of risks, situations, or correlation schemes can be introduced linearly, and the system will always be easy to update and easy to maintain.

6.1.7 SQLite Database

SQLite is a lightweight relational database management system based on files that does not require a dedicated server. This system is ideal for local applications where it is important to prioritize performance and simplicity. SQLite has been used to store entities found during the analysis.
This is a persistent storage that allows escaping from single page analysis. By using an ongoing database of entities that are identified with hashed identifiers, the system can see correlations between unrelated websites over time. This allows to uncover high level threats, like profiling, identity theft, or doxing, which may not exist in a single interaction but become clear when a collection of unrelated data points is combined.

Finally, SQLite provides a safe and efficient foundation for correlation analysis that equates performance with privacy requirement to not keep data (that is regarded as sensitive) in plaintext.

6.2 Backend

In this section, the Flask backend is introduced, and its components are described. The backend serves as the analysis engine, it offers robust APIs for text anonymization, contextual risk assessment, and the maintenance of a persistent digital memory that enables the detection of cross-site correlations.

6.2.1 Database

The database is a key component of the backend, as it provides long-term memory for the analysis engine. A single-page analysis can underline the immediate risk, however, in this case, a persistent storage allows to detect more complex threats, such as repeated instances of the same entity appearing across different domains or dangerous combinations of attributes across different sessions. To achieve this, without compromising the privacy of the user, the schema is built around the concept of **profile**. A profile does not only represent a user account, but a secure container that groups entities belonging to the same individual.
This allows correlation between different websites and sessions. Profiles are anchored to strong identifiers, but instead of storing them directly, the system generates a hashed version. In this way, when the same anchor appears on different domains, the corresponding entities are associated with the same profile. This allows the system to infer cross-site privacy risks and detect threats that otherwise would remain invisible in an isolated analysis.

Implementation of the Database

The relational schema is defined by the **schema.sql** file, which specifies two primary tables: **profiles** and **entities**. These tables, together, enable permanent storage and correlation of sensitive information with the guarantee that raw personally identifiable information (PII) is never kept in unencrypted plaintext.
The **profiles table** serves as the backbone of this mechanism: rather than representing an actual user account, a profile acts as a secure container that groups together entities detected as belonging to the same individual.
Each profile is identified by a **profile_id**, which is not the original attribute but a hashed form of strong anchors such as an e-mail address, a telephone number, a fiscal code, or an IBAN. Additionally, the table holds an **anchor_preview**, a cut-down and non-sensitive form of the original identifier, that can be shown or utilized for debugging without revealing private data. A **created_at** timestamp saves the date and time when the profile was first created. This allows user profiles to potentially be identified and linked uniquely from session to session without ever having their sensitive attributes exposed publicly in their raw form.
The entities table saves every sensitive instance of data that was found in the text being analyzed. For each entity, its **entity_text, its entity_type** (such as EMAIL,

PHONE, PER, or LOC), the **source_domain** on which it was found, and a **created_at** timestamp are contained in the table. Each record has a corresponding matching *profile_id* in the profiles table, thereby allowing more than one entity to be referred to the same user profile. A unique constraint over the combination (profile_id, entity_text, entity_type, source_domain) prevents duplicate records from being created and maintains consistency and performance in the data store.

6.2.2 Flask Server

The flask server is the brain and heart of the system. It is implemented in **app2.py**, and it orchestrates every stage of the analysis pipeline going from the entity detection and anonymization to contextual risk evaluation and correlation across user profiles.
The backend, built with Python and Flask, is the intelligence core of the system. Its primary responsibilities include:

- **Data Processing**: Receiving text inputs and contextual metadata from the Chrome extension.
- **Sensitive Entity Recognition**: Identifying various types of sensitive information (e.g., names, locations, emails, financial identifiers) within the provided text.
- **Contextual Risk Assessment**: Applying a hierarchical set of rules to determine the privacy risk level of detected entities based on the context of their use.
- **Digital Profile Management**: Storing and managing detected entities in a secure SQLite database, building a long-term digital profile for the user.
- **Correlation Risk Inference**: Analyzing the aggregated profile data to detect complex, cross-contextual privacy risks that wouldn't be apparent from single-page analysis.
- **API**: Exposing a RESTful API for secure communication with the frontend.

Database Management

The database creation and connection management are achieved by two dedicated functions in app2.py. The **get_db()** function that establishes and caches SQLite connections within the Flask application context, ensuring the efficient reuse and proper closure of connections after each request. The **init_db_command()** function that initializes the schema by executing the SQL instructions defined in **schema.sql**, guaranteeing consistency and preventing schema drift. Together, these functions offer assurance that the database is consistent and that connections are managed efficiently throughout the whole lifecycle of the application.
At the core of the profiling process there is the **save_entities_with_profile()** function, which is responsible for associating recognized entities with persistent digital profiles securely. The function operates according to the following logic:

Algorithm 1: Secure Profiling Procedure (save_entities_with_profile)

Data: Detected entities E, source domain D
Result: Securely stored profile and related entities in the database

if *entity contains strong anchor* **then**
  normalize anchor (lowercase or digits only);
  $profile\_raw \leftarrow$ anchor;
end
else
  $profile\_raw \leftarrow$ temporary group identifier;
end

$profile\_id \leftarrow SHA256(salt + profile\_raw)$;

Insert $profile\_id$ and entities into database;

Commit changes to database;

The function first checks if there are any strong anchors (for instance, EMAIL, PHONE, CF, or IBAN). If such anchor is present, it is normalized (for instance, by converting emails to lowercase or retaining only numbers in phones) to ensure consistency. The anchor is then normalized and concatenated with a secret salt and hashed with the **SHA-256** algorithm to produce a stable though irreversible **profile_id**. This approach ensures that all the entities related to the same anchor are linked under the same hashed profile across different analyses, enabling the correlation in multiple domains.
In the absence of a strong anchor, a **fallback identifier** is generated by the system to temporarily group entities without exposing sensitive attributes. In this case, a temporary identifier is generated from the first detected entity and a contextual label, which is also hashed with the same salt. This allows the system to group related entities temporarily without exposing any sensitive information.
As a result, in the analysis report some profiles may appear as readable anchors, while others appear as hashed codes, corresponding to the fallback identifier.
Finally, entities found in the current analysis session are stored in the entities table under the corresponding profile_id. Because of the unique index of the database (seen in the last part of 6.2.1), repeated inserts are silently dropped, which ensures data integrity and efficient storage space.
Through this design combined with secure hashing and careful normalization, the system can construct long-term digital profiles and deduce cross-site correlations while maintaining a strong guarantee of privacy and confidentiality.

Entity Recognition

One of the central elements of the system's backend is the **find_sensitive_entities()** function, whose role is to extract potentially sensitive content from text generated by users.
The function uses a **multiple layered detection approach**, combining classic pattern matching (regex) with ML frameworks based on BERT based NER.
The architecture balances between precision and recall, minimizing false negatives

while avoiding spurious detections.
The first part of the procedure follows the logic illustrated in *Algorithm 2*.

Algorithm 2: Regex-based Entity Detection - part 1

Data: Input text T
Result: List of Regex-based entities E_{regex} and protected spans

Define Regex patterns for all sensitive data;

foreach *pattern in Regex map* **do**
  find all matches in T and append to E_{regex};
end

Mark coordinates of each match as protected zones;

Return E_{regex} and protected spans;

The first detection layer relies on Regex based detection, which applies constructed regular expressions to detect entities that follow rigid, carefully defined syntactic patterns. These include email addresses ([EMAIL]), phone numbers ([PHONE]), international bank account numbers ([IBAN]), Italian fiscal codes ([CF]), and dates ([DATE], [DATE_PARTIAL]).
Since these patterns are highly standardized, the entities that match regex rules are regarded as high confidence matches and are given high priority in subsequent processing.
The second part of the pseudocode, reported in *Algorithm 3*, expands the detection process with the use of AI-based entity recognition.

Algorithm 3: AI-based NER and Entity Filtering - part 2

Data: Input text T, protected spans from Regex layer
Result: Filtered AI-based entities E_{AI}

Run BERT-based NER model on T to obtain preliminary E_{AI};

foreach *entity in* E_{AI} **do**
  if *entity overlaps protected span or confidence* $<$ *threshold* **then**
    discard entity;
  end
  else
    retain entity;
  end
end

Transform T to Title Case (T') and re-run NER for missing PER entities;

Combine all valid AI entities and return E_{AI};

The second layer is based on AI-powered Named Entity Recognition (NER).
In this case, the system employs the **dslim/bert-base-NER** Transformer model, implemented in PyTorch through the Hugging Face framework.

The model extends the detection beyond strict patterns to identify context dependent entities like persons (PER), locations (LOC), and organizations (ORG).
Unlike regex, which only matches strings of characters, the Transformer-based model considers semantic context, allowing it to disambiguate homonyms (e.g., telling apart "Apple" as a company from a fruit).
Conflict resolution and filtering are another important innovation in this pipeline. Because regex and AI-based methods could overlap, the system features a filtering component to prevent redundancy and misclassification. For example, if a string such as mario.rossi@example.com is already matched by a regex as an [EMAIL], the model is forced to avoid marking "mario" or "rossi" as a PER entity.
Entity specific confidence requirements are also applied to AI outputs (e.g., 0.90 for PER, 0.85 for ORG, and 0.80 for LOC), ensuring that **only high confidence outputs are retained**. This step significantly reduces the number of false positives.
Finally, the role contains a case-insensitive scan for person entities. By capitalizing the input to the title case (for instance, from "mario rossi" to "Mario Rossi"), the model can recognize proper names which were entered in all lowercase. This step improves recall for person names, which are a very sensitive category of data in privacy analysis.
Through this multi-step and finely calibrated process, this function allows the deployment of accurate and reliable entity recognition.
It combines the traditional regular expressions with Transformer-based NER contextual understanding, employing conflict resolution and normalization steps to deliver results that are both accurate and understandable for downstream risk determination.
Once the entity extraction is completed, the detected entities are passed to the risk evaluation and anonymization module. Each entity is enriched with contextual information, such as the source domain and the specific user action (micro-context), before being evaluated through a hierarchical engine based on rules.
The results of this stage are then stored in the SQLite DB, associated with a secure profile identifier through the hashing procedure described in Section 6.2.2. This integration between entity detection, contextual risk assessment and persistent profiling enables the system to detect not only isolated privacy risks within a single text, but also correlation and combination of sensitive data in multiple domains.

Contextual Risk Assessment

The **anonymize_and_analyze()** method is the core of the system's privacy intelligence. It transforms raw entity detections into a risk assessment that is well structured and aware of the context.
Its design combines reasoning based on rules with environmental sensitivity, ensuring that the importance of each detected entity is evaluated in relation to the specific environment in which it is found.
In general, its purpose is not only to assign sensitivity scores to individual entities, but to interpret their *semantic meaning within a specific environment*, reflecting the principle of *privacy by context* defined in modern data protection frameworks.
This ensures that a piece of information such as name or phone number is not

evaluated in isolation, but rather in relation to *where* and *how* it appears, whether inside a private chat, a social post, or a form on an ecommerce website.
At the center of this mechanism lies an externally defined hierarchical rule application methodology, implemented in the **rules.json** file. This modular design allows risk logic to be expanded or updated without changing the core code and ensuring adaptability in the long term.
These rules are processed by the system in a tri-structured format, reflecting the increasing levels of contextual specificity:

- **Base Rules**: Establish a default risk level for all entity types. For instance, a phone number is always sensitive and always assigned a "High" level of risk regardless of context.
- **Category Overrides**: Adjust these default levels, saw previously, according to the context in which the text is being identified. As an example, while a phone number might be "High" risk by default, its appearance on a social_public site raises its severity to "Very High," considering the added exposure.
- **Micro-Context Overrides**: Provide the most granular level of control, tailoring risk to specific user actions such as creating a *social_public_post* or entering a *search_query*. These overrides take precedence over both base rules and rules dependent on the category and permit more specific judgments.

The general flow of the algorithm can be summarized as follows:

Algorithm 4: Contextual Risk Assessment and Anonymization

Data: Original text T, context category C, micro-context M
Result: Anonymized text and structured risk report

1. Detect entities in T using `find_sensitive_entities()`;
2. Apply hierarchical rule logic:
   - Base rules → assign default risk per entity type
   - Category overrides → adjust risk by environment
   - Micro-context overrides → refine to user action
3. Identify combination risks;
4. Evaluate keyword and pattern risks (e.g., "password", "home address");
5. Replace detected entities with category tags ([PER], [EMAIL]);
6. Compute overall risk level based on maximum individual severity;
7. Return anonymized text and structured entity report;

Beyond threats found on a single entity, the algorithm identifies **combination risks**, where presence in combination of more than one entity increases the potential for privacy exposure.
For instance, the combination of a person's name (PER) and the birthday date (DATE_BIRTHDAY) raises the risk of identity theft, while the occurrence of the name and a location (PER + LOC) increase the likelihood of stalking or doxing.

In addition, the system also supports **keyword** and **pattern recognition**, flagging inherently sensitive words such as "password" or "home address" as risky, and their presence is indicated whether accompanied by traditional entity types or not. Moreover, advanced regex patterns allow for the recognition of more structured identifiers. Those detections are evaluated under the same hierarchical framework, ensuring that their risk levels reflect the specific contextual conditions in which they appear.
Once the risk assessment has been completed, the method performs a **robust anonymization process**. This includes replacing all entities recognized with a standardized placeholder for its category (e.g., [PER], [EMAIL], [PHONE]). The anonymization is carefully done to ensure that there are no conflicting replacements, preserving the overall structure and readability of the original text and making sure sensitive elements are no longer directly revealed. Important is to avoid overlapping substitutions, guaranteeing semantic coherence and preventing text corruption.
Finally, the algorithm aggregates all the results into a unified report containing:

1. the **anonymized text**,
2. the list of **detected entities** with their assigned risk levels, and contextual justifications;
3. the computed **overall risk score** of the analyzed segment.

Each analysis result is then stored in the local DB and associated with a profile identifier. This enables the system to perform correlation analysis within different domains, detecting recurring entities or dangerous patterns across multiple domains and sessions.
Important is also the *anonymization phase*, which allows to replace sensitive entities with standardized placeholders, the system prevents any possibility of data leakage while maintaining the logical structure and interpretability of the original text.
This design ensures compliance with fundamental privacy principles such as *privacy by design* and *data minimization*.
Through the hierarchical and context aware framework, the function allows the system to evaluate situational meaning of each data point, identify risky data combinations, and apply privacy preserving transformations, archiving a precise balance between analytical accuracy and protection of sensitive information.

Correlation Risk Inference

The function **find_correlation_risks()** extends the analytical capabilities of the system beyond point-in-time analysis to address the long-term, cross domain character of privacy exposure. While earlier modules focus on the contextual risk associated with a single message or post, this component enables the system to reason over time, identifying correlations and recurring patterns that emerge from the aggregation of user data across different online interactions. This capability is relevant in the era of persistent digital traces, where apparently harmless fragments of information can be combined to reconstruct highly sensitive user profiles.
The system can infer risks that exist when information is aggregated over time and across domains by leveraging the permanent storage of entities in the SQLite

database. By analyzing this dataset, the function can infer privacy risks that stem form the *cumulative reuse or co-occurrence of data* across multiple sessions and domains.
The algorithm 5 highlights the logic of the correlation inference process, which identifies both cross-site profiling risks (same entity found on multiple domains) and entity combination risks (occurrence of different sensitive data types within a single user profile).

Algorithm 5: Correlation Risk Inference (find_correlation_risks)

Data: Entities and profiles stored in the SQLite database
Result: List of inferred cross-domain and combination risks

Initialize empty list R for detected risks;

if *correlation rules are loaded* **then**
  foreach *rule in correlation_rules* **do**
    if *rule.type == SAME_ENTITY* **then**
      Query database for entities with identical text and type;
      if *entity appears in* $\geq$ *min_domain_count distinct domains* **then**
        Add Cross-Site Profiling Risk to R;
      end
    end
    if *rule.type == ENTITY_COMBINATION* **then**
      Identify profiles containing all required entity types;
      For each profile, compute domain count and involved domains;
      Add Entity Combination Risk to R;
    end
  end
end

Return list R of all correlation risks;

A key category of detection handled by this function is the **Cross-Site Profiling Risk**. This risk applies when the same piece of information, i.e., an email address, phone number, or IBAN, is found on multiple and distinct websites. The function scans the database for entities that appear in at least two different domains, as defined by the following condition:

$$min\_domain\_count \geq 2 \tag{6.1}$$

The presence of the same identifier across various contexts is an indication of potential cross-site tracking and profiling, since malicious individuals could link those fragmented instances together to construct a more complete digital identity of an individual.
Another set of threats is identified by **Entity Combination Risks**, which arises when different types of entities can occur within the same digital profile, even if they were collected from separate sources or sessions.
As an example, the co-occurrence of a name of a person (PER) and a location (LOC) could be a threat of doxing or stalking, while the combination of a name

and a date of birth (DATE_BIRTHDAY) represent a typical indicator of identity theft.
The system is capable of **contextually upgrading** generic data entities (DATE or DATE_PARTIAL) to the more specific DATE_BIRTHDAY whenever contextual keywords such as "birthday" or "born on" are present during the analysis. Moreover, combinations of that nature can be identified even if the respective entities were not collected from the same webpage or during the same session, but rather across multiple interactions over time.
Importantly, the logic underlying both cross-site and entity combination risks is not hardcoded in the system. Instead, it is defined externally in the **correlation_rules.json** file.
This separation of the rule specification from the rest of the code allows the system to have a high degree of maintainability and flexibility. In addition, new types of correlations may be added, thresholds may be adjusted, and risk categories may be constrained without modifying the underlying Python implementation. As a result, the correlation engine can evolve dynamically to address emerging privacy threats.
Through this process, find_correlation_risks() serves as the system's "detective layer", combining separated pieces of sensitive data into higher level conclusions. This allows the platform to identify privacy threats that are invisible when analyzing a single post or message in isolation and hence giving a deeper understanding of the long-term risk associated with the sharing of digital data.
From a broader perspective, this component underlines the dual importance of persistence and protection; while long term data retention enables advanced risk reasoning it also introduces ethical challenges regarding the storage of personal traces.
By combining secure hashing, selective data retection and anonymized profiling, the system ensures that these analyses can be performed without exposing sensitive identifiers, demonstrating that meaningful privacy analytics can coexist with strong confidentiality guarantees.

API Endpoints and Security

The Flask backend offers a series of RESTful API endpoints through which the analysis engine communicates with the Chrome Extension.
The endpoints not only perform text analysis and database maintenance but also benefit from multiple layers of security to ensure the confidentiality and integrity of the system.
The first and most important endpoint is **/api/analyze** (HTTP POST), which serves as the **main entry point** for analyzing requests.
It receives text and contextual metadata (category, domain, and micro-context) from the extension and processes them through the anonymization and risk assessment pipeline. The output is returned in structured JSON format and includes the anonymized text, the detected entities with their respective risk levels, and any correlation risks derived from the persistent database.
This endpoint holds the system's core logic, and it is central to its integration with the browser extension.
The algorithm summarizes the workflow of the main analysis endpoint used to

process incoming requests from the Chrome Extension:

Algorithm 6: API Analysis Endpoint (/api/analyze)

Data: User text and contextual metadata (category, domain, micro-context)

Result: Structured JSON response containing anonymized text, detected entities, and correlation risks

1. Verify API key and JSON validity using decorators;
2. Extract text and contextual information from request body;
3. Call `anonymize_and_analyze()` to process text;
4. Save entities with `save_entities_with_profile()`;
5. Retrieve correlation risks with `find_correlation_risks()`;
6. Return combined analysis results as JSON response;

Moreover, the backend includes a secondary endpoint, **/api/reset_database_for_development** (HTTP POST), provided as a development and testing convenience utility. This resets and rebuilds the database to a default state. To prevent misuse, this action is protected by a special **RESET_API_KEY** so that it can only be invoked by approved developers or test environments.

Algorithm 7: Database Reset Endpoint

Data: HTTP POST request with header `X-Reset-Key`

Result: Database reinitialized to default state

1. Verify that request header contains valid `RESET_API_KEY`;
2. **if** *key is invalid* **then**
   return HTTP 401 Unauthorized response;
   end
3. Execute `init_db_command()` to recreate database tables;
4. Return JSON response confirming successful reset;

To secure access, all API endpoints are protected by a custom API key authentication mechanism. The **@require_api_key** decorator checks for the presence of a valid X-API-Key within the request headers and rejects unauthenticated requests. This stops unauthorized clients from accessing the analysis service.

Algorithm 8: API Key Authentication Decorator (@require_api_key)

Data: HTTP request headers

Result: Access granted or denied based on API key validity

1. Extract `X-API-Key` from request headers;
2. **if** *API key is missing or invalid* **then**
   log unauthorized attempt and abort with HTTP 401;
   end
3. Allow request to proceed to the wrapped function;

This is accompanied by the **@require_json_and_text** decorator, which checks requests by ensuring incoming requests are in JSON and possess the anticipated text field. The structure, thus, places early error checking and reduces the likelihood of malformed or malicious inputs.

Algorithm 9: Request Validation Decorator (@require_json_and_text)

Data: HTTP request with potential JSON body

Result: Validation of request format and required fields

1. Verify that request body is in JSON format;
2. **if** *request is not JSON* **then**
   log warning and abort with HTTP 400;
   end
3. Check that JSON includes non-empty `text` field;
4. **if** *`text` field missing* **then**
   log warning and abort with HTTP 400;
   end
5. Log valid request and allow function execution;

Since the Chrome Extension will be executed inside the browser but, at the same time, communicates with a local backend, the system uses CORS (Cross-Origin Resource Sharing) configuration provided by Flask CORS.
This will intentionally enable requests originating from the extension, enabling secure communication while blocking any unauthorized cross-origin access from external sites.
Finally, the backend has **auditing and transparency logging**. A dedicated **api_logger** logs every API request, including successful analysis or unauthorized attempts to access. The logs are managed in a rotating file system, enabling complete inspection during debugging as well as facilitating accountability for security monitoring. The file .log is not present on GitHub for security purposes, but it will be automatically created after the first request.
With all these combinations of bounded endpoints, authenticated layers, request validation, cross-origin configuration, and fine-grained logging, the API is not only

functioning but also strong, secure, and resilient interface between the Chrome Extension and the backend analysis engine.

6.3 Chrome Extension

The **Chrome Extension** has been designed to be integrated into the user's browsing experience, as it provides **real-time monitoring** of the text given in input by the user and it is used to deliver a contextual privacy risk analysis through an intuitive report. Its architecture is defined and orchestrated by a set of configuration and script files, each of which plays a specialized role in enabling this interaction. Its functions include:

- **Real-time Monitoring**: Intercepting user input in text fields and content editable elements on webpages.
- **Micro-Context Detection**: Inferring the specific user action and context based on DOM elements and attributes.
- **User Feedback**: Providing immediate visual feedback with the use of warning icons on the potential exposure of sensitive information and privacy risks.
- **Detailed Reporting**: Presenting a unified privacy risk report in a popup interface.
- **Communication**: Acting as the intermediary, sending user input and contextual information to the flask backend and displaying the analysis results.

6.3.1 Extension Configuration and Permissions

The **manifest.json** is the heart of the extension, it defines the structure, permissions and execution logic. The manifest informs the Chrome browser of the capabilities of the extension and registers the components that make up its functionality.
It follows the **Chrome Extension Manifest V3 specification**, which ensure higher security and performance. This standard introduces explicit permission management and a clear separation between background, content, and user interface scripts, making the extension more efficient and easier to maintain.
The manifest first declares the permissions required for operation. These includes:

- **contextMenus**, which allows to the extension to add a custom right-click option, allowing the users to manually trigger a privacy risk analysis on selected text.
- **storage**, which permits the extension to cache the results of the previous analyses and make them accessible in a popup report.
- **activeTab**, which enables permissions to the extension to access to the URL and the title of the currently active page in which the search is being conducted. This ensures that the analysis results can be contextualized with the domain.

- **host_permissions** set to **all_urls**, which authorizes the extension's content script to operate on any visited webpage. This allows the system to be universally applicable across platforms and services.

```
{
  "manifest_version": 3,
  "name": "Privacy Risk Analyzer",
  "version": "1.1",
  "description": "Automatically analyzes text as you type to identify
     sensitive data.",
  "permissions": [
    "contextMenus",
    "storage",
    "activeTab"
  ],
  "host_permissions": ["<all_urls>"],
  ....
}
```

The manifest specifies the **background script**, saving **background.js** as a **service worker**. This design makes the background logic persistent, enabling the extension to handle events such as API calls to the Flask backend, context menu interactions, and communication with other extension components.
The content script, defined as **content_script.js**, is configured to run with the document_idle setting on all URLs. This ensures that it loads only after the main page content has finished rendering, preventing interference with page performance while still allowing real-time observation of user input fields.
Finally, the manifest configures the action associated with the extension's toolbar icon. The default action is linked to popup.html, ensuring that clicking the extension icon opens the reporting dashboard, where users can review detailed results of the most recent analysis. Finally, all these mechanisms allow the system to manage the extension's behavior without disrupting the user's browsing experience.

6.3.2 Background Communication and Event Handling

The **background.js** file is the **communications hub** of the Chrome extension. This hub allows mediating between the user facing components (context script and popup which are discussed in next sections) and the flask backend (discussed in 6.2). This module manages event handling, API calls and state caching, ensuring that the extension behaves in a reliable way across the browsing sessions.
The principal function is **analyzeTextWithApi()**, which is an asynchronous wrapper around the fetch call to the Flask backend's **/api/analyze** endpoint (seen in 6.2.2). This function is responsible for serializing the analyzed text and context into JSON, appending the required API key in the request header, and handling any errors that may arise during communication. By doing so, it encapsulates all backend communication logic, to ensure both security and robustness.
To provide **contextual awareness**, the function **determineContextFromUrl()** parses the current page's URL and maps it to a high level category. For instance, domains such as *facebook.com* or *twitter.com* are classified as social_public, while

amazon.com is categorized as e_commerce_checkout. This categorization is essential because the same piece of information may present very different levels of risk depending on the website type.
The **message listener**, registered with **chrome.runtime.onMessage.addListener**, serves as the main channel of communication with the content script. When an analyzeText message is received, the listener extracts the submitted text, the current URL, and, crucially, the micro-context provided by the content script (for example, distinguishing between a social_public_post and a social_public_comment). A context object is then constructed, combining the category, domain, and micro-context. This object is sent to the Flask backend via **analyzeTextWithApi()**.

Algorithm 10: Message Handling Logic in background.js

Data: Messages from content script (type, text, micro-context)

Result: Analysis results forwarded to popup and UI updates in the webpage

1. Listen for messages via `chrome.runtime.onMessage.addListener`;
2. **if** *message.type == "analyzeText"* **then**
   Extract text and current URL;
   Determine context category from URL;
   Build context object (category, domain, micro-context);
   Call `analyzeTextWithApi()` with text and context;
   Save results in `chrome.storage.local`;
   Send response back to content script;
   end
3. **if** *message.type == "openPopup"* **then**
   Open reporting popup and clear badge notifications;
   end

Upon receiving the backend's response, the results are stored in **chrome.storage.local**, making them accessible to the popup interface.
At the same time, a sendResponse is returned to the content script, enabling it to update the user interface, typically by displaying or removing a warning icon next to the text field based on the detected risk level.
Finally, the background script handles popup management. It listens for openPopup messages from the content script to open the reporting dashboard when the user clicks on a warning icon. Additionally, it clears the extension's badge text whenever the popup is opened, ensuring that notifications remain consistent and unobtrusive.

6.3.3 Real Time Input Monitoring and Context Detection

The **content_script.js** is a **field level agent** of the Chrome extension, representing the direct interface of the system with the user's browsing activity.
Unlike the backend components that operate on aggregated data, this file operates directly with the **Document Object Model (DOM)** of any visited page. Its

purpose is to observe user interactions in real time, detect contextual scenarios from the interface and trigger the privacy risk report when sensitive information is entered by the user.
This approach translates the principle of *privacy by design* into an operational form: instead of performing post-hoc analysis, the system anticipates privacy violations before data is ever transmitted to a remote server.
The core element is the **event listener** that is linked to document.addEventListener ('keyup'). This listener continuously monitors user input across the entire page, intercepting keystrokes in editable fields. To correctly associate the input with its source, the utility function **findEditableContainer()** traverses the DOM from the event target upward and locates the actual editable element. It can handle a variety of field types, including standard $< textarea >$ and $< input >$ elements as well as modern [contenteditable="true"] containers, which are widely used in social networking sites.
Algorithm 11 illustrates the event handling logic that detects the editable fields, infers their contextual meaning and triggers a privacy analysis when the user enters potentially sensitive data.

Algorithm 11: Event Handling and Micro-Context Detection

Data: User keystrokes within editable DOM elements

Result: Micro-context inference and triggered privacy analysis

1. Listen for `keyup` events across the document;
2. Identify the editable container using `findEditableContainer()`;
3. Infer micro-context based on element attributes:
   - "What's on your mind?", "Scrivi un post" → `social_public_post`
   - "comment", "scrivi un commento" → `social_public_comment`
   - "message", "messaggio" → `social_private_message`
   - "search", "cerca" → `search_query`
4. Assign a unique ID to the editable container if missing;
5. Call debounced analysis function with text and inferred micro-context;

To allow balancing the responsiveness with performance, the system deploys a **debouncing strategy**. The core analysis routine, **performAnalysis()**, is wrapped inside a debounced function that introduces a delay of 1500 milliseconds. This ensures that the backend is only contacted once the user has paused typing, to prevent unnecessary API calls during rapid input. This strategy not only optimizes the user experience but also reduces computational and network load on the Flask server.
The algorithm 12 summarizes the debounced execution routine that limits API calls and provide near real time feedback on detected privacy risks.

Algorithm 12: Debounced Analysis Execution

Data: Editable DOM element E, detected micro-context M

Result: Efficient backend analysis request and real-time risk feedback

1. Initialize debounced function with delay of 1500 ms;
2. Extract text content from E (`value` or `innerText`);
3. **if** *text is empty or shorter than the minimum threshold* **then**
   remove any existing warning icon and terminate execution;
   end
4. Send message to the background script containing:
   - `type` = "analyzeText"
   - `text` = extracted user input
   - `microContext` = M
5. Wait for response from backend analysis;
6. **if** *response indicates risk detected* **then**
   display warning icon next to the input field;
   end
   else
   remove any existing warning icon;
   end

A distinctive innovation of this script is its **micro-context detection** capability. The script infers the user's specific intent with semantic hints in the placeholder and aria-label attributes of the input fields.
This part comprises Italian and English typical placeholders for social networks. For instance, a placeholder such as "What's on your mind?" or "Scrivi un post" signals that the user is composing a public post (social_public_post), whereas phrases like "comment" or "scrivi un commento" in Italian indicate a public comment (social_public_comment). Similarly, attributes referencing "message" or "messaggio" imply a private message (social_private_message). This enables the creation of highly specific risk rules that distinguish between different modes of sharing on various platforms.
Once the analysis is performed, the script provides immediate feedback through a dynamic user interface. The functions **showWarningIcon()** and **removeWarningIcon()** manage the injection and removal of a warning icon positioned next to the relevant input field. For instance, when the user is typing sensitive information the warning icon will appear and clicking on the icon will reveal a report that contains the sensitive information and its related risk.
Algorithm 13 resumes the feedback mechanisms that visualizes privacy risks and allows the user to open a detailed report interactively.

Algorithm 13: Dynamic Risk Feedback and User Interaction

Data: Editable container E, computed risk level R

Result: Visual feedback via dynamic warning icon

1. Create or update warning icon linked to E;
2. Set icon color:
   - red → `High` or `Very High`
   - orange → `Medium`
   - grey → `Low`
3. Set tooltip text describing detected risk;
4. Add click event listener:
   - Send "openPopup" message to background script
   - Display full report in popup interface
5. **if** *risk no longer detected* **then**
   remove associated warning icon;

end

The icon serves as a **real-time alert** and its color reflect the different level of risks. Clicking on the icon would send a message to background.js, which in turn opens the report in the popup interface.
Finally, context_script.js not only detects sensitive information as the user types but also interprets the specific context of sharing, allowing for an accurate privacy risk assessment.

6.3.4 User Interface for Reporting

The reporting interface of the Chrome extension is implemented through the combined use of **popup.html** and **popup.js**, which together form the **user-facing dashboard**.
This component transforms the raw analysis output from the backend into a clear, structured, and intuitive report. This ensures that users can easily interpret privacy risks without requiring any technical expertise, translating quantitative and contextual data into actionable insights.

Risk Reporting Interface

The file **popup.html** defines the **visual layout** of the report, it is organized into modular sections.
At the top there is the card that summarizes the **overall risk level** that is contextualized by the domain in which the information is typed and the microContext in which it was analyzed.
This part is followed by a dedicated section displaying the **anonymized version of the text**, which allows understanding a possible way to mask the identities while preserving the readability of the content. All the detected sensitive entities are replaced by their corresponding placeholders (e.g., [PER], [EMAIL], [PHONE]).

This anonymized representation allows users to see how their content could be shared safely, preserving the structure and meaning of the text and, at the same time, removing personally identifiable information (PII).
Then, a detailed **Risk Details** table summarizes each detected entity, specifying the type, risk level and explanation of the possible threats, specifying:

- **Type**: the category of the identified information;
- **Level**: the assessed severity based on the contextual rules in JSON files (such as Very High, High);
- **Suggestion**: an explanation and recommended mitigation action.

In the figure, it is possible to observe an example of this report. After inserting an input text on Facebook, the tool finds the sensitive entities and shows the report, highlighting all the identified data.

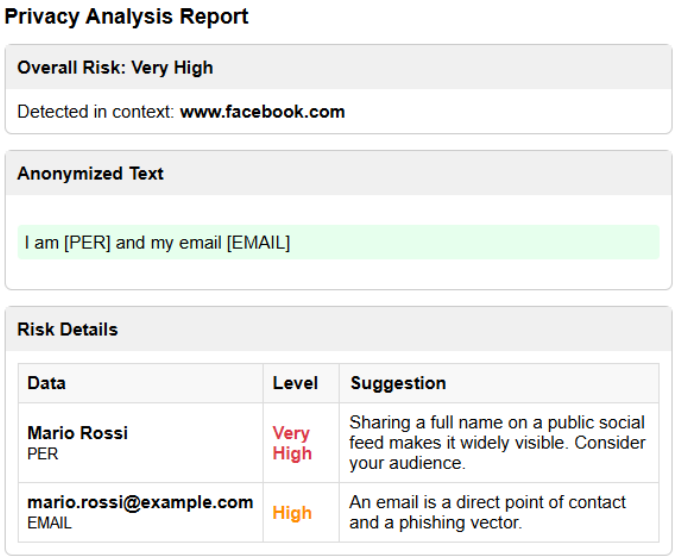

Figure 6.2: Risk Report Interface (Single-Site Analysis)

The first part focuses on the analysis on a **single site**, allowing the user to understand the sensitivity of the information shared within the current platform.
However, the system extends its analysis across different domains to underline the risks that emerge when data is reused or correlated over time on different domains.

Cross-Site and Correlation Risks

The interface includes two specialized sections, **Cross-Site Correlation Risks** and **Profile Combination Risks**, which visualize long term and multiple domain privacy threats (Figure 6.3). The **Cross-Site Correlation Risks** card displays the cases in which the same entity (for example, email address or phone number) was detected across multiple distinct websites. Such findings suggest potential for cross site profiling or identity tracking, since malicious parties could link this information to reconstruct a broader digital identity of a user.
On the other hand, the **Profile Combination Risks** highlights dangerous correlations that exists within the same digital profile, even if gathered from separate

sessions or websites.
Examples include:

- The co-occurence of a person's name and a location (PER + LOC), which could be enable the possibility for a malicious actor to detect the location of a victim.
- The combination of name and date of birth, which could be associated with identity theft.
- The pairing of name and organization could reveal information about the workplace of the victim.

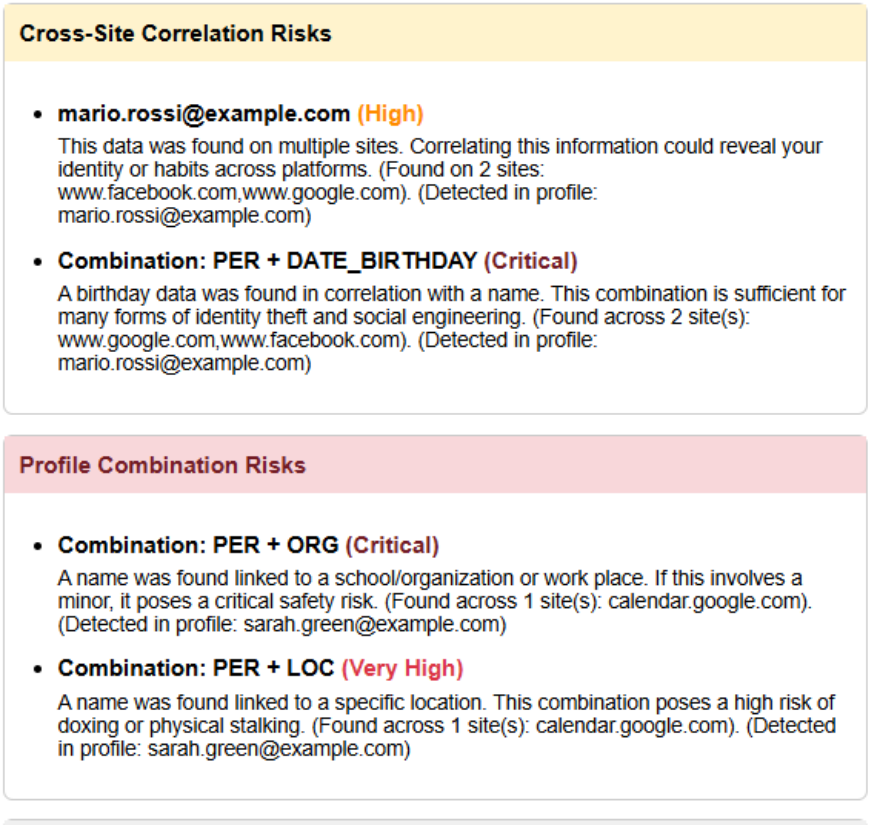

Figure 6.3: Risk Report Interface (Cross-Site Correlation Analysis)

Each entry probides detailed context, listing where the entity was found, how many distinct domains were involved, and what type of privacy implications it carries.

Visualization and Behavior Logic

The **popup.js** script provides the logic for dynamic rendering. It begins by retrieving the lastAnalysis object from chrome.storage.local, which contains the most recent analysis results computed by the backend. The script then populates the placeholders defined in popup.html with actual data, ensuring that the report is updated seamlessly whenever a new analysis is performed.
The key feature is the **Conditional styling logic** that is implemented by the function **getRiskColor()**. This function applies color coding depending on the

risk identified. This allows the system to give less cognitive effort to interpret the result.
Moreover, the script distinguishes between two major classes of risks. **Cross-Site Correlation Risks** are presented when the same entity is detected across multiple distinct domains and signals the potential for tracking and profiling. **Profile Combination Risks**, on the other hand, arise when sensitive combinations of entities (e.g., name and date of birth, or name and location) are associated within the same user profile, regardless of whether they were extracted from the same site or at different times. These categories are represented separately, allowing the report to provide a granular view of threats, differentiating between risks that come from an exposure across sites and those arising from depth of information within a single profile.
The integration between popup.js and popup.html ensures the analysis is communicable and actionable. The user interface and report allows the system to translate the technical findings into something accessible by everybody. This allows the user to make informed decisions when sharing their personal data in real time.

Chapter 7

Results and Performance Evaluation

7.1 Experimental Setup

This section describes the configuration and methodology used to evaluate the performance and accuracy of the proposed monitoring system for privacy awareness. The experiments were conducted with the goal of assessing the functional correctness of the detection and the computational efficiency of the overall architecture. During the experimental phase two main aspects were considered:

1. The **execution environment** and **testing tools** used for reproducibility.
2. The **evaluation metrics** applied to measure the effectiveness of the system.

7.1.1 Test Environment and Tools

The experiments were performed on a mid-range workstation configured as follows:

Table 7.1: Hardware and Software Configuration of the Test Environment

Component	**Specification**
Operating System	Ubuntu 22.04 LTS (64-bit)
Processor	Intel® Core™ i7-10750H @ 2.60GHz (6 cores, 12 threads)
Memory	16 GB DDR4 RAM
Storage	512 GB SSD
Browser	Google Chrome v126 (Developer Mode enabled for extension testing)
Backend Environment	Python 3.10 with Flask v3.0.3
Database	SQLite 3.45 (schema defined in `schema.sql`)
AI Model	`dslim/bert-base-NER` (Hugging Face Transformers + PyTorch)

The backend was executed locally, while the Chrome extension was installed in **Developer Mode** to allow direct inspection of the background messages and network requests.
This setup ensured a communication with low latency between the frontend and backend, allowing privacy analysis in real time and risk visualization.
To validate the system's robustness, tests were conducted on multiple types of content such as social media posts, cross-site scenarios, and multiple contexts.

7.1.2 Evaluation Metrics

The quantitative evaluation of the **entity recognition** component relied on the same set of metrics introduced in 3.1.2, which are **Precision**, **Recall** and **F1-Score**.
Those measures are widely used in Machine Learning (ML) to quantify both the accuracy and the completeness of detection models. In the context of this work:

- **Precision** evaluates how many of the detected entities were actually correct, thus allowing measuring the reliability of the output of the system.
- **Recall** measures the ability of the system to identify all sensitive entities present in the text, assessing its coverage.
- **F1-Score**: provides an indicator of the overall performance of the system, balancing Precision and Recall through the harmonic mean.

These metrics were computed by comparing the system's detections against a manually annotated *ground truth* dataset containing a variety of entity types.
The evaluation focused on balancing two major objectives:

1. **Minimizing false positives**, to prevent unnecessary alerts; and
2. **Minimizing false negatives**, to ensure no sensitive data remains undetected.

In addition to accuracy metrics, **performance indicators** such as **execution time** and **memory consumption** were also recorded for each analysis session.
These parameters were selected to evaluate the **computational efficiency** of the system and its ability to operate effectively in interactive browsing scenarios. The **Execution Time** measures the latency between the moment the text is submitted for the analysis and the moment the complete report is returned by the backend. In other words, it measures the **overall temporal efficiency** of the system, meaning *how long the backend takes to produce complete results.* This value is calculated as the time difference (in *milliseconds (ms)*) between the start and the end of the request, including all the intermediate stages, like entity recognition (with regex and BERT), risk evaluation, response serialization and database access.
The **Memory Consumption** represents the variation in memory allocation during the execution of each analysis task. In other words, it measures the **efficiency in terms of resource usage**, that is, *how much memory the model and the backend occupy in RAM during the execution.*
This was monitored using the **psutil** library in Python, in order to estimate the memory footprint of the NLP model and the overall backend process. This measure

was monitored because the system relies on a pre-trained Transformer model, which requires significant RAM for loading and inference.
This metric is derived from the difference in *resident memory* (RSS) allocated to the backend process before and after the execution, expressed in *megabytes (MB)*.
Table 7.2 summarizes the performance metrics considered in this evaluation:

Table 7.2: Performance Metrics Adopted for System Evaluation

Metric	What it Measures	Why it is Important
Execution Time (ms)	Total latency of the analysis process	Evaluates the system's responsiveness and overall user experience.
Memory Change (MB)	Additional RAM usage during model inference and data processing	Assesses the system's scalability and resource efficiency.

These parameters were collected through a custom decorator, which wraps each endpoint function. Before and after the request, the decorator retrieves the process statistics using the psutil library and computes the total elapsed time and memory variation.
This approach is lightweight and allows evaluating the efficiency under real usage conditions. Together, these quantitative and performance metrics provide a complete assessment of both the effectiveness and technical robustness of the system.
The data collected through this mechanism are later used in Section 7.3 to evaluate the overall responsiveness and memory usage of the system in different scenarios.

7.2 Functional Evaluation

This section presents the functional evaluation of the developed system, focusing on its ability to identify, classify, and anonymize sensitive entities in diverse contexts. Unlike the quantitative assessment provided by the performance metrics, this part focuses on the *qualitative behavior* and how the system components interact to deliver accurate privacy assessments.
The evaluation covers three main aspects:

1. **Entity Detection Accuracy**, validating the use of approaches based on regex and AI;

2. **Anonymization Effectiveness**, assessing whether sensitive data is masked while preserving its readability;

3. **Contextual and Correlation Risk Assessment**, checking their accuracy on reasoning based on rules and profiling.

7.2.1 Entity Detection Accuracy

The entity recognition mechanism combines **Regular Expressions** (regex) with **Named Entity Recognition** (NER) using *dslim/bert-base-NER* model from *Hugging Face*.
This strategy is introduced in Section 6.2.2 and was designed to leverage the complementary strengths of both approaches: the *deterministic precision* of regular expressions for structured data, and the *semantic flexibility* of the Transformer based models for the entities that depend on the context.
This approach was validated using different test scenarios designed to cover different entities, text structures and risk levels.
The validation included test samples with different linguistic structures, domains and sensitivity levels, allowing verification of how the system behaves across different input conditions.
In particular, the experiments were structured to validate the detection accuracy under three main scenarios:

- **Detection using regular expressions**: targeting entities with clear and standardized formats such as phone numbers, fiscal codes, IBANs and emails. Those entities are expected to be identified with high confidence due to their rigid syntactic structure.

- **Detections using the NER model**: focusing on unstructured and contextual entities such as names, organizations or locations. This category tests if the model has the ability to understand the linguistic context and correctly classify entities that cannot be captured with regex only.

- **Conflict Resolution Logic**: which prevents overlapping detections, such as avoiding tagging "mario" and "rossi" as a [PER] inside an email like "mario.rossi@example.com". This allows the system to guarantee semantic consistency and prevents false positives (FP) in multiple layered analysis.

Test 1: Entity based detections

The first test validates the ability of the system to detect highly structured and unambiguous entities through regular expressions. The text given in input was:

> "Contact me at andrea.bianchi@uniroma.it or +39 340 1122334. My IBAN is IT60X0542811101000000123456. My fiscal code is RSSMRA85T10A562S"

This text shows multiple sensitive and personally identifiable entities such as IBAN, fiscal code and phone number.
Those identifiers are appear in publicly accessible content (*facebook.com*), thereby representing a realistic high exposure privacy scenario.
The results of the analysis are reported in Figure 7.1, all entities were correctly identified and associated with specific risk level that reflects the intrinsic sensitivity and the contextual exposure of the data.

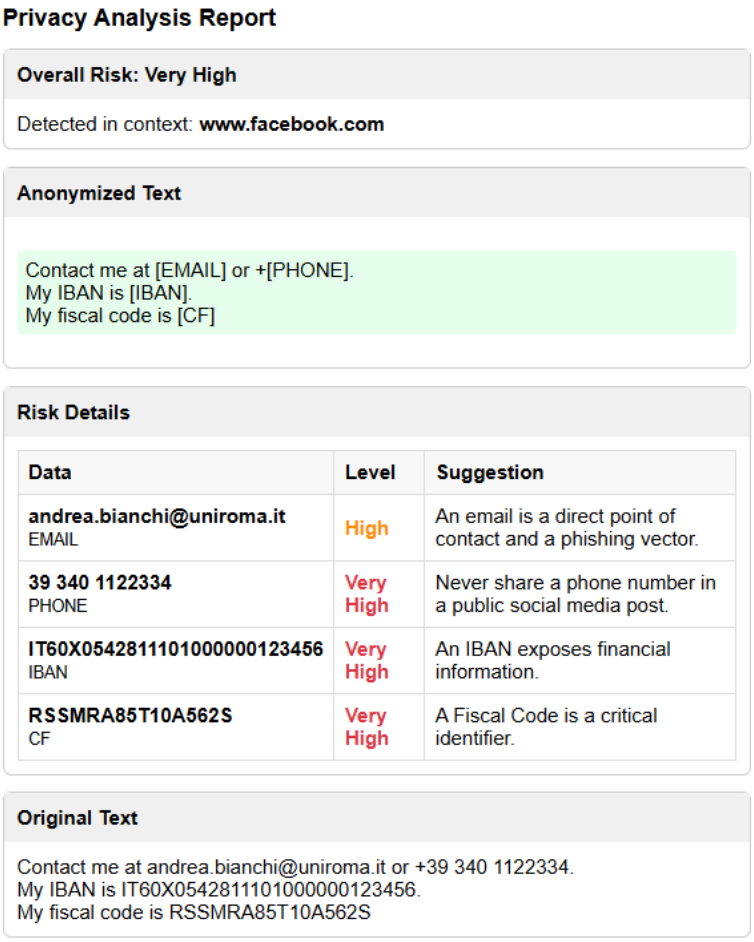

Figure 7.1: Risk Report Interface (Entity Recognition)

This report confirms the **reliability of the regex based detection layer**, which ensures high Precision for structured patterns. All the entities have been classified with *High* and *Very High* levels of risks, which reflects the potential damage that the disclosure of such information could cause.
When such identifiers are made publicly accessible, they can be exploited by malicious actors in a variety of ways:

- **Email Address**: Publicly exposed email address are one of the most common entry points for **phishing campaigns** and **spam attacks**. Attackers can impersonate trusted recipients to extract sensitive information or credentials from the victim. Moreover, the use of an institutional email (such as *uniroma.it*) may reveal professional affiliation, enabling targeted attacks.
- **Phone Number**: personal phone numbers can be exposed to **social engineering**, **smishing (SMS phishing)** or unauthorized enrollment in online services. Combined with the other identifiers, they facilitate **entity correlation** across platforms, allowing the adversaries to link multiple information about an individual.
- **IBAN**: Although IBAN alone does not grant direct financial access, it can be exploited in **bank fraud schemes** or **invoice manipulation attacks**.

For instance, a threat actor could fabricate fake invoices using a legitimate looking IBAN to trick victims into transferring funds to fraudulent accounts.

- **Fiscal Code**: National identifiers can be leveraged for **identity theft** or **document forgery**. With sufficient personal information, attackers could request financial products, loans, or even simulate an individual's identity in bureaucratic procedures.

Test 2: Named Entity Recognition

The second test focused on contextual entities requiring semantic understanding by the BERT-based NER model. The evaluated text was:

> "Hello, my name is Sarah Green. I live in Boston and work for NovaTech Corporation. Please email me at sarah.green@example.com or call me at +39 333 1234567 or 555-123-4567."

As shown in Figure 7.2, contextual entities such as Sarah Green, NovaTech and Boston were successfully recognized and identified as [PER], [ORG] and [LOC].

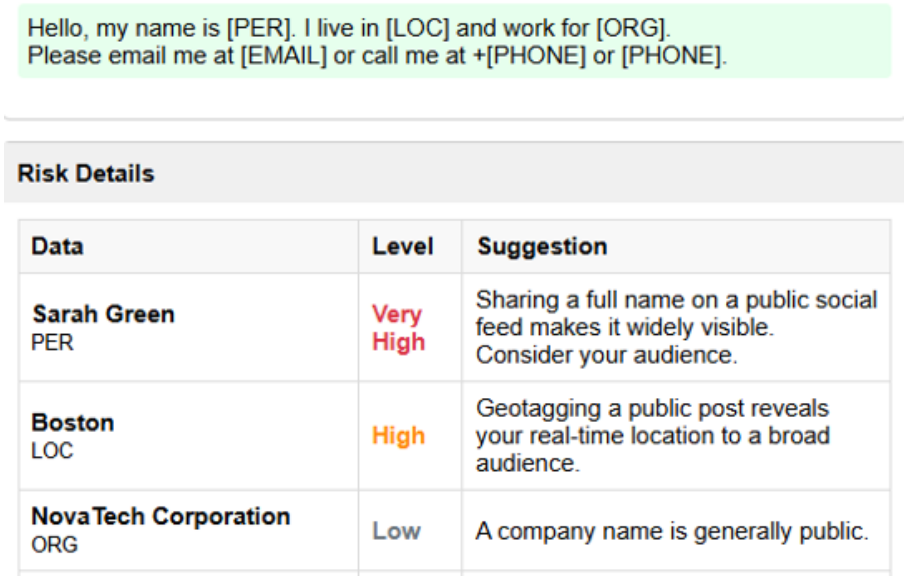

Figure 7.2: Risk Report Interface (Entity Recognition with NER)

The other entities are identified with the recognition through regular expressions, however those identifiers are not reported in the figure because those elements were not the focus of the test.
This report confirms the accuracy of the **Transformer Model** and its integration into a hybrid pipeline. It is important to note that the risk associated with each entity individually differs from the risk when they are considered in combination. For this reason, ORG is regarded as *low risk* in this scenario. The combination between those information is explained in Section 7.2.3, which underlines how those information in combination increases the potential for privacy violations.

Test 3: Conflict Resolution and Overlap Filtering

The third test evaluated the conflict resolution logic that merges or filters overlapping detection between the regex and layers implemented with AI. The sentence analyzed was:

"I am Mario Rossi and my email mario.rossi@example.com"

The conflict resolution rule prevents the model from detecting "Mario" or "Rossi" separately as additional person entities within the already detected email address. Without this resolution, the model would identify the identifiers as [PER] and [EMAIL] at the same time. The Figure 7.3 depicts the system correctly identifying the person name and the email without redundant tagging.

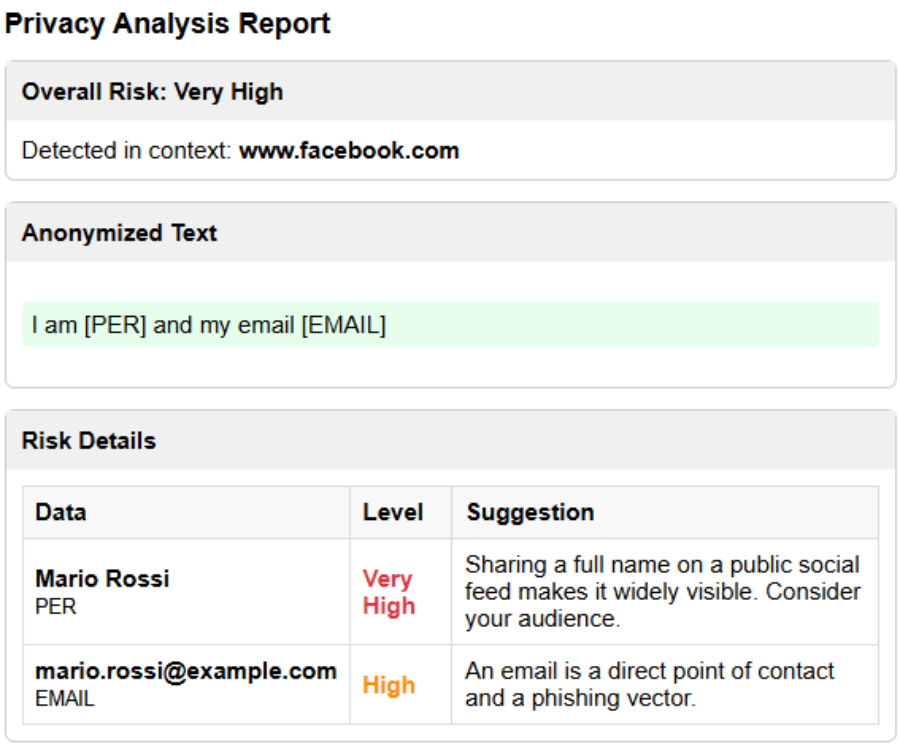

Figure 7.3: Risk Report Interface (Entity Recognition without conflicts)

This behavior confirms the **priority and without overlap system** between the regex and NER layers operates as intended. The regex detection dominates for structured entities, while the AI detection fills in contextual gaps and the overlapping spans are automatically filtered.

Results

These qualitative results demonstrate that: the **regex based** detection correctly identifies the sensitive elements in structured data, the **BERT based NER Model** ensures contextual understanding and adaptability, and the **conflict resolution** maintains consistency between both layers.
Overall, this analysis confirms that the architecture performs robustly in both simple and complex linguistic contexts, delivering accurate entity recognition across

different input texts.

7.2.2 Anonymization Effectiveness

The anonymization component, that was explained in the previous chapter in Section 6.2.2, was tested to verify its ability to replace detected entities with standardized placeholder (such as [PER], [LOC], [PHONE]) while preserving the semantic meaning of the original message.

Test with ChatGPT

The anonymization module of the report could be applied outside the scope of traditional browsing.
The anonymization part can be used in websites, such as *chatgpt.com*, to generate an anonymized version of an email before it is shared in the platform. This ensures that **sensitive personal data is not transmitted or stored in external servers**, protecting the privacy of the individual and at the same time, preserving the meaning and context of the message.
This idea comes from the fact that tools like *ChatGPT* are widely used to improve the tone and the style of emails. Users are unaware that their personal information could be processed or stored during this interaction.
By anonymizing the content before submission, the system prevents the exposure of privacy or identifying details, making the editing process both safer and that preserves privacy.
An example could be sending an email containing sensitive data like the one shown below:

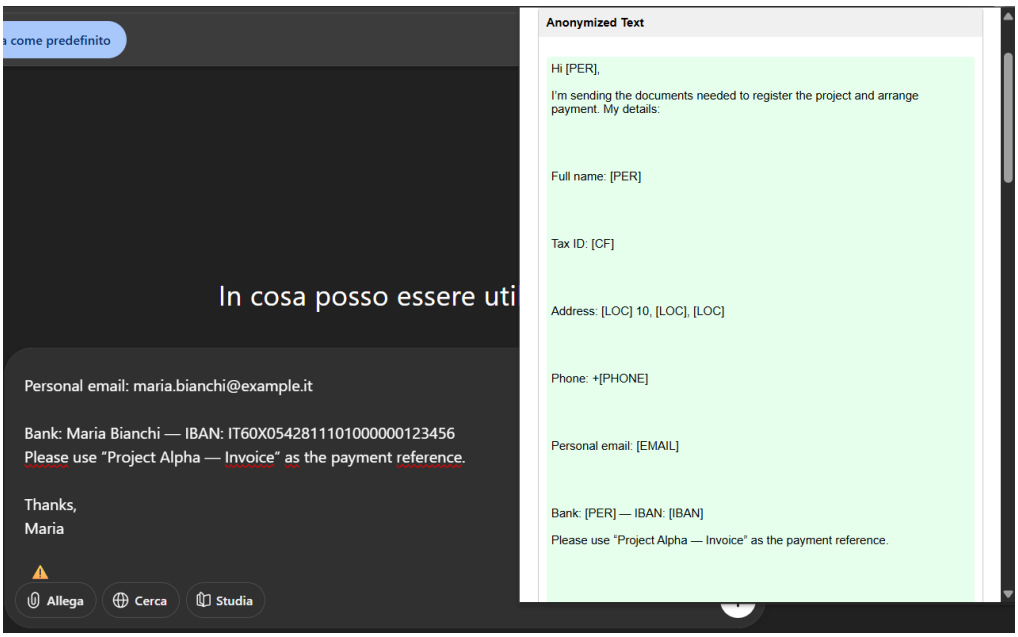

Figure 7.4: Risk Report Interface (AI Data Sharing Risks)

In this scenario, all the data that is found as sensitive is anonymized.
By automatically detecting and masking sensitive data before the text is shared

online, the system minimizes the risk of unintended data exposure. This demonstrates how important and versatile the tool is, highlighting how it can prevent the share of personal information with third party servers.
Before sending this email, the user could copy the anonymized version and paste it on the website, to avoid the unintentional exposure of sensitive data. The anonymized version could be:

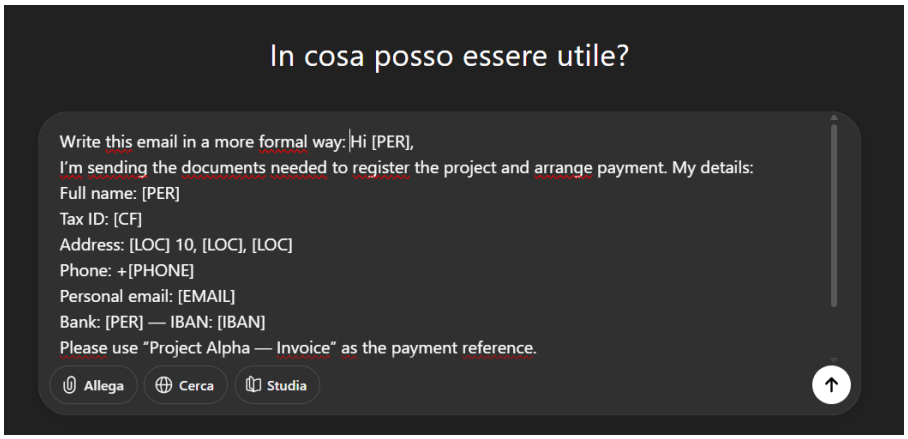

Figure 7.5: Example of Anonymized Email Output

This feature not only safeguards privacy, but also raises awareness of how personal data can be propagated through digital tools. This demonstrates how the system anonymization capabilities can be extended to real scenarios use cases, promoting a **privacy by design** approach even in modern tools such as AI assistants.

Results

The results showed that the replacement mechanisms maintained the **syntactic integrity of the text**, allowing anonymized content to remain fully readable. Moreover, the **replacement strategy without overlap**, which is when two entities could be mistaken as belonging to different entities at the same time, prevents the double substitutions and ensures that no parts are corrupted or truncated.
Furthermore, the placeholders conveyed efficiently the meaning of the hidden data, helping the users to understand which categories were exposed (such as names, contact details or financial data).
The anonymization functionality is particularly useful in practical use cases, such as in the ones shown in Figure 7.5 and in the figures in Section 7.2.1, where the readability and privacy must coexist.

7.2.3 Contextual and Correlation Risk Assessment

The **contextual risk assessment module**, implemented through the function saw in Chapter 6 (**anonymize_and_analyze()**) explained in Section 6.2.2, was evaluated to verify the correct application of the rule logic in different contexts:

- **Base Rules** defined generic sensitivity levels for each entity type;

- **Category Rules** adapted to the different domain in which the information is found
- **Micro-Context Rules** based on specific user actions (searching, posting, commenting).

In addition, **correlation analysis** function (**find_correlation_risks()**), saw in Section 6.2.2, identify:

- **Profiling across multiple sites Risks**, where the same entities are found across multiple domains or across at least two distinct domains ($min\_domain\_count \geq 2$);
- **Combination Risks**, which occur when multiple entities are detected in correlation within the same page or within the same profile, are associated with a risk.

The system's ability to perform these cross domain and cross session analysis validated the design goal of the system and offers a monitoring tool for privacy awareness that evolves from the single page session and expands to multiple domains.

Test 1: Combination Risks

The first test concerns **combination risks**, which are the threats coming from the co-occurrence of multiple entity types within the same user profile.
In contrast with single entity detection, these risks emerge when individual data, that is harmless in isolation, is combined to form a more complete and potentially identifying digital fingerprint.
In this experiment, the analysis was conducted using different websites with different prompts containing sensitive data.

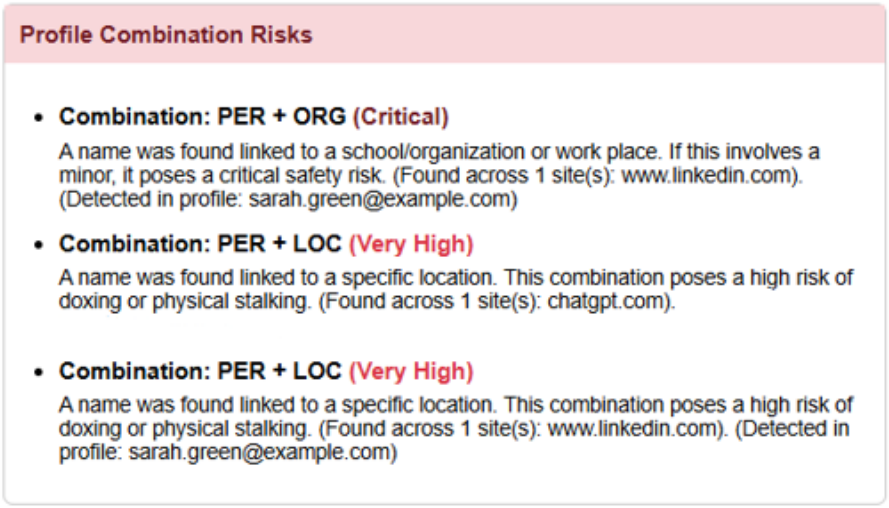

Figure 7.6: Risk Report Interface (Combination Risk)

As shown in Figure 7.6, the system correctly identifies the following correlations:

- **PER + ORG (Critical)**: A person name linked to an organization or workplace indicates a professional identity disclosure. In specific case, if the ORG

is not a workplace but a school and the individual is a minor, it poses a critical safety risk. For instance an attacker can exploit the affiliation to craft personalized phishing messages that appear legitimate, increasing the likelihood of internal compromise in an organization. Another risk with linking personal identity to a specific organization is that it can lead to reputational risks, as opinions or actions in personal posts might be incorrectly attributed to the employer. Moreover, this information can allow adversaries to map company structure, identify the staff members with access privileges, and plan target intrusion attempts.

- **PER + LOC (Very High)**: The association of a personal name with a location increases the exposure to doxing or physical stalking. This is detected in two different profiles and different domains in the report. Attackers can cross reference such data obtained from social media to reconstruct detailed personal profiles. This information can be used to assess where an individual lives and can be used in correlation to other information, such as holiday photos shared online, to enter the individual's house. For example, an Instagram story can be used to determine whether a person is at home or not, and this information, when correlated with PER+LOC (which could represent the street or neighborhood), can be exploited by an attacker to infer when the house will be unoccupied.

Each of these rules is dynamically evaluated on the configuration defined in **correlation_rules.json**, introduced in Section 6.2.2, which allows modular updates without code modification.
The system displays such risks under the **Profile Combination Risks** section of the report, assigning a color-coded severity risk ranging from *Low* to *Critical*.
These results confirm the capability of the correlation engine to identify not trivial privacy threats resulting from data aggregation, extending the analysis from simple entity detection to combination of different sensitive entities.
When the sensitive data is exposed, the combination of this data could be potentially more critical than the sharing of a single entity.

Test 2: Cross-Site Profiling

The second test focused on **cross-site correlation risks**, which refers to the privacy threats that appear when the same entity is found across multiple domains.
This test was designed to validate the system capability to leverage the persistent SQLite database to correlate data collected over time and across different contexts.
The dataset simulated multiple user interactions on different platforms, including professional networks (**linkedin.com**), social media (**facebook.com**) and AI interfaces (**chatgpt.com**).
Each platform provided partial data fragments, but when aggregated, they revealed a more comprehensive user identity.

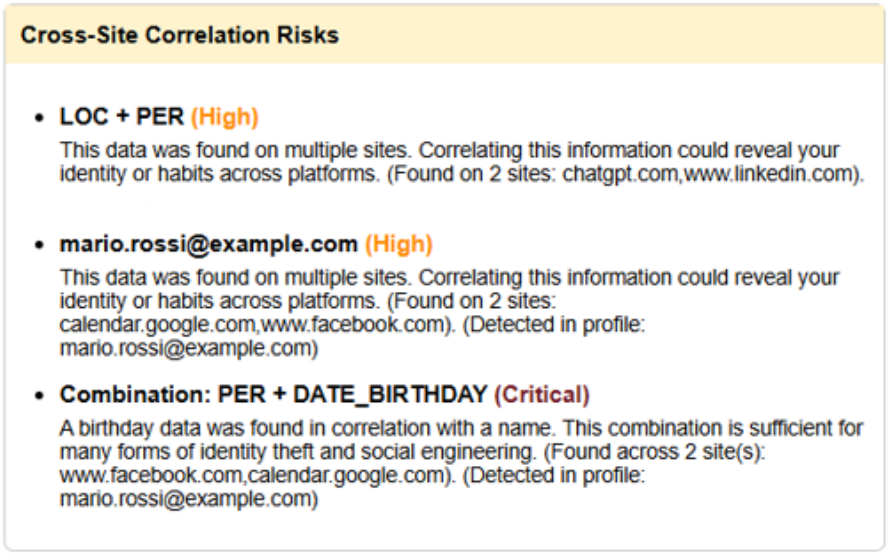

Figure 7.7: Risk Report Interface (Cross-Site Risk)

As shown in Figure 7.7, the system successfully detected:

- **Repeated entity matches** such as the email of Mario Rossi, was identified across two or more domains triggering high risk.
- **Relationships across entities** like LOC+PER, where the combination of a name and location appeared in multiple contexts. The risk is the same as explained in *Combination Risks* (7.2.3).
- **Aggregated Correlations**, such as PER+DATE_BIRTHDAY, identified across different websites, indicating potential for identity reconstruction. It facilitates the re-identification or identity theft; attackers can use this combination together with other leaked data to create or validate fraudulent accounts or financial requests. Moreover, that information can be used to reconstruct the *fiscal code* of an individual easily.

These findings are detected when $min\_domain\_count \geq 2$, in this way the information shared on different platforms can be linked to reconstruct a user's digital identity or to conduct advanced attacks using the information available online.
Overall, it demonstrates that the backend can infer complex, long term risks that go beyond isolated text inputs, highlighting the danger of exposing different information on different domains.

Test 3: Micro Context Rules

The third test was designed to assess the behavior of the **micro contexts** analysis module, which dynamically adjusts the privacy risk level according to the type of user interaction detected within the platform.
While the *macro contexts* (like facebook.com), define the general environment, the *micro context* represents the specific action performed by the user, such as commenting on a public post or sending a private message.
These distinctions are automatically inferred by the Chrome Extension based on

active input field's attributes, such as *placeholder* and *aria-label* (concept saw in Section 6.3.3). This information is crucial since depending *where* the information is shared, the risks change. A sensitive data shared in a private message have a different level of risk than the same data shared in a public post.
For instance, sharing a phone number in a *private message* typically limits its exposure to a single recipient and relies on an implicit expectation of confidentiality. Although still sensitive, the potential for misuse is restricted, as the information is not publicly accessible or indexed by search engines. In this scenario, the system assigned a *medium* level of risk, reflecting a controlled but still not negligible privacy concern.
In contrast, disclosing the same information in a **public post or comment** increases the risks. Once published, the content becomes visible to an undefined audience, can be archived, shared or scraped by automated bots, and may remain accessible indefinitely even after the deletion attempts. Such exposure enables secondary threats such as **data harvesting** or **identity profiling**, which justify a *Very High* or even *Critical* risk classification.
Moreover, the system also considers intermediate risk scenarios, like when data is shared in group posts or limited audiences, which present partial visibility. In these cases, the risk is adjusted dynamically, balancing between visibility, control and sensitivity of the disclosed information.
By differentiating between these micro context scenarios, the platform can provide a **granular and realistic assessment** of privacy risks, helping users understand that the same act of disclosure can vary dramatically in severity depending on *where* and *how* it occurs.
To demonstrate the impact of this micro context, the *same text* was evaluated in two different scenarios, a public comment and a private message.

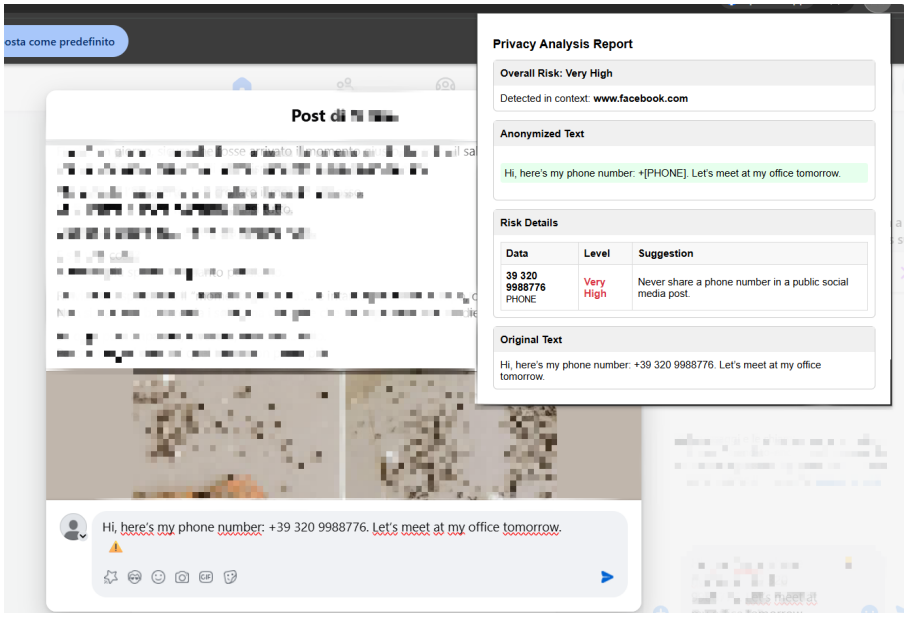

Figure 7.8: Risk Report Interface (Public Comment Context)

In the first case, as shown in Figure 7.8, the text was entered within a public post field (classified as **social_public_post**). The system correctly recognizes the

presence of a phone number and assigned a *Very High* risk level.
The corresponding explanation stated that sharing a phone number in a social media post exposes the user to unsolicited contact and potential privacy violations.
In the second test, the same message was sent through a *private chat* (classified as **social_private_message**).
As shown in Figure 7.9, the risk associated with the same entity is different and goes from *Very High* to *Medium*.

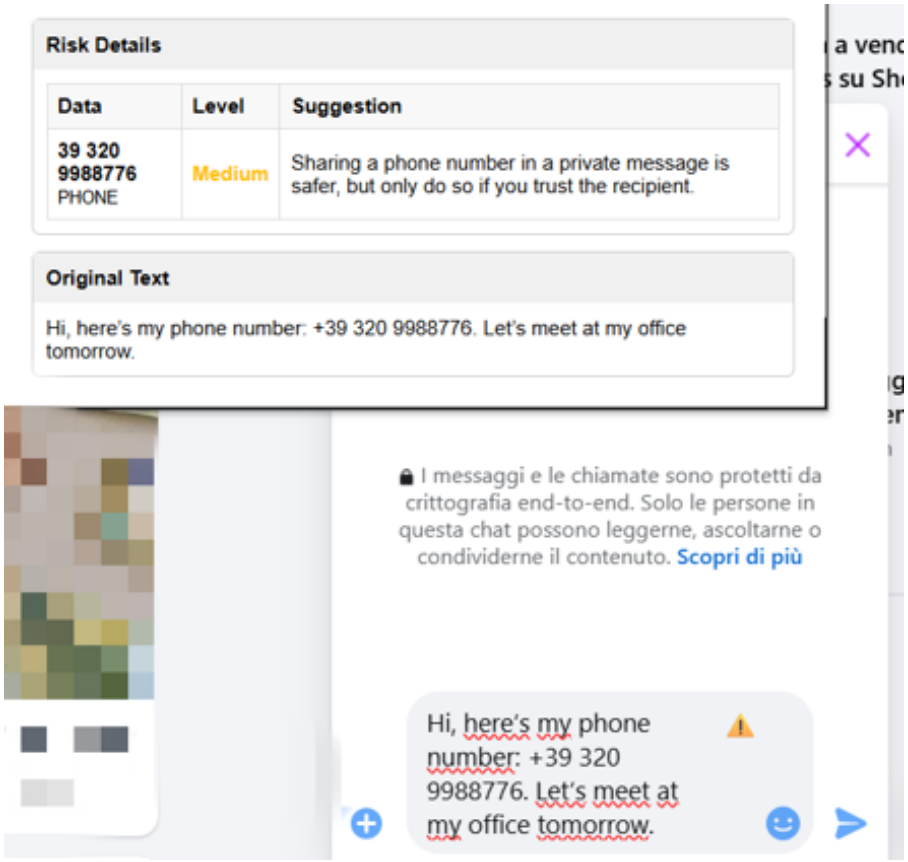

Figure 7.9: Risk Report Interface (Private Message Context)

Sharing a phone number in a private message is acceptable, provided that the recipient is trusted.
This adjustment reflects a more limited exposure surface, while the content still contains sensitive data, the likelihood of a public dissemination or automated collection is substantially reduced.
The contextual rule successfully overrides the base risk configuration, proving that the system can reason adaptively according to user intent and data exposure scope.

7.3 Performance Analysis

This section presents the quantitative performance evaluation of the system proposed by this thesis. The analysis focuses on two complementary aspects: the **accuracy** of the entity recognition component, and the **computational performance** of the backend and browser extension. Together, these results provide an

objective validation of the effectiveness of the system, its efficiency and its suitability for privacy monitoring in real time.

7.3.1 Detection Accuracy Analysis

The entity recognition component was evaluated using the metrics introduced in Section 7.1.1: **Precision**, **Recall** and **F1-Score**. These metrics quantify the system capability to correctly identify sensitive entities within the user generated content while avoiding false alarms.
The evaluation was conducted on a manually annotated dataset comprising multiple test samples, each containing one or more sensitive entities. The *ground truth* was constructed to include both **structured** entities (easily detectable by regex) and **contextual entities** (requiring interpretation by the NER model).
The algorithm 14 illustrates how the calculation of the metrics was performed.

Algorithm 14: Evaluation of Entity Detection Accuracy

Data: Evaluation dataset D containing text samples and ground-truth G entities

Result: Computed Precision, Recall, and F1-Score

1. Load dataset D from `evaluation_dataset.json`;
2. Initialize counters: $TP = 0$, $FP = 0$, $FN = 0$;
3. **foreach** *sample* (t, G) *in* D **do**
   Run `find_sensitive_entities(`t`)` to obtain predictions P;
   foreach *entity* g *in* G **do**
      if $g \in P$ **then**
         TP++;
      end
      else
         FN++;
      end
   end
   foreach *entity* p *in* P **do**
      if $p \notin G$ **then**
         FP++;
      end
   end
end
4. Compute metrics;
   $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, $F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$;
5. Print final results and percentages;

A detection is considered correct only when both the character span and the entity type coincide (strict span + label matching). This enforces a more rigorous evaluation by avoiding partial matches and ambiguous overlaps between entities.

Baseline Evaluation

Before integrating the hybrid detection architecture and contextual reasoning, an **initial baseline analysis** was performed to validate the testing methodology.
This early version of the system relied on a small set of regex patterns and basic entity tagging rules, without performing a more complex analysis.
The goal was to confirm that the dataset format and evaluation script with simple scenarios could achieve perfect accuracy.
The evaluation on a minimal dataset of three sentences yielded perfect results, as shown in the table below:

Table 7.3: Baseline Evaluation Results (Simplified Entity Detection Prototype)

Metric	**Value**
True Positives (TP)	5
False Positives (FP)	0
False Negatives (FN)	0
Precision	100.00%
Recall	100.00%
F1-Score	100.00%

At first glance, the results may appear ideal, however the test conditions were intentionally simplistic:

- All the entities were simple, explicit and clearly formatted.
- There was no contextual ambiguity,
- And the text samples did not include more complex and intricate scenarios.

This setup allowed performing a functional check of the testing procedure and to understand which are the performance under the simplest scenario.

Evaluation on Extended Dataset

A second and more complete evaluation was subsequently performed on an extended dataset of mixed domain samples, containing different sensitive elements from social platforms and diverse communication contexts (both private and public).
This version used a **hybrid detection architecture** combining regex rules with a BERT-based-NER model and contextual filtering to reduce overlaps and ambiguities.
The resulting metrics, reported in Table 7.4, demonstrate a more realistic level of performance.

Table 7.4: Final System Evaluation Results (Hybrid Detection Approach)

Metric	Value
Precision	92.31%
Recall	85.71%
F1-Score	88.89%

While the overall F1 score decreased to 88.89%, this reduction reflects the increased complexity and diversity of the dataset rather than a degradation of system capability.
Unlike the baseline prototype, the final implementation successfully managed an ambiguous, lowercased or partially specified entities, handling more complex configurations and contextual reasoning.

Comparative Analysis

In summary, the comparative analysis can be summarized in Table 7.5 that illustrates the evolution of the system analytical capabilities between a simpler scenario and a more complex one.

Table 7.5: Comparison Between Baseline and Final System Performance

Metric	Baseline Prototype	Final System
Precision	100.00%	92.31%
Recall	100.00%	85.71%
F1-Score	100.00%	88.89%

The **initial baseline** achieved *nominally perfect performance* (100% across all the metrics), because it was tested on a small and highly structured dataset where entities were explicit, unambiguous and syntactically consistent.
This configuration was useful to verify the correctness of the evaluation framework.
In contrast, the **final system** was tested on a *heterogeneous dataset* containing noisy, unstructured, and multilingual text that is more typical in real world content.
While this complexity caused a moderate reduction in numerical accuracy, it also validated the model's robustness, adaptability and context awareness.
In particular:

- The **drop in Precision** (-7.69%) corresponds to a broader interpretation of potential entities, acceptable in privacy contexts where false negatives are more critical than occasional false positives. However, this still indicates that the system most of the times correctly identifies the entities.
- The **decrease in Recall** (-14.29%) reflects the increased linguistic variability in the dataset, but still demonstrates strong coverage across multiple entity categories. Almost all the entities are found, but there are still some entities that are not correctly identified.

- The **F1 score** of 88.89% confirms a balanced tradeoff between accuracy and completeness, proving that the hybrid architecture generalizes well and effectively without compromising reliability.

These findings are significant because they demonstrate generalization since the system is able to handle complex and unpredictable user input. Based on this, this hybrid approach was validated on a more realistic dataset that represents the most common scenarios that could be found in social media posts, comments and others. Furthermore, the achieved *F1 score* near 90% confirms that the system is still providing high analytical accuracy while maintaining low latency and memory footprint as will be demonstrated in the next subsection 7.3.2.
Overall, these results validate the design choices adopted during the system development and confirm that the final prototype offers optimal balance between analytical accuracy and robustness of the system in detecting entities in dynamic scenarios.

7.3.2 System Performance Analysis

In addition to detection accuracy, performance was measured using a decorator, which records execution time and memory consumption for each analysis request.

Algorithm 15: Performance Profiling Decorator

Data: Flask API endpoint function f

Result: Execution time and memory variation per request

1. Retrieve process information using `psutil`;
2. Record initial timestamp t_{start} and memory usage m_{start};
3. Execute the target endpoint function $f(*args, **kwargs)$;
4. Record final timestamp t_{end} and memory usage m_{end};
5. Compute:
   - Execution time = $(t_{end} - t_{start}) \times 1000$ (ms)
   - Memory change = $m_{end} - m_{start}$ (MB)
6. Print formatted profiling results in the backend logs;
7. Return the original function result;

The decorator is a lightweight profiling mechanism designed to monitor the backend's **execution time** and **memory consumption** for each API request.
When applied to the server endpoint (such as /api/analyze, as shown in Section 6.2.2), the decorator captures the start time and current memory usage of the running process before the function is executed. After the endpoint completes, it measures the end time and final memory usage. Using these values it computes the execution duration (in milliseconds) and the change in resident memory (RSS), expressed in megabytes.

It is important to note that the **Execution Time** measured by the decorator represents exclusively the processing time of the backend server, which is the time required by the server application to perform all the mathematical and logical operations after receiving a request.
This value depends entirely on the **computational power of the local machine (CPU or GPU, if enabled)**, and it is not influenced by Internet connectivity.
Even though browser communication occurs via HTTP requests, all the data exchanges take place in a loopback interface (127.0.0.1). Therefore, variation in network speed or internet connection quality cannot directly influence the latency. The only thing that can be influenced by network quality is the initial loading of the AI model.
This profiling method offers a not intrusive, since it does not require any modification to the API logic, and provides performance monitoring in real time after every request. Moreover, it allows identification of all the potential bottlenecks or memory leaks during the testing phase.
In the Table 7.6, there are the results obtained in the first run:

Table 7.6: Profiled Requests: Session A

Request #	**Execution Time (ms)**	**Memory Change (MB)**
Run 1	**393.62**	**235.80**
Run 2	130.78	0.00
Run 3	150.55	0.00
Overall Average	**224.98**	**78.60**
Steady State Average	**140.67**	**0.00**

*The highlighted row indicates the model **warm up** phase, during which the BERT model is loaded into memory. The **steady state average** excludes this initialization run, representing the typical runtime performance under normal operating conditions.*

A significant memory change of approximately 236 MB was observed during the first request (Run 1), together with an execution time of 393.62 ms.
The execution time is the first time the API was called after starting the server, this action took about 0.4 seconds. This time includes not only the text analysis itself, but also the initial "warmup" of AI library components, which are fully loaded into the memory only on their first use.
The memory change is the most important part, since it tells that during this first call the application had to allocate about 236 MB of additional RAM. This is not a "memory leak", but an expected behavior. Libraries such as PyTorch and Transformers are lazy: they load the base models at startup but only allocate the actual working memory (for computations, tensors, etc.) the first time they perform an analysis.
In general, this behavior corresponds to the **model warmup phase**, in which the *dslim/bert-base-NER* Transformer Model is loaded into memory for the first time.
The spike in both memory allocation and latency is expected and it is not indicative of inefficiency, but rather of **normal initialization overhead**.
Once the model was fully loaded, the subsequent requests (Runs 2 and 3) have

zero memory variation and reduced execution times (130-150ms).
This confirms that the model and related components are **cached in memory** and **reused efficiently** by the backend without reloading between analyses.
The analysis takes only about 0.13-0.15 seconds. This is the *actual execution time* of the algorithm, at this moment all the components are "warm" and ready in the memory. This is very fast for a model that performs Regex analysis, two passes of AI models and complex risk logic.
Moreover, the fact that the memory change is 0 MB, which means that there is no need to allocate any new memory, demonstrates that the system efficiently reuses the memory it had already allocated.
Overall, the results are highly positive, showing the backend behaves as expected: a short initialization phase followed by fast and stable memory operation.
Table 7.7 shows the results of a second analysis conducted:

Table 7.7: Profiled Requests: Session B

Request #	**Execution Time (ms)**	**Memory Change (MB)**
Run 1	**1232.66**	**235.88**
Run 2	746.62	0.00
Run 3	590.23	0.00
Run 4	239.74	0.00
Run 5	214.67	0.00
Run 6	816.43	0.00
Overall Average	**640.06**	**39.31**
Steady State Average	**521.54**	**0.00**

The second profiling section was conducted in a new session.
As seen in the previous case, the first request (Run 1) is the largest computationally with an execution time of 1.23 seconds and a memory allocation of 235.88MB.
Again, this increase corresponds to the loading of the BERT model and tokenizer from the disk into the memory.
The initialization delay is higher than in Session A and this is largely due to two main reasons: **larger and more complex input text** were used in this test, which required additional preprocessing and entity parsing; and **other applications** running concurrently have **consumed part of the GPU resources**. Since this process is dependent on the GPU, this causes latency on the execution time.
The subsequent requests (Runs 2-6) displayed stable memory usage (0 MB variation) and execution times between 214 ms and 816 ms.
The variability across these runs depends mainly on: the **length of the analyzed text**, the **number of detected entities** and corresponding regex/NER operations and the presence of **correlation checks** in the database.
Despite these natural fluctuations, all the requests are completed well below 1.5 seconds threshold for browser interaction in real time.
The average performance across this session, which is 640 ms per request, is **excellent**, given that this test included more complex analyses across domains.
The results of Session B reinforce the **model warmup pattern** that is reproducible

and limited to the first request of a session. The system has consistent performance stability after initialization and even with more complex texts the backend maintains **responsive latency and controlled memory change**.
Finally, the results of this session are not negative, but instead reflect the system ability to process larger and multiple context inputs without sacrificing the responsiveness or by exceeding the resource limits.

Comparative Analysis

The figure 7.10 shows the execution time for both Session A and B.

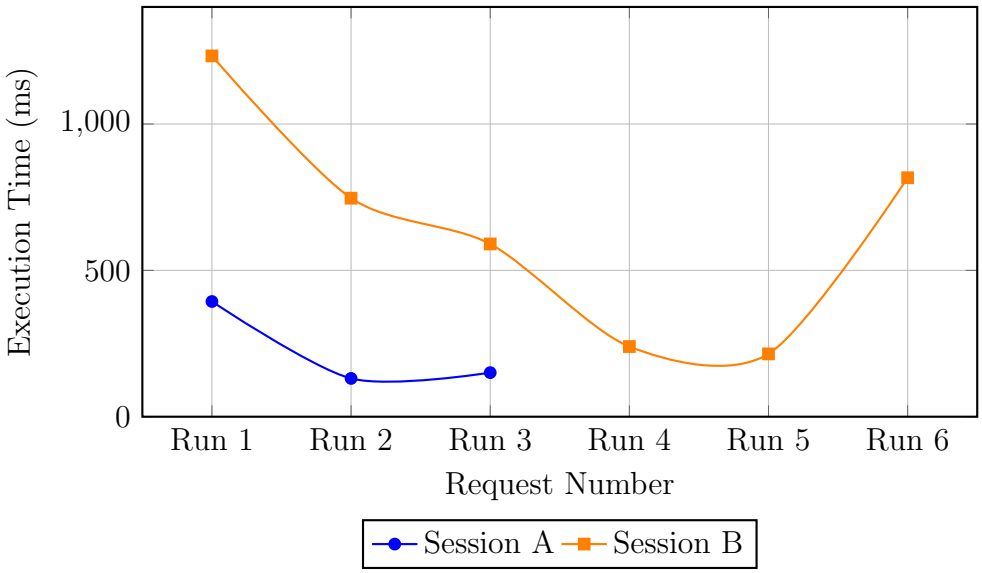

Figure 7.10: Execution Time Trends Across Requests for Session A and Session B

Figure 7.11 shows the memory variation recorded after each request. This is similar for Session A and B since they have almost the same memory change.

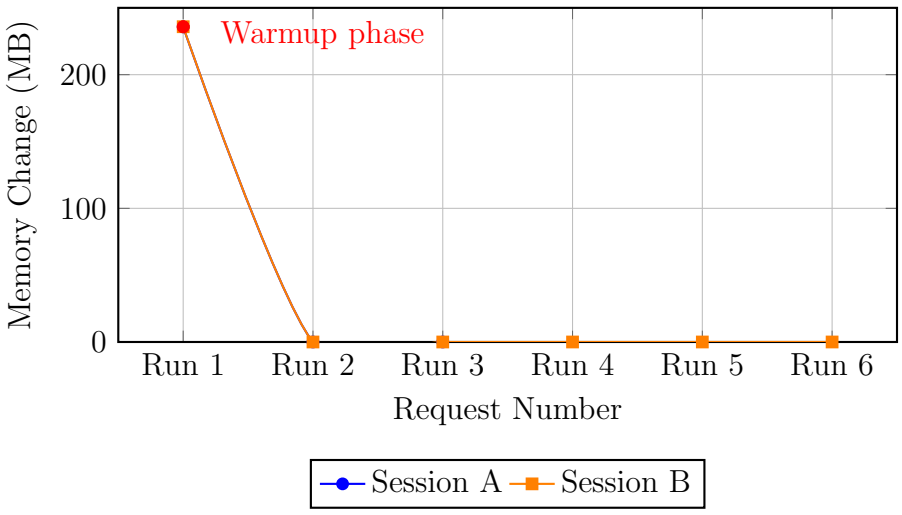

Figure 7.11: Memory Variation Across Requests for Sessions A and B

When the two sessions are compared, a clear trend emerges:

- Both sessions have a **single warmup peak** in memory usage, confirming predictable and controllable model initialization.
- After this phase the system operates in a **steady state** with lower latency and negligible memory fluctuations.
- The difference in average execution time is attributed primarily to input complexity and GPU/CPU consumption.

This behavior is correct in monitoring systems in real time, since it ensures that the user experience is unaffected.
Moreover, the fact that the memory stays constant after the initialization highlights that the system is optimized for repeated analysis, which is important for a continuously active Chrome extension backend.
In conclusion, the results from both sessions demonstrate excellent performance stability, resource reuse and scalability.
The backend achieves a good balance between analytical depth (due to the Transformer model) and computational efficiency, confirming the fact that it can be used for privacy text processing in real time.

Additional Tests

In order to verify whether the system could maintain its analytical performance while reducing GPU and memory consumption, an additional series of execution tests was performed.
All the experiments were executed under the same hardware configuration described in Table 7.1 using the decorator to record the performance profile of each API call.
These experiments were conducted to validate the hypothesis that the increased latency observed in Session B (Table 7.7) was caused by higher GPU workload.
By repeating the same analysis under controlled conditions, it was possible to assess the benefits of the model reuse and confirm the system's stability over multiple consecutive runs.

Table 7.8: Profiled Requests: Session C

Request #	**Execution Time (ms)**	**Memory Change (MB)**
Run 1	**423.83**	**240.92**
Run 2	148.48	0.00
Run 3	146.96	0.00
Run 4	147.97	0.00
Overall Average	**216.81**	**60.23**
Steady State Average	**147.80**	**0.00**

The second analysis showed the following results:

Table 7.9: Profiled Requests: Session D

Request #	Execution Time (ms)	Memory Change (MB)
Run 1	**421.02**	**235.99**
Run 2	147.92	0.00
Run 3	157.72	0.00
Run 4	146.92	9.07
Overall Average	**218.40**	**61.27**
Steady State Average	**150.85**	**3.02**

In this case the occasional small memory change (+9 MB) is due to the fact that the operation conducted required tokenization of the input text and it is not a sign of a memory leak.
This is demonstrated by the report below, which was extracted when the execution time (146.92 ms) and memory change (9.07 MB) were calculated.

Risk Details

Data	Level	Suggestion
G PER	Low	Sharing your full name on a professional network like LinkedIn is expected and necessary.
##iu PER	Low	Sharing your full name on a professional network like LinkedIn is expected and necessary.
Microsoft ORG	Low	A company name is generally public.

Figure 7.12: Risk Report Interface (Tokenization)

The *dslim/bert-base-ner* model uses *WordPiece subword tokenization* (which is the standard in BERT). Instead of operating on whole words, the tokenizer breaks *rare or out of vocabulary (OOV)* words into *subtokens* that exist in its vocabulary.
In the report, "Giulia" was tokenized in ["G", "##iu", "##lia"]. The model then predicts NER labels at the subtoken level. Typically, only the first subtoken of an entity is labeled at the beginning (B-PER), while the continuation receives inside tags (I-PER). Since "Giulia" is **uncommon in the English training data** (the model is trained on English corpora like CoNLL-2003), the tokenizer splits it into subtokens and the model *may not reconstruct the full name clearly.* This leads to a fragmented output such as G - PER, ##iu - PER (or unlabeled/low confidence), ##lia (often ignored or mislabeled). In the report only G and ##iu were tagged as PER, instead of a single and clean "Giulia".
That small bump in memory is expected and it is caused by **temporary tensor allocations** that grow with the number of subtokens produced by the tokenizer:

1. Subwords tokenization produced more tokens than usual, three tokens instead of one.

2. Each token becomes an *embedding vector*, which means more tokens and larger input tensors.
3. During inference, the model allocates *temporary buffers* (on GPU/CPU) proportional to the input length.
4. This accounts for the transient 9 MB increase in memory when processing the input.

After forward pass completes, these allocations are released and memory returns to its baseline, which is why in subsequent runs the memory returns to a 0 MB change.
A possible solution could be to *merge contiguous subtokens* that share the same entity group; if consecutive tokens belonging to PER are adjacent in text they are merged back to the original surface form using the original character spans. Moreover, using a *multilingual* model could represent another possible solution.
Despite this behavior, those scenarios represented in Tables 7.8 and 7.9 highlight that the initial behavior, when the model is loaded, follows the same pattern and the optimization of the model reuse is always present. This underlines how the system is **energy-efficient** and **scalable**.

7.4 Overall System Evaluation

The integrated evaluation of the system provides a complete view of both its analytical accuracy and its computational efficiency. The results collected across the different testing scenarios (functional and performance) confirm that the proposed solution achieves a balanced tradeoff between *reliability, adaptability and resource optimization.*
From a *functional* perspective, the detection architecture successfully combines regex precision with NLP reasoning.
The system was able to identify structured entities with high accuracy while maintaining consistent performance for contextual entities that require semantic understanding.
Quantitatively, the *entity detection accuracy* reached a precision of *92.31%*, a recall of *85.71%* and an overall F1 score of *88.89%*, which represents a strong level of reliability considering the heterogeneity of the dataset deployed. These results indicate that the system generalizes well across multiple linguistic and contextual variations, maintaining robust detection even with more complex texts.

Table 7.10: Comparison Between Baseline and Final System Performance

Metric	**Baseline Prototype**	**Final System**
Precision	100.00%	92.31%
Recall	100.00%	85.71%
F1-Score	100.00%	88.89%

Regarding *performance evaluation*, the system exhibited predictable and stable behavior across all the profiling sessions (A-D).

This is illustrated by the plot below that summarizes the performance across all sessions:

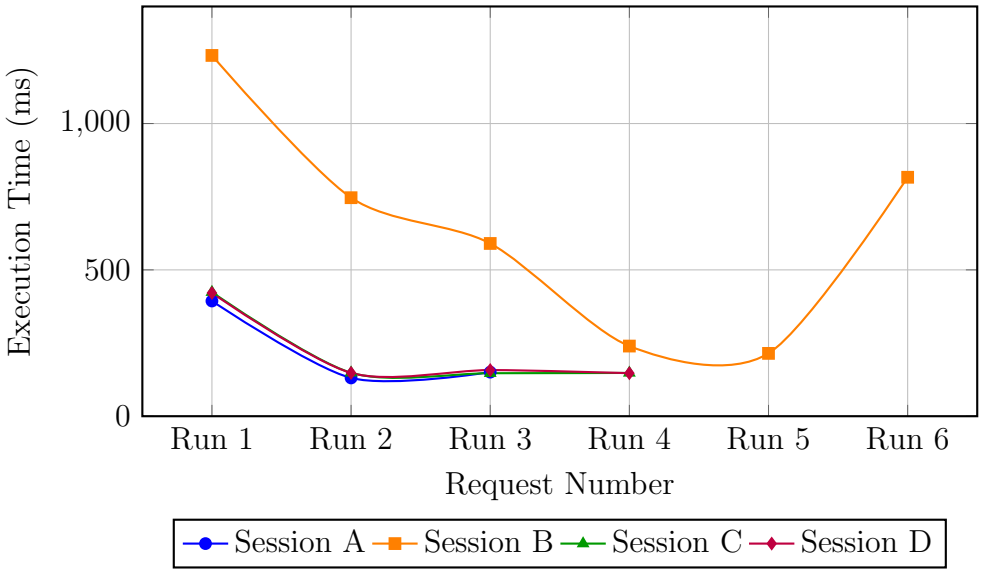

Figure 7.13: Execution Time Trends Across Requests for Sessions A, B, C, and D

The average processing time per request stabilized at approximately 150-200 ms after the model warmup, ensuring usability in real time in an interactive browser extension.
Memory consumption was also efficiently managed: after the initial model load, subsequent analyses showed *negligible additional memory usage*, confirming that the model is reused efficiently across multiple API calls.

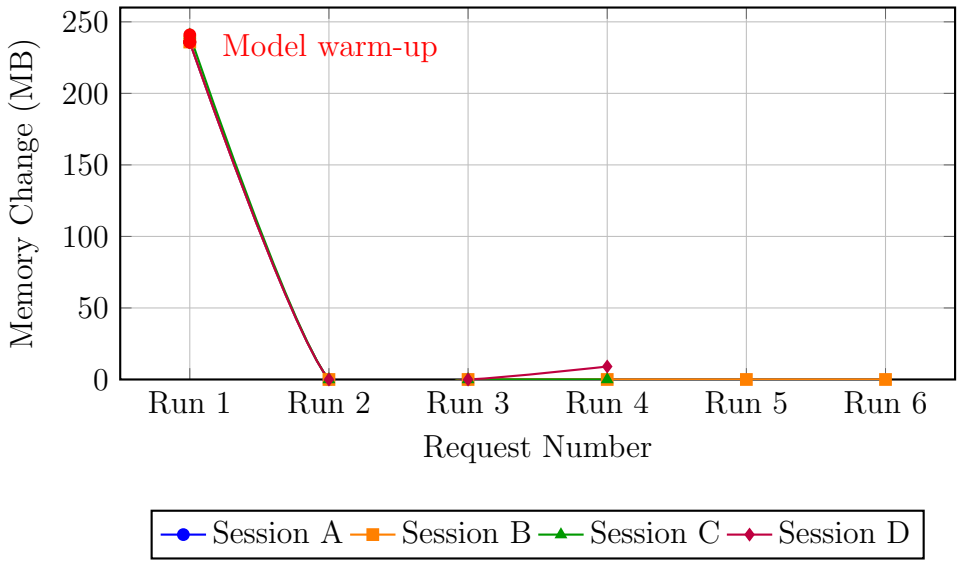

Figure 7.14: Memory Variation Across Requests for Sessions A–D.

Minor increases (9 MB) were observed during Session D involving complex tokenization, corresponding to dynamic tensor allocations.

The comparison among sessions demonstrated a consistent improvement in runtime efficiency and GPU utilization, confirming caching and model persistence mechanisms significantly reducing system overhead. These results validate the architectural choices adopted in the design phase, particularly the decision to offload the model inference to a local backend rather than reloading it at each request.
Finally, from an operational perspective, the system achieved a good balance between *privacy protection, interpretability and scalability.* The report and the anonymization pipeline successfully identify and replace sensitive entities, allowing the users to understand how their personal data could be exposed.
The *contextual engine* proved capable of reasoning dynamically based on *where* and *how* information is shared, assigning appropriate levels of risk based on context, micro-context, domain in which the information is shared.
In summary, overall evaluation confirms that the system meets its objectives: it operates with high accuracy, maintains computational performance and can adapt risk assessment based on the context. These characteristics allows to have real-time monitoring to ensure privacy-aware web environments, offering both *reliability* and *interpretability* for end users.

Chapter 8

Conclusions and Future Works

The goal of this project was to overcome the limitation of traditional privacy and anonymization tools, which typically work in a reactive and decontextualized manner. The objective was to design and implement a proactive system for digital privacy monitoring that is capable not only of identifying sensitive data but also of assessing its risk based on deep contextual analysis, and of tracking the exposure of the user's digital profile over time.

This objective has been successfully achieved through the development of a full-stack architecture, composed by a Flask backend and a Chrome browser Extension.

The main accomplishments of this work are several. First of all, a successful implementation of an analysis engine that combines the precision of Regular Expressions (regex) for structured data with the flexibility of a Transformer based AI model (BERT/NER) for more complex entities. The logic was refined to handle edge cases, such as lowercase scenarios and to resolve conflicts between different detection methods.

Secondly, the system implements a sophisticated three layered risk logic (Base<Site Category<Micro-context), configurable through external JSON files. This allows the system to dynamically assess the severity of a potential data leak, distinguishing between sharing a phone number in a public post versus in a private message. Also, to differentiate between the share of information in different domains.

Moreover, by integrating a SQLite database, the application evolved from stateless tool into a stateful system. User profiles, anonymized through SHA-256 hashing, act as anchors for aggregating information collected across multiple websites. The correlation engine can infer complex risks that emerge only from linking seemingly innocuous data pieces.

Lastly, the system's performance has been objectively evaluated. Tests demonstrated excellent efficiency, with an average response times at steady state between 150-220 ms, ensuring a seamless user experience. Accuracy test, conducted on a validation dataset, yielded an F1 Score near 90%, confirming the reliability of the detection engine.

In summary, the developed prototype is not merely a masking tool, but rather a **Privacy-as-a-Service** platform that provides user with real time awareness of their digital footprint, representing a significant step toward a more proactive and intelligent management of online privacy.

Although the current system is already robust, it provides a strong foundation for several promising future developments. For instance, introduce a multimodal analysis to process images before upload, including OCR (Optical Character Recognition) with Tesseract to extract and analyze text from images and use Zero Shot Classification using models like OpenAI CLIP to detect visual risk contexts based on customized risk labels. Another evolution could be to evolve the correlation engine from SQL queries to a Graph Neural Network (GNN) system. This would enable the discovery of more complex relationships between entities, modeling the digital profile as a true knowledge graph. Moreover, extend the extension to other browsers and develop plugins for other platforms where sensitive data is shared, such as email clients. Finally, the system could be extended by adopting a multilingual model, enabling the interpretation of text in multiple languages.
These directions define the natural evolution of this work, aiming to transform the current prototype into a comprehensive platform for dynamic personal data protection.

Bibliography

[1] L. Sion and W. Joosen, "Linddun pro privacy threat modeling tutorial (version 0.1)," Department of Computer Science, KU Leuven, Tech. Rep. Technical Report, LINDDUN PRO, Apr. 2023, pre-release version, Version 0.1, April 2023; distrinet, KU Leuven. [Online]. Available: https://downloads.linddun.org/tutorials/pro/v0/tutorial.pdf

[2] Wikipedia contributors, "Facebook: Cambridge analytica data scandal," Wikipedia, The Free Encyclopedia, Tech. Rep. Wikipedia Article, 2010. [Online]. Available: https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal

[3] Save the Children Italy, "Sharenting: what does it mean and which are the risks for children," Save the Children Italia, Tech. Rep. Blog post, Jun. 2024, last Update: April 2025. [Online]. Available: https://www.savethechildren.it/blog-notizie/sharenting-cosa-significa-e-quali-sono-i-rischi-i-bambini

[4] European Union Agency for Network and Information Security (ENISA), "Handbook on security of personal data processing," European Union Agency for Network and Information Security (ENISA), Tech. Rep. WP2017 O-2-2-5 GDPR Measures Handbook, Dec. 2017. [Online]. Available: https://www.enisa.europa.eu/sites/default/files/publications/WP2017%20O-2-2-5%20GDPR%20Measures%20Handbook.pdf

[5] European Data Protection Board (EDPB), "Guidelines 07/2020 on the concepts of controller and processor in the gdpr," European Data Protection Board, Tech. Rep. Guidelines 07/2020, Jul. 2021. [Online]. Available: https://www.edpb.europa.eu/sites/default/files/consultation/edpb_guidelines_202007_controllerprocessor_en.pdf

[6] European Union, "Regulation (eu) 2016/679 of 27 april 2016, article 4: Definitions," Official Journal of the European Union, Tech. Rep. General Data Protection Regulation, Apr. 2016. [Online]. Available: https://gdpr-info.eu/art-4-gdpr/

[7] National Institute of Standards and Technology (NIST), "Guide to protecting the confidentiality of personally identifiable information (pii)," National Institute of Standards and Technology, Tech. Rep. Special Publication 800-122, Apr. 2010. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf

[8] European Data Protection Board (EDPB), "Guidelines 4/2019 on article 25: Data protection by design and by default," European Data Protection Board, Tech. Rep. Guidelines 4/2019, Oct. 2020, last updated: 20 october 2020. [Online].

Available: https://www.edpb.europa.eu/our-work-tools/our-documents/guidelines/guidelines-42019-article-25-data-protection-design-and_en

[9] A. Cavoukian, "Privacy by design: The seven foundational principles," The Sedona Conference, Tech. Rep. Recommended [08b], Feb. 2017, pDF accessed provides guidance on the 7 foundational principles of Privacy by Design. [Online]. Available: https://www.thesedonaconference.org/sites/default/files/conference_papers/Recommended%20%5B08b%5D%20Privacy%20By%20Design_Cavoukian.pdf

[10] National Institute of Standards and Technology (NIST), "Glossary: Cybersecurity incident," NIST: Small Business Cybertraining Corner, Tech. Rep. Online Glossary, Feb. 2019, definition of "Cyber Incident" retrieved from the NIST glossary. [Online]. Available: https://www.nist.gov/itl/smallbusinesscyber/training/glossary#:~:text=An%20occurrence%20that%20actually%20or%20potentially%20jeopardizes%20the,security%20policies%2C%20security%20procedures%2C%20or%20acceptable%20use%20policies.

[11] Victoria Drake, "Threat modeling," OWASP, The Open Web Application Security Project, Tech. Rep. OWASP Community Page, Sep. 2025, retrieved September 2025. [Online]. Available: https://owasp.org/www-community/Threat_Modeling

[12] OWASP Developer Guide community, "Linddun go: Owasp developer guide," OWASP: The Open Web Application Security Project, Tech. Rep. Online Guide Page, Sep. 2025, retrived September 2025. [Online]. Available: https://devguide.owasp.org/en/04-design/01-threat-modeling/05-linddun-go/

[13] European Union, "Regulation (eu) 2024/1689, article 3: Definitions," Official Journal of the European Union, Tech. Rep. Artificial Intelligence Act, Jul. 2024. [Online]. Available: https://artificialintelligenceact.eu/article/3/

[14] David S. Lim (dslim), "dslim/bert-base-ner," https://huggingface.co/dslim/bert-base-NER, 2024, model BERT fine-tuned for Named Entity Recognition on dataset CoNLL-2003; license: MIT.

[15] National Institute of Standards and Technology (NIST), "Risk management framework for information systems and organizations: A system life cycle approach for security and privacy," National Institute of Standards and Technology, Tech. Rep. Special Publication 800-37, Revision 2, Dec. 2018. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

[16] K. Pakhale, "Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges," https://arxiv.org/pdf/2309.14084, Tech. Rep., Sep. 2023, retrived September 2025.

[17] GeeksforGeeks, "Named entity recognition with bert," GeeksforGeeks, Tech. Rep. Tutorial / Web article, Jul. 2025, last updated: 18 Jul 2025. [Online]. Available: https://www.geeksforgeeks.org/data-science/named-entity-recognition-with-bert/

[18] DeepLearning.ai, "Natural language processing (nlp): A complete guide," DeepLearning.ai, Tech. Rep. Online Guide, Jan. 2023, last updated January 11, 2023. [Online]. Available: https://www.deeplearning.ai/resources/natural-language-processing/

[19] GeeksforGeeks, "Named entity recognition," GeeksforGeeks, Tech. Rep.

Tutorial / Web Article, Jul. 2025, last updated: 23 Jul 2025. [Online]. Available: https://www.geeksforgeeks.org/nlp/named-entity-recognition/

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint, Tech. Rep. arXiv:1810.04805, May 2019, version v2, submitted 24 May 2019. [Online]. Available: https://arxiv.org/pdf/1810.04805

[21] G. Rodolà and psutil community, "psutil documentation," https://psutil.readthedocs.io/en/latest/, 2025, version current at access time.

[22] G. E. Vaciago, "Lecture notes from the course "cybersecurity laws and regulations"," Politecnico di Torino, 2024, unpublished lecture notes, 2024.

[23] E. Tjong Kim Sang, "eriktks/conll2003: Conll-2003 named entity recognition dataset," https://huggingface.co/datasets/eriktks/conll2003, Jan. 2024, last updated Jan 18, 2024.

[24] GeeksforGeeks, "Flask tutorial," GeeksforGeeks, Tech. Rep. Online Tutorial, Oct. 2025, last Updated: 07 Oct 2025. [Online]. Available: https://www.geeksforgeeks.org/python/flask-tutorial/

[25] P. S. Foundation, "time: Time access and conversions (python 3.13.6 documentation)," https://docs.python.org/3/library/time.html, 2025, consulted on August 2025.