UNIVERSIDADE D
COIMBRA

João Filipe Carnide de Jesus Nunes

DATA PROCESSING AND VISUALISATION IN REPUTATION SYSTEMS FOR ENTITIES IN IOT

Dissertation in the context of the Master in Informatics Engineering, specialisation in Software Engineering, advised by Professor Bruno Sousa and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

João Filipe Carnide de Jesus Nunes

Data Processing and Visualisation in Reputation Systems for Entities in IoT

Dissertation in the context of the Master in Informatics Engineering, specialisation in Software Engineering, advised by Prof. Bruno Sousa and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

João Filipe Carnide de Jesus Nunes

Processamento e Visualização de Dados em Sistemas de Reputação para Entidades em IoT

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pelo Professor Doutor Bruno Sousa e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro 2023

Acknowledgements

The work presented would not be possible without the commitment and support of several people. Therefore, I would like to thank everyone who helped me during this dissertation, directly or indirectly.

First and foremost, I would like to thank my advisor, Professor Bruno Sousa, for his patience, trust and assertiveness. Having had the chance to collaborate with him was genuinely delightful. For his understanding when there was a problem in the project and it was not solved immediately, and for his dedication in always being available when any doubt arose, no matter how small. Undoubtedly, this work would not be possible without the support of Professor Bruno.

This work was supported by the ARCADIAN-IoT project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101020259. I would also like to thank some members of this project who helped me throughout this work, namely André Melo and Tomás Caçoete, for their help with the data visualisation platform and Professor Luís Paquete for his advice on reputation models.

Next, I would like to thank my colleagues and friends Diogo Dória, Alexandre Tapadinhas, Duarte Meneses, Mariana Paulino, Rafael Baptista, Oleksandr Yarotskyi, Patrícia Costa and Eduardo Nunes for their tremendous help in the usability tests. Their collaboration helped to develop a cleaner product.

To NEI/AAC for helping me become more organised, selfless and responsible over the last three years. To the many dedicated people I've met during this journey who work towards making DEI a better place for students, a sincere thank you.

To Projeto TAU and its members who have helped me become a better person, for reminding me that there is more beyond work by making our Friday night meetings an escape from the hustle and bustle of my life.

I want to name a few people, some of them already mentioned above, who during this year were an asset in supporting me, whether to motivate me or to make me take a break from work. A huge thank you to Dinis Carvalho, João Cruz, Bernardo Graça, Eva Filipe, Pedro Gil, José Henrique, Jaime Marques, Marta Santos, Mariana Loreto, Bernardo Arzileiro, Ana Sofia Santos, Mariana Silva, Lucas Anjo, Maria Mansilha, Sara Panão, Daniel Diogo and Sofia Bidarra.

Finally, to my family, who always believed in me and gave me unconditional support in my endeavours despite having little or no understanding of the subject.

Abstract

In today's age, reputation systems are becoming increasingly important to measure entities' trustworthiness and reliability in various contexts, including online marketplaces, social media platforms, and Peer-to-Peer networks. Moreover, modern reputation systems often distribute one of their critical features. As a result, decentralised networks like blockchain technology build them rather than a single entity or organisation controlling them.

There are several methods for determining the reputation of entities (e.g., people or objects). These methods include using ratings and reviews and analysing data from the entity's previous behaviour.

One of the critical challenges in implementing and maintaining a reputation system is the need to visualise the collected data effectively. The visualisation helps users understand the reputation of a particular entity at a glance and helps to identify trends and patterns in the data that may be useful for making decisions about trust and reliability. Different approaches to visualising reputation data include graphs, charts, and other visual aids.

The main goals of this work are to validate several reputation models for the system's entities - people, applications/services and objects -, integrate these models into a reputation system component, and evaluate their performance. Additionally, we developed a data visualisation platform to make it easier to understand and interpret the reputation information of all the entities present in the system.

It is necessary to consider particular elements when building a reputation system to achieve these goals. This work presents these considerations. Additionally, we discuss the implementation and testing of the developed components. We aim to contribute to the field of reputation systems and provide a valuable resource for those interested in understanding and utilising these systems.

Keywords

Reputation Systems, Data Visualisation, Trust, Distributed Systems, Entity.

Resumo

Atualmente, os sistemas de reputação tornam-se cada vez mais importantes para medir a confiabilidade e fiabilidade das entidades em vários contextos, incluindo mercados online, plataformas de redes sociais, e redes *Peer-to-Peer*. Além disso, os sistemas de reputação modernos distribuem frequentemente uma das suas características críticas. Como resultado, redes descentralizadas como a tecnologia de *blockchain* constroem-nas em vez de uma única entidade ou organização que as controla.

Existem vários métodos para determinar a reputação das entidades (p.e., pessoas ou objetos). Estes métodos incluem a utilização de classificações e avaliações e a análise de dados do comportamento anterior da entidade.

Um dos desafios cruciais na implementação e manutenção de um sistema de reputação é a necessidade de visualizar eficazmente os dados recolhidos. A visualização ajuda os utilizadores a compreender a reputação de uma determinada entidade num relance e ajuda a identificar tendências e padrões nos dados que podem ser úteis para a tomada de decisões sobre confiança e fiabilidade. Existem muitas abordagens diferentes à visualização de dados de reputação, incluindo gráficos, diagramas, e outros auxílios visuais.

Os principais objetivos deste trabalho são validar vários modelos de reputação para as diferentes entidades do sistema - pessoas, aplicações/serviços e objetos -, integrar estes modelos numa componente de sistema de reputação, e avaliar o seu desempenho. Adicionalmente, foi desenvolvida uma plataforma de visualização de dados para facilitar a compreensão e interpretação da informação de reputação de todas as entidades apresentadas no sistema.

É necessário considerar elementos particulares na construção de um sistema de reputação para alcançar estes objetivos. Este trabalho apresenta estas considerações. Além disso, é discutida a implementação e teste dos componentes desenvolvidos. Pretende-se que este trabalho contribua para o campo dos sistemas de reputação e forneça um recurso valioso para os interessados em compreender e utilizar estes sistemas.

Palavras-Chave

Sistemas de Reputação, Visualização de Dados, Confiança, Sistemas Distribuídos, Entidade.

Contents

Acronyms

AI Artificial Intelligence.

API Application Programming Interface.

CERT Computer Emergency Response Team.

CORE Community Reputation Mechanism.

CSIRT Computer Security Incident Response Team.

CSV Comma-Separated Values.

CTI Cyber Threat Intelligence.

D3 Data-Driven Documents.

DAG Directed Acyclic Graph.

DDoS Distributed Denial of Service.

DEI Department of Informatics Engineering.

DGA Drone Guard Angel.

DRBR Dominance Relationship-Based Reputation.

DTRMS Distributed Trust and Reputation Management Systems.

EC European Commission.

eUICC embedded Universal Integrated Circuit Card.

FIRE Fair, Incentivised, Reputable, and Engaging.

GDPR General Data Protection Regulation.

HTTPS Hypertext Transfer Protocol Secure.

IoT Internet of Things.

JSON JavaScript Object Notation.

LOS Line of Sight.

MIoT Medical IoT.

MITM Man-in-the-Middle.

ML Machine Learning.

P2P Peer-to-Peer.

REGRET Reputation Mechanism for Generalised Trustworthiness Evaluation.

RoT Root of Trust.

SSI Self-Sovereign Identity.

SSL Secure Sockets Layer.

TOTP Time-Based One-Time Password.

UC University of Coimbra.

List of Figures

List of Tables

Chapter 1

Introduction

Reputation systems include information on certain levels of trust between two entities that may be strange to each other. For example, the web widely uses evolved reputation systems, electronic transaction sites like eBay, and social networks.

Solutions to build reputation models can be based on devices' interactions with each other and in which hashing mechanisms assure that the data is kept with integrity. Some models, though seemingly basic, like counters and accumulators, are sophisticated versions of reputation primitives [8]. Their simplicity doesn't undermine their utility, as variations of the following models are widely prevalent online:

- Favourites-and-Flags - provide the community with tools to recognise and highlight items that are either exceptionally good or exceptionally poor in quality;
- Voting - allows community members to vote on the usefulness, accuracy, or appeal of something;
- Ratings-and-Reviews - a user rates a target with scores and adds freeform text opinions, which collectively contribute to a community average;
- Karma - the reputation of users can be assessed through models that evaluate both the extent of their participation and the quality of their contributions.

Many website operators initially find these straightforward models highly suitable for their requirements.

Additional tools use Machine Learning (ML) techniques to distinguish benign from malignant nodes, where ML models reduce learning time by distributing reputation information using different data sources to guide the learning process. For example, if a particular data source has a high reputation, the model might trust the data from that source more. Conversely, if a data source has a low reputation, the model might place less trust in that data and possibly exclude it from the training process.

The reputation calculation should also consider the type of entity in question or even the information they gave, for example, when people rate a service on a 5-star scale. In this case, dominance models that aggregate the data from several evaluations can help determine reputation.

Users of a Distributed Trust and Reputation Management Systems (DTRMS) can rate and review one another based on past interactions. Users rate and check each other based on past interactions. This information calculates the user's reputation score in a stored decentralised platform like blockchain. People can use DTRMS in various contexts, including online marketplaces like those mentioned above, with eBay or Amazon.

One key benefit of DTRMS is that it can provide a more accurate and trustworthy representation of a user's reputation. The decentralised nature of the platform makes it more difficult for a single entity to manipulate the reputation of an individual. Additionally, DTRMS can help to increase accountability and reduce the risk of fraud or malicious behaviour. Users are more likely to behave responsibly if they know their actions will be recorded and used to calculate their reputation score.

One of the main challenges online communities face is the problem of accurately assessing the reputation of their members. Reputation systems are designed to address this problem by allowing community members to rate and evaluate each other based on their contributions and behaviour. Unfortunately, most reputation systems can only use one way to measure reputation, which may only work well for some groups or communities with unique needs. For example, in online forums, a reputation model based on the number of posts and likes may not accurately reflect a user's reputation. This project aims to address this limitation by implementing a reputation system that is more flexible and adaptable to the specific requirements of different communities.

This project aims to provide a more effective solution to the problem of reputation assessment in online communities.

1.1 Main Objectives

This dissertation has two main objectives.

The first objective is to process and analyse the reputation data received from different components, each with distinct functions. The analysis is done by selecting the essential information from an entity's events and applying a reputation model to calculate the reputation of an entity. Additionally, this project's innovative aspect lies in using the most appropriate reputation model to accurately determine the reputation of diverse entities and adapt the model more favourably given additional factors (e.g., the alpha-beta model adapted to include a severity factor).

The second objective is to develop a visualisation platform, facilitating data analysis and providing a clear and intuitive representation of the results.

Further information about the objectives of this project is featured in Section 3.1.

1.2 Contribution

This dissertation is inserted in the ARCADIAN-IoT project[1], a research and innovation project funded by the European Commission (EC). The project aims to develop a trust, security, and privacy management framework for Internet of Things (IoT) systems. This framework aims to accelerate the development of decentralised, transparent, and user-controllable privacy in IoT systems, focusing on three real-world use cases, two of which are included in this dissertation in Chapter 4. Furthermore, the project aims to make this framework available to various contexts and applications.

Therefore, the main contributions of this dissertation are the following:

1. **Research and analysis of technologies for reputation measurement** (see Section 2.2.5). By understanding the technologies and methods that are available for measuring reputation, it is possible to select the most appropriate approach for a given context or application and to design and implement IoT systems that are secure, reliable, and trustworthy.

2. **Investigation of technologies to visualise reputation information** (see Section 2.3.1). Data visualisation can improve the transparency and accountability of reputation systems. By providing a clear and easily understood representation of the data, data visualisation can make reputation systems more transparent and open to scrutiny, enhancing trust and confidence in these systems.

3. **Implementation of a reputation manager for the different entities of the system**. With the identification of adequate reputation measurement models, we apply these models in the project to calculate the reputation of each entity and manage all the received events.

4. **Evaluation of the reputation models**. Upon implementation of the reputation models in the project, a thorough validation was conducted to assess the suitability of these models for calculating the reputation of the system's entities.

5. **Development of a data visualisation platform**. We created a data visualisation platform to help users quickly see all the reputation information of the different entities in the system. This platform makes it easier for users to decide how trustworthy a specific entity is during its stay in the system.

6. **Writing a scientific article**. The work compiled also allows us to write a scientific article regarding data processing and visualisation in reputation systems by organising the information and results of the project clearly and concisely and presenting it in an engaging and informative way.

[1] *ARCADIAN-IoT*: `https://www.arcadian-iot.eu/`

1.3 Document Structure

The remainder of the document consists of the following chapters.

Chapter 2 introduces the environment and related work with the fundamental concepts necessary to understand this work. It shows the definition of a distributed reputation system, the different dimensions we can get a value of the reputation of a system and the other existing solutions of specific platforms. It also provides existing solutions for visualising data and, respectively, a way of getting the reputation information of certain entities in a system.

Chapter 3 explains the motivation behind this dissertation and the main goals to achieve. Furthermore, this chapter presents the research and development approach, the documentation of the main issues, and how we will handle them.

Chapter 4 contains requirements artefacts produced, such as use cases, non-functional requirements, mockups of the visualisation platform, and a requirement listing that establishes the scope of our work.

Chapter 5 provides a detailed description and analysis of the project's architecture, including its components, design, and functionality.

Chapter 6 delves into the implementation of the reputation system and the data visualisation platform, which provides valuable insights into the performance and activity of the entities present in the system.

Chapter 7 covers the testing and validation of the models implemented in the project. It involves the development of appropriate test cases and scenarios, the execution of the tests, and analysis of the results to determine the accuracy, reliability, and effectiveness of the models.

Chapter 8 provides the closing remarks summarising this work's outcome and providing content for future work regarding the project's scope.

Appendix A documents all the tests to validate the alpha-beta model.

Appendix B documents all the exchanges that the reputation system analyses to calculate the reputation score.

Appendix C presents all the pages and functionalities of the data visualisation platform.

Chapter 2

Background & Related Work

This chapter covers the fundamental concepts necessary to understand our work. Section 2.2 the definitions and reasons behind the need for a Reputation System in a specific platform, along with different dimensions of taxonomy and ways of calculating an entity's reputation. Section 2.3 explains the various forms of implementing the data visualisation platform, including the mechanisms for making graphs with reputation information.

For the reader to understand the whole project, Section 2.1 presents the ARCADIAN-IoT project, its different domains, and the role of University of Coimbra (UC) in this project.

2.1 ARCADIAN-IoT

The ARCADIAN-IoT project is an initiative that aims to address the trust, security, and privacy challenges that arise when implementing IoT systems. The project aims to develop a framework to accelerate the development of IoT systems that prioritise decentralised, transparent, and user-controllable privacy. The project aims to demonstrate the framework's effectiveness through three real-world use cases.

The European Commission has funded the project, focusing on Autonomous trust, security and privacy management, which means the framework is designed to work independently, seamlessly integrate into various contexts, and manage the trust, security and privacy of the IoT system.

The ARCADIAN-IoT framework is evaluated in three realistic domains, two presented in further detail in Section 4.1.

- **Emergency and vigilance using drones and IoT.** This domain was chosen due to the significance of IoT in digital services for emergency and vigilance within the business model of LOAD. However, robust solutions must address trust, security, and privacy aspects. The implementation of ARCADIAN-IoT will address these issues and improve the data privacy

compliance, such as General Data Protection Regulation (GDPR), of the primary operational platform of LOAD, which includes drones equipped with IoT devices for emergency and vigilance scenarios.

- **Secured early monitoring of grid infrastructures.** This domain was chosen given the relevance of IoT in Industrial Control Systems for public infrastructures and factories. ARCADIAN-IoT will innovate the rapid deployment, secured and efficient solution for early monitoring of industrial and public (e.g. smart cities) grid primary circuits.

- **Medical IoT.** This domain has been selected in light of the growing trend towards utilising IoT solutions for telemedicine applications. Providers of these devices must be equipped with reliable and secure solutions.

The University of Coimbra is responsible for implementing the reputation system to provide a trust model for various entities, including objects, individuals, and applications/services. This component will model reputation based on the trust properties of the ARCADIAN-IoT trust plane, which will consider the interactions between entities of the same type and different types. The reputation system component will establish a trust model by considering the behaviour of devices, social interactions between entities, the mobility of nodes/objects, and the resources of IoT devices. This will ensure a secure and trustworthy model to evaluate the reputation of diverse entities in the ARCADIAN-IoT ecosystem.

2.2 Reputation System

The need for reputation systems can only grow in importance as the world becomes increasingly interconnected. The main goal of a reputation system is to facilitate trust between two or more entities who might have never interacted with each other. Social networks and e-commerce are perfect examples of platforms that might benefit from a reputation for the different entities involved [16].

The term reputation has several definitions. According to the Cambridge English Dictionary[1], reputation is "the opinion that people, in general, have about someone or something, or how much respect or admiration someone or something receives, based on past behaviour or character". On the other hand, some articles describe reputation as the "perception that an agent creates through past actions about its intentions and norms" [24].

Reputation systems collect, distribute, and aggregate feedback about participants' past behaviour [30], seeking to establish the shadow of the future of each transaction by creating an expectation that other people will look back on. The idea is that referring to the reputation data will help people decide whom to trust, encourage trustworthy behaviour and deter participation by unskilled or dishonest people.

[1] *Cambridge Dictionary*: `https://dictionary.cambridge.org/`

The system can also use reputation to describe a group or an individual. The system can compute a group's reputation in many ways, such as considering the average of all the members' reputations or how the exterior perceives them.

These systems have three challenging operating phases: eliciting, distributing, and aggregating feedback. We encounter three related problems regarding this feedback elicitation: people might need to be more bothered to provide input, and it is difficult to elicit negative feedback and ensure direct reports.

2.2.1 Trust, Risks and Reputation

Defining reputation and trust can be complicated [25], as they are often used as synonyms, even though their meaning is distinctly different.

Every person's opinion differs, making reputation a highly personal and subjective quantity [32]. Therefore, reputation is not what character someone has but rather what character other thinks someone has.

In terms of trust, Jøsang et al. [15] define trust as "the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible". The key concepts in this definition are dependence and reliability, which are measured partially through a person's reputation. We can build that trust through an entity's reputation, and a better reputation can potentially lead to more considerable trust.

Risk can be seen as a situation where the outcome of a particular transaction can be crucial to someone, but there is a possibility of failure. Considering the concepts referred above of reputation and trust, they can be related: the amount of risk a party is willing to tolerate is directly proportional to the amount of trust the party has in the other party [12].

The reputation systems aim to support the establishment of trust between unknown entities and, according to Dellarocas [7], "generate sufficient trust among buyers to persuade them to assume the risk of transacting with complete strangers".

2.2.2 Reputation Models

Understanding the different available reputation models is essential for designing and implementing effective reputation systems. Other models may be more suitable for different contexts and offer additional benefits and limitations. By examining the various reputation models, we can better understand how they can be used to support decision-making processes and how they can be improved. By considering the different available reputation models, we can choose the most appropriate for a given context and use it to measure and represent reputation accurately.

A taxonomy and commercial reputation systems are presented in [12]. To index

reputation systems consistently and meaningfully, two reference models were developed.

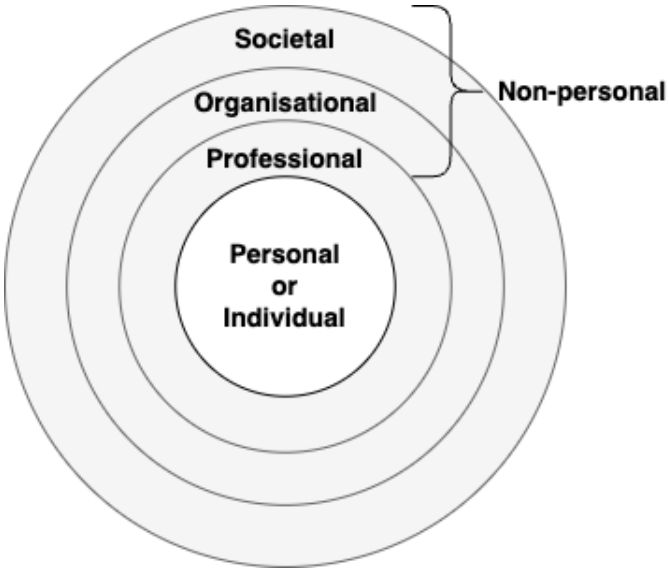


Figure 2.1: Reference Model for Reputation Context

Figure 2.1 depicts a reference model for the reputation context to present different contexts in which we could retrieve reputation (described in [12] as *Contextual Attributes*). These Contextual Attributes will help us reach a more accurate reputation score. This model, starting from the innermost ring, represents the various contexts that go from personal (who), professional (what), organisational (which/membership) and societal (where).

Most online reputation systems focus only on a person's reputation, whilst several real-world situations deal with non-personal aspects, such as the person's professional and organisational membership.

Defining all the entities involved and their potential interactions is essential in reputation systems. Figure 2.2 presents this idea and aims to generalise the model for reputation systems approaches.

The entities presented in this model are the Trustor, the Trustee and the Recommender. The Trustor is the entity that wants to trust and interact with a target entity, the Trustee. To decide whether to trust the Trustee, the Trustor must evaluate the Trustee's reputation [18]. This evaluation is done by first consulting its reputation information. However, suppose there were no previous interactions between the Trustor and Trustee. In that case, the Trustor will then query 1 or n Recommenders that may have had previous interactions or observed an interaction with the Trustee for their opinion.

A Recommender may be an entity that provides information from its history of transactions or a system that observes the interaction between two entities or collects data from other sources. With the appropriate information, a Recommender may reply with a recommendation (also called feedback). Using this reputation information, the Trustor can decide to trust the Trustee. The roles of the three

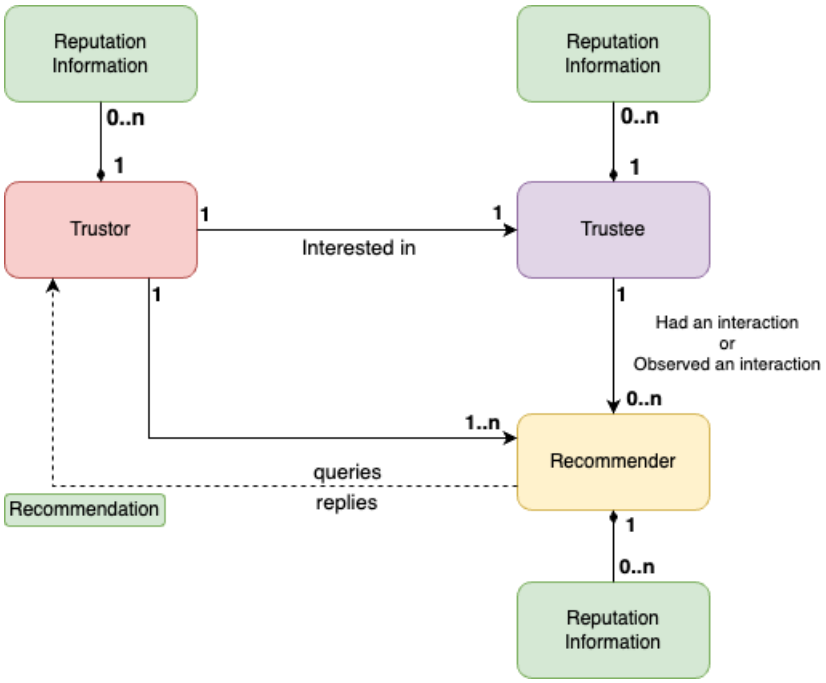


Figure 2.2: Reference Model for Reputation Systems

entities in the model may be interchangeable [19]. If the transaction proceeds, both entities will have their reputation information that may be available to other parties making equivalent decisions.

2.2.3 Reputation Systems Dimensions

Reputation systems evaluate and measure individuals or entities' trustworthiness, reliability, and credibility in a given context. According to [12], several dimensions can be used to evaluate the effectiveness of a reputation system. In this section, we discuss these dimensions in detail and examine how they impact the functioning of reputation systems.

Understanding the various dimensions of reputation systems is crucial for designing and implementing effective systems that accurately reflect the reputation of individuals or entities. These dimensions can also help us understand the strengths and limitations of different reputation systems and how they can be improved. By examining these dimensions, we can better understand how reputation systems operate and how they can support decision-making processes in various contexts.

Firstly, [12] presents a taxonomy for reputation systems that distinguishes implicit from explicit reputation.

Implicit reputation mechanisms represent systems that have not defined a reputation system, have little to no structure, and are used to ensure that participants remain honest. Recent platforms such as Facebook or LinkedIn use this type of reputation where entities extract some degree of trust through friends of friends or connections of connections.

Explicit reputation systems are the ones whose implementation is purposeful to facilitate the estimation of trust between members of an environment and rely on interactions with diverse members. This type of reputation has a set of dimensions, some of which are used more commonly than others.

Figure 2.3 presents a visual representation of all the dimensions in this section and their different types.

Common Dimensions

1. **History**

   A user's history is classified as the stored information recording their past interactions and outcomes. This history helps determine the likely outcome of current or future transactions and is crucial for reputation.

   - Personal: Personal history is created and maintained using directly collected or observed information, leading to a unique view of other entities.
   - Global: Global history is created and maintained from information shared by other members interacting with the system, leading to a consistent, broad view of every entity in the system.

2. **Context**

   Contextual information can give a lot of meaning to data by describing various details regarding interactions occurring [2]. Therefore, as mentioned in Section 2.2.2, we adopt contextual attributes to discuss and categorise reputation systems that utilise fine-grained and transaction-specific contextual information.

   - Single: A single context is assumed or maintained within the context of the system.
   - Multiple: The system maintains one or more contexts.
   - Attribute: The system maintains contextual attributes, called multi-faced, dimensional, or attribute-based.

3. **Collection**

   For a reputation system to demonstrate trust, it is necessary to capture the behavioural information of entities.

   - Direct: Information is generated explicitly from an individual's interactions or observing others' transactions.
   - Indirect: Obtaining information from other entities based on transactions that the querying entity was not privy - witness information.
   - Derived: Information is obtained from a source not explicitly designed to be used as a reputable source in the current context.

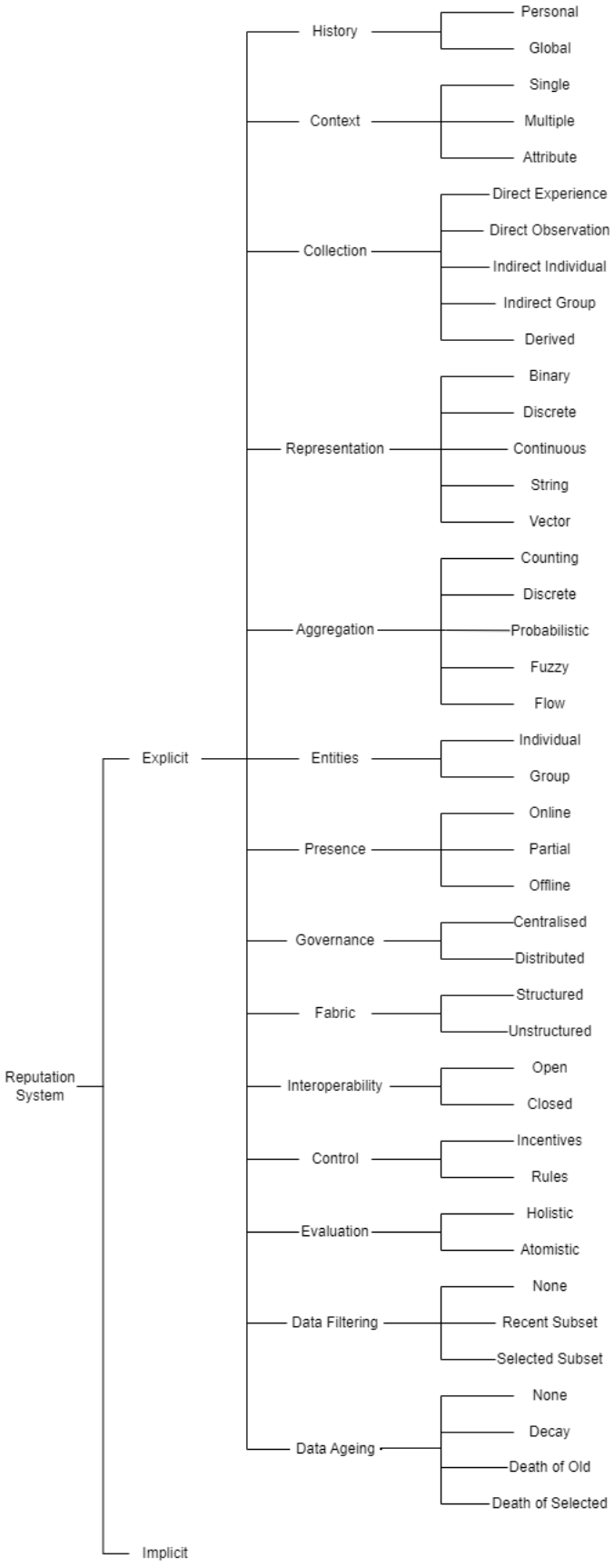


Figure 2.3: Visual representation of the taxonomy

4. **Representation**

   The format selected to describe, exchange and interpret reputation data. It has different types of information used.

   - Binary: Information is stored using boolean values.
   - Discrete: Information is stored using discrete integer values.
   - Continuous: Information is stored as a floating point number.
   - String: Information is stored in a textual form, allowing a wide range of data to be maintained.
   - Vector: Information provided by multiple sources or explicitly separated for individual use.

5. **Aggregation**

   Describes how a reputation score for an entity is computed. The simplest form of aggregating reputation is the summation of all the positive and negative ratings for an entity. A slightly improved approach is to average all the ratings to produce a single rating for each entity. These methods, like summation, averaging, weighting and normalisation, are aggregation methods that fall into a single class of simple computation called counting. We can classify different types of aggregation methods:

   - Counting: Reputation is calculated by either summing positive and negative ratings or average ratings.
   - Discrete: Reputation is computed by converting discrete rating values using look-up tables.
   - Probabilistic: Ratings are fitted into a probability model and used to predict the likelihood of a hypothesis being correct, often taking the shape of "Is entity x trustworthy?".
   - Fuzzy: Fuzzy logic is used to process or compute ratings, allowing these systems to work with a degree of uncertainty.
   - Flow: Reputation is computed by examining the flow of transitive trust - a two-way relationship created between the two entities involved.

Uncommon Dimensions

6. **Entities**

   Entities are the main focus or target of a reputation system. A reputation system's targets are typically people or resources (for example, books or electronic products), and both are typical system, first-class members.

   - Individual: These systems focus on people or specific resources
   - Group: These systems are focused on groups rather than individuals, possibly both formal and informal, with the former assuming some of the characteristics of an organisation.

7. **Presence**

   Describes how closely a reputation is tied to its underlying reputation system.

   - Online: Those systems that require the continuous presence of authority to distribute reputation information.
   - Partial: Those systems that do not require the continuous presence of authority to be able to distribute reputation information.
   - Offline: Those systems that do not require the presence of authority to be able to distribute reputation information.

8. **Governance**

   Reputation systems are volatile environments with entities and information changing frequently. Therefore, some level of authority is required for the system to work correctly. Governance describes how the system is controlled and is described in Section 2.2.4.

   - Centralised: The system is organised by a central group or organisation. Most commercial reputation systems use this centralised governance where the underlying architecture may be distributed, but the management is most likely by a single organisation.
   - Distributed: Multiple entities working together, frequently with no centralised management. Most current Peer-to-Peer systems display distributed governance.

9. **Fabric**

   Describes how the nodes of the reputation system are organised to allow the systems to be easily categorised and differentiated between them.

   - Structured: New nodes are assigned a location and a set of neighbours in an organised way when connecting to a network [22].
   - Unstructured: Allows new nodes to connect randomly, without organisation [22].

10. **Interoperability**

    Describes the fundamental principles by which the system operates and shares information. Nowadays, with restricted control of the reputation system, the information presented is not shared with any third parties.

    - Open: Entities freely access and utilise the reputation data in a system using data standards.
    - Closed: Reputation information is proprietary and not often shared outside the system.

11. **Control**

    Expressing how a reputation system motivates and controls entities to act in a desired manner is essential to implementation.

- Rules: An entity is obligated or limited to act restrictedly.
- Incentives/Disincentives: An entity is motivated or guided using rewards and punishments to obtain appropriate behaviours.

12. **Evaluation**

    In terms of evaluation, the reputation system can provide two different views of past transactions when obtaining or viewing the available reputation information of the trustee.

    - Atomistic: A detailed transaction-based view that potentially shows all of the trustee's interactions.
    - Holistic: All of an entity's interactions are considered and weighted to provide a single, overall view of the trustee to the trustor.

13. **Data Filtering**

    - None: There is no limit or filter for the data in the system.
    - Subset: The data is limited by the application of data filtering, where the trustor may only use a subset of all the available history of the trustee.

14. **Data Ageing**

    It is helpful to reduce confidence in information as time passes and more information is collected. This decay of information allows entities to distance themselves from historical behaviour.

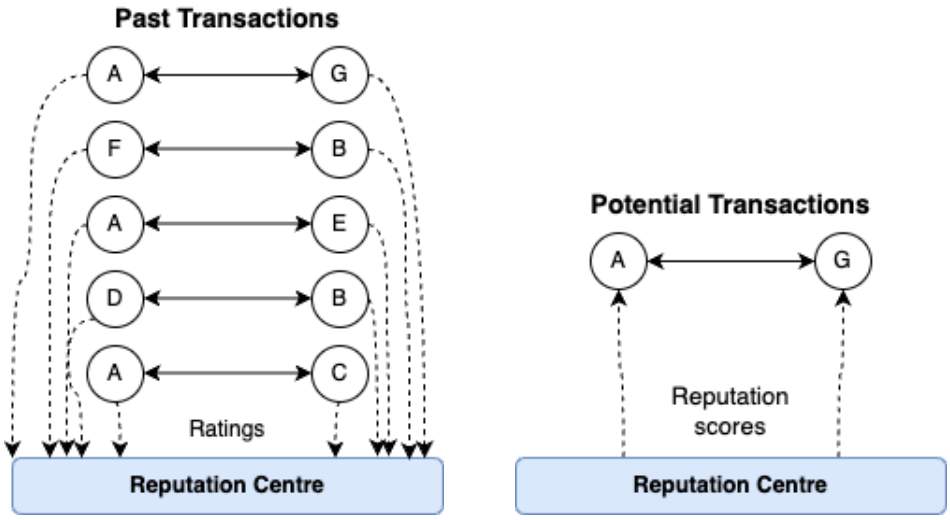
    - None: Reputation information is retained indefinitely.
    - Decay: Reduces the confidence and granularity of older reputation information as time passes.
    - Death: An extension of decay allowing older reputation information to be discarded [38]. This discard of information is based on age or a manual selection.

2.2.4 Reputation Network Architecture

The network architecture determines how ratings and reputation scores are used between entities of a reputation system [14]. The two fundamental types of architecture are centralised and distributed.

Centralised Reputation Systems

In centralised reputation systems, information about a particular entity is collected as ratings from other members in the system who have had direct experience with that entity. The central authority (reputation centre) that contains all the ratings typically derives a reputation score for every entity, making every



(a) Past (b) Present

Figure 2.4: General framework for a centralised reputation system

score available to the public. Transactions with reputable participants will likely result in better outcomes than transactions with disreputable participants.

As shown in Figure 2.4, A and B represent transaction partners with a history of past transactions who consider transacting with one another in the future.

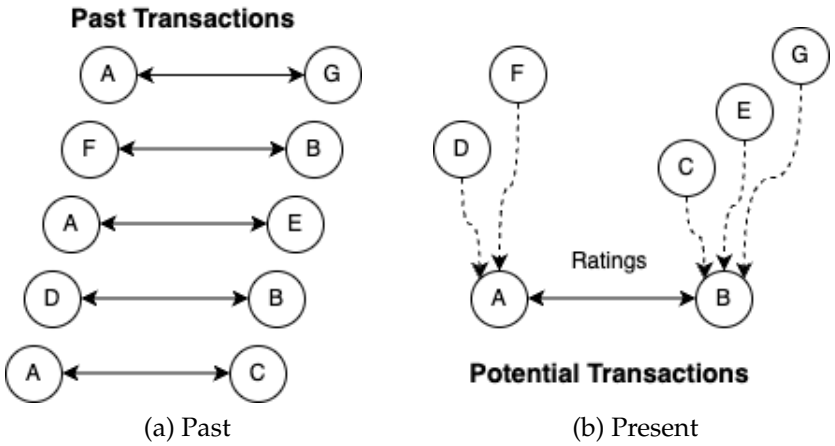
When a transaction occurs, the entities provide ratings about each other's transaction performance, and the reputation centre will then collect the ratings, updating the reputation score of each entity. There are two main aspects of centralised reputation systems:

a) **Centralised communication protocols** allow entities to provide ratings about the transaction partners to the central authority and obtain reputation scores of potential transaction partners from the reputation centre.

b) The reputation centre uses a **reputation computation engine** to manage reputation scores for each entity based on the ratings received and other possible information.

Distributed Reputation Systems

There are domains where distributed reputation systems, i.e. without any centralised functions, are much better suited than centralised ones. In these systems, there are no central locations for the rating or reputation score submissions. As a replacement, there can be distributed stores where ratings can be submitted, or the entities can register their opinion about their experience with other system members, providing information on request from relying parties. For example, to get information from the trustees that a certain trustor is willing to do a transaction with, the trustor must find the distribute stores or try to obtain ratings from as many members of the system as possible who have had previous direct experience with the trustee, as seen in Figure 2.5.



(a) Past (b) Present

Figure 2.5: General framework for a distributed reputation system

The relying party calculates the reputation score based on the ratings it receives. Considering a direct experience between the trustor and trustee in the past, that can be considered private information. There are two main aspects of distributed reputation systems:

a) A **distributed communication protocol** allows participants to obtain ratings from other entities present in the system.

b) Each agent uses a **reputation computation method** to derive the reputation scores of target parties based on received ratings and additional possible information.

Peer-to-Peer (P2P) networks represent an environment well-suited for distributed reputation management. In these networks, every node of the system plays the client and server role, also called *servent*, allowing users to overcome their passive role typical of web navigation and engage in an active role, providing their resources. The P2P networks have two phases:

1. The **search phase** involves locating the servent where the requested resources reside. In some P2P networks, this phase relies on centralised functions. One example of this implementation is Napster [27], which uses a resource directory server. With a purely distributed P2P network, Gnutella [10] and Freenet are good examples.

2. The **download phase** follows the search phase, where the requested resources have been located, which consists of transferring the help from the exporting to the requesting servent.

P2P networks have a range of security threats, as users can use them to spread malicious software, such as viruses and Trojan horses, and easily bypass firewalls. These networks also have confirmation that they suffer from free-riding [3] - someone who wants others to pay for the public good but plans to use it themselves. The purpose of a reputation system in P2P networks is to determine the following:

a) The number of servents that are most reliable at offering the best quality resources and

b) Whichever servents provide the most reliable information concerning soft security.

Participants in a distributed environment are responsible for collecting and combining ratings from other members. Ratings resulting from all interactions with an agent may be impossible or too expensive in some cases. Users can also use ratings from the relying party's neighbours to calculate the reputation score.

The project will adopt a distributed approach due to its advantages over centralised systems. This approach is deemed more secure, scalable, and fair, providing a refined architecture. The utilisation of a distributed reputation system, as opposed to a centralised one, offers increased security through decentralisation and reduces the risk of a single point of failure. This decentralisation is achieved by implementing various components of the reputation system across multiple nodes with distinct roles within the system.

2.2.5 Reputation Measurements

Reputation calculation is determining the reputation of an individual, organisation, or other entity based on various factors. Reputation is a subjective measure of an entity's perceived quality, trustworthiness, or other characteristics, and it can significantly impact an entity's success and influence.

Many different methods and approaches can be used to calculate reputation, and the specific method used will depend on the calculation's context and purpose. Some common approaches to reputation calculation include using feedback or ratings from other entities, analysing an entity's past behaviour or performance, and comparing an entity's characteristics or attributes to those of other entities.

Regardless of the approach used, reputation calculation provides a quantifiable measure of an entity's reputation that a certain trustor can use to evaluate its quality or trustworthiness. System members can then use this measure for various purposes, such as determining an entity's eligibility for specific opportunities or privileges or as a factor in decision-making processes.

Ultimately, at the end of this section, a summary of all the reputation measurement techniques is presented to provide the reader with a comprehensive understanding of the criteria that informed our selection of the appropriate measurement for our reputation system.

Dominance Relationship Based Reputation Measurement

The Dominance Relationship-Based Reputation (DRBR) method uses a ranking of services derived from ordinal preferences to measure the reputations qualitatively rather than use numbers with intensity (like averaging the ratings) [9].

Rank indicates how two services are related such that, for any two services ranked adjacently, the first dominates the second based on the dominance conditions.

The three main steps for the ranking construction are as follows.

1. **Determine the dominance relationship of each pair of services with given rules.**

   This step identifies the relative importance or influence within a given system. It can help understand how different services interact with one another and how they contribute to the overall reputation of the system.

   There are several ways of determining service dominance, including:

   - Network analysis involves using techniques from network science to analyse the structure of the relationships between different services within a system. This analysis can help identify patterns and trends in how services interact and can be used to infer the relative importance of other services.
   - User feedback: By collecting feedback from users about their services, it is possible to determine which services are most important to them and how they rank different services in terms of their overall reputation.
   - Expert evaluations: Members can ask experts in a particular field or domain to evaluate their work and the relative importance or influence of different services within a system. Users of the system can use their opinions to weigh the reputation scores of various services accordingly.

   A dominance relationship exists between two services if one service can complete a task that the other cannot. For example, if service A can complete a task that service B cannot, then A is considered dominant over B.

2. **Construct a Directed Acyclic Graph (DAG) based on the dominant relationships of service pairs.**

   The DAG represents the hierarchy of services, with the more dominant services at higher and the less dominant services at lower levels.

   Following identifying dominance relationships, the DAG is constructed by creating a vertex for each service and adding directed edges between pairs of services based on the dominance relationships. For example, if A is dominant over B, a directed edge would be added from A to B in the DAG.

   The resulting DAG represents the hierarchy of services, with the more dominant services at higher levels and the less dominant ones at lower levels. Users of the system can use this hierarchy to measure the reputation of each service based on the services it dominates and those that dominate it.

3. **Find a total order of vertices from the DAG and use the order of vertices as the eventual ranking of services.**

One approach is to use a topological sort algorithm to find the total order of the vertices in a DAG. A topological sort is an algorithm that takes a DAG as input and produces an entire order of its vertices as output.

The basic idea behind a topological sort is to choose a vertex with no incoming edges and add it to the total order. Then, remove the vertex and all its outgoing edges from the DAG. Repeat this process until there are no more vertices left in the DAG.

For example, suppose we have a DAG with the following vertices and edges:

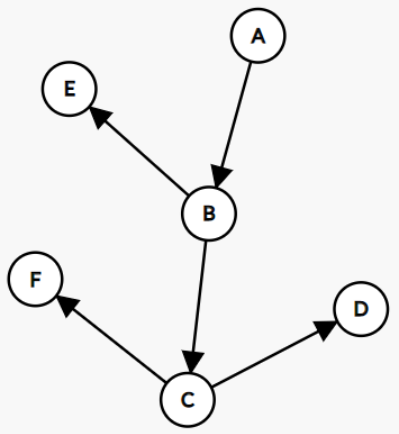


Figure 2.6: Directed Acyclic Graph Example

We can find the total order of the vertices in this DAG using the following steps:

(a) Choose a vertex with no incoming edges, which is A. Then, add it to the total order and remove it from the DAG.

(b) Choose the next vertex with no incoming edges, B in this case. Then, add it to the total order and remove it from the DAG.

(c) Choose the next vertex with no incoming edges, which is C. Then, add it to the total order and remove it from the DAG.

(d) Choose the next vertex with no incoming edges, which is E. Then, add it to the total order and remove it from the DAG.

(e) Choose the next vertex with no incoming edges, which in this case is F. Then, add it to the total order and remove it from the DAG.

(f) Choose the next vertex with no incoming edges, which in this case is D. Add it to the total order and remove it from the DAG.

The resulting total order is A, B, C, E, F, and D.

The total order of vertices in the DAG can be used as the eventual ranking of services. Services that appear earlier in the total order have higher reputations than those that appear later. In the example of Figure 2.6 using the topological sort, we conclude that A has the highest reputation and D has the least.

Alpha-Beta Model

The reputation model can rely on approaches previously implemented in a relevant project, such as PoSeID-on [29], which considers the Beta distribution [15]. The beta distribution assumes two parameters, α and β. The events have two possible outcomes: $\{x, y\}$, r and s are the numbers of observations of the outcome x and y, respectively.

$$\alpha = r + 1 \text{ and } \beta = s + 1,$$

where $r, s \geq 0$ and r and s represent positive and negative feedback, respectively.

The beta distribution function ($f(p|\alpha\beta)$) is expressed as follows:

$$f(p|\alpha\beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta+1}, \tag{2.1}$$

where $0 \leq p \leq 1$, $\alpha > 0$, $\beta > 0$.

As a result, the probability expectation value is as follows:

$$E(p) = \frac{\alpha}{\alpha + \beta}. \tag{2.2}$$

Therefore, the beta reputation provides a probabilistic value for the reputation.

We chose this model for the project's implementation, and Section 7.1 describes its validation process.

CORE

Community Reputation Mechanism (CORE) [23] is a decentralised reputation system that allows users to rate and review other users in a community, such as a peer-to-peer marketplace or social networking platform. Users can use it to assess other users' trustworthiness and reliability within a community, helping to build trust and facilitate more successful interactions.

In a CORE system, users can rate and review other users based on their experiences with them. These ratings and reviews are then aggregated and used to calculate a reputation score for each user, which trustors can use to assess the user's trustworthiness and reliability. Users with higher reputation scores are generally seen as more trustworthy, while trustors may view those with lower scores more cautiously.

Implementing CORE systems can be done using blockchain technology, which allows them to be decentralised and resistant to tampering or censorship. In addition, this blockchain implementation can help ensure the integrity and reliability of the reputation scores, making it more difficult for users to manipulate their ratings.

Overall, CORE helps build trust and facilitate successful interactions within online communities. We can apply it to a variety of different types of platforms and markets.

EigenTrust

EigenTrust [17] is a reputation mechanism to help establish trust among nodes in a P2P network. It was developed to ensure that nodes in a P2P network are behaving correctly and are not participating in malicious activities such as distributing spam or distributing copyrighted materials without permission.

In the EigenTrust system, each node maintains a reputation score that reflects the node's trustworthiness. This reputation score is calculated based on the ratings other nodes in the network have given the node in question. Nodes can provide positive or negative ratings to different nodes based on their observations of the node's behaviour.

The EigenTrust system uses an iterative process to calculate the reputation scores of nodes in the network. This process involves taking the ratings from other nodes and using them to compute a new reputation score for each node. This process is repeated until the reputation scores of the nodes stabilise.

One of the critical features of the EigenTrust system is that it allows nodes to give more weight to the ratings of specific other nodes based on their reputation scores. This allows the system to consider the relative trustworthiness of the nodes providing ratings, which can help ensure that the reputation scores are accurate and reliable.

The EigenTrust reputation mechanism is designed to help ensure that nodes in a P2P network behave trustworthy and provide a way for nodes to make informed decisions about which nodes they can trust.

REGRET

Reputation Mechanism for Generalised Trustworthiness Evaluation (REGRET) [32] is a decentralised reputation system that aims to measure the trustworthiness of entities (such as individuals, organisations, or machines) on the Internet. It is designed to be a flexible and scalable solution that members of the system can use in various contexts, including online marketplaces, peer-to-peer networks, and other decentralised systems.

The REGRET system is based on a decentralised network of nodes that collectively maintain a global reputation database. Each node in the network is responsible for maintaining a local copy of the database and participating in the consensus process that determines the reputation scores of entities.

In the REGRET system, reputation scores are calculated based on the feedback and interactions that entities have with other entities in the network. This feedback can be positive, negative, or neutral and is used to update the reputation scores of the entities involved. For example, if two entities engage in a successful transaction, their reputation scores may be increased. On the other hand, if an entity engages in fraudulent or malicious behaviour, its reputation score may be decreased.

The REGRET system also includes mechanisms for dispute resolution, such as a mediator system or a reputation court, to help resolve conflicts and ensure that reputation scores are accurate and fair.

The REGRET reputation mechanism is intended to provide a decentralised and transparent way to evaluate the trustworthiness of entities on the Internet, enabling users to make informed decisions about who they interact with and transact with online.

FIRE

The Fair, Incentivised, Reputable, and Engaging (FIRE) [13] reputation mechanism, an extension of REGRET, is a system for evaluating the reputation of entities within a community or network. It is designed to encourage positive behaviour and discourage negative behaviour within the community.

In a system with a FIRE reputation mechanism, users can earn reputation points for contributing positively to the community by posting helpful content or participating in discussions. However, they can also lose reputation points for engaging in harmful behaviour, such as spamming or harassing others. The reputation points determine a user's overall reputation within the community. The system can use this reputation to assess their access to certain features or privileges within the community.

The FIRE reputation mechanism is often used in online communities, such as forums or social networks, to help maintain a positive and respectful environment. However, it can also be used in other contexts, such as P2P networks or decentralised systems.

P-Grid

P-Grid [1] is a decentralised P2P system for distributing and sharing data in a distributed network. It allows users to search efficiently and access data within the network without needing a central server or index.

One key feature of P-Grid is its reputation mechanism, which is used to maintain the integrity and reliability of the network. "Reputation points" are the basis of the reputation mechanism, assigned to each peer in the network based on their behaviour. Peers who contribute valid data or resources to the network and behave responsibly are rewarded with high reputation points. At the same time, those who engage in malicious or disruptive behaviour are punished with low reputation points.

The reputation points are used to determine the trustworthiness and reliability of a peer, and they play a crucial role in routing data within the P-Grid network. Peers with high reputation points are likelier to be selected as intermediaries in routing data. In contrast, those with low reputation points may be excluded from the routing process. This helps to ensure that data is routed through reliable and trustworthy peers, improving the overall reliability and integrity of the network.

In addition to the reputation mechanism, P-Grid also uses other tools to maintain the integrity of the network, such as digital signatures and encryption. These measures help to prevent tampering with data and ensure that the data transmitted within the network is accurate and trustworthy.

PeerTrust

PeerTrust [37] is a reputation mechanism used to assess nodes' trustworthiness in a P2P network. It is based on the idea that a node's reputation is determined by the trustworthiness of the nodes it chooses to connect with and the trustworthiness of the nodes that connect with it.

In a P2P network, nodes may have different roles and responsibilities and may be more or less trustworthy depending on their behaviour. PeerTrust uses a reputation score to assess a node's trustworthiness based on its past behaviour and the other nodes with which it interacts. Other nodes in the network can use the reputation score of a node to decide whether to trust the node and engage in transactions or other exchanges with it.

There are various ways in which PeerTrust can be implemented, but one common approach is to use a decentralised database to store and update the reputation scores of nodes in the network. Other nodes in the network can access this database to retrieve the reputation scores of nodes they consider interacting with.

PeerTrust is a helpful tool for ensuring the integrity and reliability of a P2P network. It can help reduce the risk of fraud and other malicious behaviour by encouraging nodes to act trustworthy to maintain their reputation.

RateWeb

RateWeb [21] is a reputation mechanism that was proposed as a way to establish trust among web services. It is based on using a network of trusted agents to assess the reputation of web services and provide this information to users.

In the RateWeb system, web services are rated by trusted agents based on various factors such as their reliability, security, and performance. These ratings are then aggregated and used to calculate the overall reputation of the web service. Users can access this reputation information when choosing which web services to use, helping them to make more informed decisions about which services are trustworthy and reliable.

RateWeb was designed to address the problem of establishing trust in distributed systems, where it can be difficult for users to determine the trustworthiness of a particular service. By relying on a network of trusted agents to assess the reputation of web services, RateWeb aims to provide a more reliable and transparent method for establishing trust among web services.

Travos

Travos [35] is a proposed reputation mechanism that addresses the problem of inaccurate information sources on the internet. It is based on using trust and reputation scores to evaluate the quality and reliability of information sources. Similarly to Beta, Travos implements a probabilistic approach using the beta probability density function, as shown in Section 2.2.5.

The Travos reputation mechanism assigns a reputation score to each information source based on its history of providing accurate or inaccurate information. This reputation score is then used to determine the source's trustworthiness, with higher scores indicating a higher level of trust.

Travos uses a combination of explicit feedback from users and automatic evaluations of the accuracy of the information provided by the source to calculate reputation scores. Precise feedback can come in the form of ratings or reviews users provide. At the same time, the automatic evaluations can be based on factors such as the source's history of providing accurate information and the credibility of the sources it cites.

Travos is intended to be used in various contexts, including social media, news websites, and online forums, where the quality and reliability of information can be challenging to determine. It is designed to be a scalable and flexible solution that can adapt to the changing nature of the internet and the constantly evolving landscape of information sources.

XRep

The XRep [6] reputation mechanism is a method for evaluating the reliability of resources in a P2P network. It is based on using reputation scores to determine the trustworthiness of resources.

In the XRep system, each node maintains a reputation score for each resource it has encountered. The reputation score is updated based on the node's experience with the resource and the reputation scores of other nodes interacting with the resource.

The XRep system uses a distributed algorithm to calculate reputation scores, which allows it to operate in a decentralised manner without the need for a central authority. This makes it well-suited for P2P networks, where there may need to be a central authority to regulate resource reliability.

The XRep reputation mechanism is designed to help nodes in a P2P network make informed decisions about which resources to use by providing a way to assess the trustworthiness of those resources based on the collective experiences of other nodes.

Summary

This section provides a comprehensive overview of the reputation measurement techniques discussed thus far. Table 2.1 presents the measurement's name, its advantages, and the entities that most benefit from its use. This table summarises the information presented previously and aims to provide the reader with a clear and concise understanding of the available options for reputation measurement.

Name	**Advantages**	**Entities**
Dominance Relationship	Consider the relationships between entities in the system, making it more reflective of the social context of the system, allowing for a more accurate representation of an entity's reputation.	People
Alpha-Beta	Adapt the reputation based on the new information coming in and can be updated dynamically over time, allowing for a reputation measurement that adjusts to the entity's behaviour over time	Services, People, Objects
CORE	Allows for a transparent and fair evaluation of an entity's reputation by providing a clear and well-defined set of criteria that are used to determine the entity's reputation.	People, Objects
EigenTrust	Uses transitive trust, which is derived from the trust of others, to calculate an entity's reputation, allowing a more accurate representation of an entity's reputation as it captures how others perceive the entity in the community.	Services
REGRET	Uses reference trust, derived from the trust of a selected group of reference entities, to calculate an entity's reputation, allowing for a more accurate representation of an entity's reputation, as it captures how a specific group of trusted entities in the community perceives it.	Services, People
FIRE	Integration of fairness and incentives into the reputation management process, using a combination of explicit and implicit feedback mechanisms, aims to ensure that reputation values are fair and unbiased while providing incentives to agents to participate in the reputation management process and to provide accurate and honest feedback.	Services, People
P-Grid	Provides a scalable, self-organising, fault-tolerant, and efficient way to manage and search for data in P2P networks, where reputation-based trust management and high availability, fault-tolerance, and load balancing are the key features to make it robust.	Services, People

PeerTrust	Is a lightweight, flexible, scalable and privacy-preserving reputation management system that allows peers to make trust decisions based on the reputation of other peers and is suitable for the dynamic environment of P2P networks.	Services, People
RateWeb	Allows for a comprehensive, real-time, privacy-preserving web service reputation measurement. It supports multiple reputation dimensions and allows for a more detailed and nuanced view of the trustworthiness of a web service.	Services
Travos	Addresses the issue of uncertain and unreliable information sources by using probabilistic models and belief functions. It provides personalised reputation, multiple dimensions support, and scalability by using distributed computation, which makes it suitable for large-scale systems with uncertain and unreliable sources of information.	Services, People
XRep	Allows for the selection of reliable resources by assessing the reputation of the peers that provide those resources. It is a self-organising, scalable, flexible and privacy-preserving system that can be adapted to different applications with customisable trust models.	Services

Table 2.1: Reputation measurements summary

2.3 Visualisation of Data

Data visualisation creates graphical representations of data sets to facilitate data analysis and comprehension of trends in data. We can use visualisation in various fields, including business, science, and engineering. It is advantageous in reputation systems as it allows stakeholders to see patterns and trends in data that might not immediately appear from raw numbers.

Various frameworks and tools are available to visualise data, including bar charts, line graphs, scatter plots, and heat maps. These visualisations can represent different data types and highlight various aspects of the data set. For example, we might use a bar chart to compare the reputation of other entities, while a scatter plot could show the relationship between different variables.

In reputation systems, members can use data visualisation to track the reputation of individuals or organisations over time, identify patterns and trends in reputation data, and make informed decisions based on the data.

By visualisations, stakeholders can better understand how reputation is affected by different factors, such as the quality of products or services, customer satisfaction, and the overall reputation of the company or organisation.

Many other frameworks and tools are available for data visualisation, including open-source options like Matplotlib[2] and Seaborn[3] and proprietary tools like Tableau[4] and Qlik[5]. Each of these tools has its strengths and limitations, and the right choice will depend on the specific needs and goals of the reputation system.

2.3.1 Data Visualisation in Reputation Systems

There are many ways to implement data visualisation in a reputation system. In [34], an approach that serves as a generic mechanism adapted to specific application areas is proposed, giving an example of how the system can be implemented in an eCommerce data set from eBay[6].

Visualisation-based approach

The visualisation-based approach describes two building blocks: "models" and "visualisation and interaction techniques". For each block in the system, a unique conceptual design is proposed and created, which is not influenced or impacted by the raw data supplied to evaluate the block's reputation.

1. *Models*

   It is important to remember that reputation is context-dependent because referrals are created in a specific situation bound to a specific transaction. Consequently, a single referral that might be provided as a uni-variate rating (e.g. $r = \{-1, 0, 1\}$) only reflects an opinion conceived in a specific situation by a particular entity. Each referral is modelled in a multidimensional referral space S, where multiple context attributes $C = \{c_1, c_2, \ldots, c_n\}$ are considered, where each element c_i takes the value of the related set of context attributes. The multidimensional referral space S is then defined where each dimension matches a relevant context attribute:

   $$S = C_1 \cdot C_2 \cdot ... \cdot C_n \tag{2.3}$$

2. *Visualisation and interaction techniques*

   Depending on the raw data and application case, various techniques are conceivable. One example of a good technique for illustrating multidimensional data sets is parallel coordinates, where n axes can represent n dimensions, as seen in Figure 2.7.

[2] *Matplotlib*: `https://matplotlib.org/`
[3] *Seaborn*: `https://seaborn.pydata.org/`
[4] *Tableau*: `https://www.tableau.com/`
[5] *Qlik*: `https://www.qlik.com/us/`
[6] *eBay*: `https://www.ebay.com/`

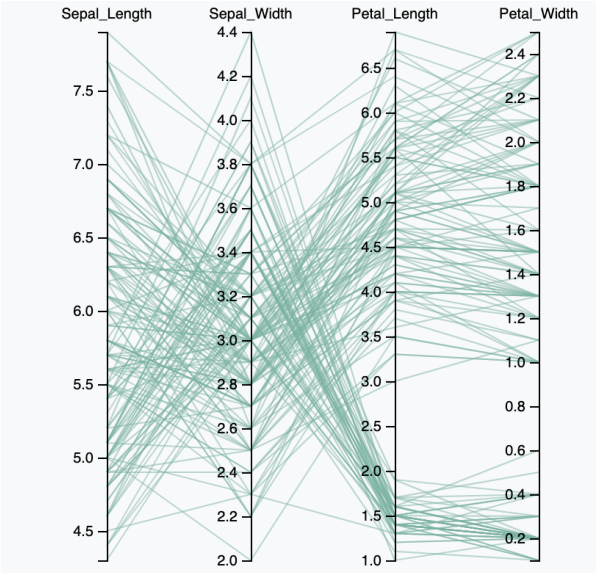


Figure 2.7: Parallel coordinates visualisation of flower data example

This technique can also explore correlations between single measurements and can be effectively supported through brushing, allowing the focus on a subset of the data visualised by highlighting the subset [11].

This project uses this approach for the data visualisation platform, for which the mockups are included in Section 4.6. We chose this approach for the project because of its scalability, interactivity and the ability to compare the values of different dimensions simultaneously, making it easier to understand their relationship.

Implementation with the eBay example

Now that we have defined the visualisation-based approach in Section 2.3.1, it can be implemented in real-world scenarios, in this case, using the eBay Germany data set [34].

The first step is to develop a model based on the raw data. Analysing and identifying the data set of a specific entity resulted in four context attribute sets C:

- Rating (C_1): The rating is the general seller evaluation the buyer conducted after a transaction. In this case, it can be a positive, neutral or negative rating.
- Time (C_2): The time describes the date the users gave the ratings. This is an essential context because old ratings might not be as relevant for the entity's reputation as newer ones.
- Price (C_3): The price describes the purchase price a transaction was coming off. This context is the leading example of the value imbalance problem, as a seller can build a high reputation by selling cheap products while cheating on expensive ones.

- Product category (C_4): The category a sold product was classified in (e.g. art, technology, music, etc.).

The result of the final referral S will be as follows: $S = C_1 \cdot C_2 \cdot C_3 \cdot C_4$ We can use the brushing technique to get a better glimpse of the reputation of a particular entity. For example, a seller can appear to have many positive ratings and is trustworthy at first sight. However, after using the brushing technique to denote negative ratings, we can verify that the seller presents many negative ratings when prices are higher, with these transactions occurring more recently than the ones with positive ratings. We have an example of the case previously mentioned where an entity takes advantage of having a better reputation with cheaper products but then presents terrible ratings on the more expensive ones.

2.3.2 Data Visualisation Frameworks

Data visualisation is integral to data analysis and helps understand and interpret data more effectively. It allows us to see patterns, trends, and relationships in the data that might take time to appear from looking at raw numbers. There are many different approaches to data visualisation, and choosing the right tool or framework depends on your project's specific needs and goals.

D3.js

Data-Driven Documents (D3) [5] is a JavaScript library commonly used to create dynamic, interactive data visualisations. It is particularly well-suited for working with large data sets, as it is designed to be efficient and scalable.

One of the critical features of D3.js is its data-driven approach, which means that the visualisations are generated based on data rather than being hardcoded. This makes it easy to update visualisations as the underlying data changes and allows users to create custom visualisations from scratch.

D3.js is also highly customisable, which means that users have a lot of control over the look and feel of their visualisations. It is compatible with modern web browsers and can be used with other libraries and frameworks, such as React[7] and Angular[8].

However, D3.js can be challenging to learn, especially for users with no previous experience with web development or data visualisation. It is also a complex tool, requiring a lot of code to create even simple visualisations.

Additionally, depending on the complexity of the visualisation and the amount of data being processed, D3.js visualisations can be resource-efficient and perform well on devices with limited resources.

[7] *React*: `https://reactjs.org/`
[8] *Angular*: `https://angular.io/`

Grafana

Grafana [33] is an open-source data visualisation and monitoring platform that allows users to create, visualise, and analyse data from various sources. It is primarily used for creating dashboards and visualising time series data, such as server, application, and device metrics. Some key features of Grafana include:

- Compatibility with various data sources: Grafana can connect to many data sources, including databases, cloud services, and APIs.
- Customisation: Grafana allows users to create custom dashboards and visualisations and offers a variety of customisable options, such as layout, colours, and labels.
- Alerting: Grafana can be configured to send alerts when certain conditions are met, such as when a metric exceeds a certain threshold.
- Collaboration: Grafana includes features that support collaboration, such as the ability to share dashboards with other users and to collaborate on dashboard design.

Grafana is widely used in monitoring and observability and is particularly popular for monitoring cloud infrastructure and applications. It is a powerful tool for visualising and analysing data, but it may have a steep learning curve for users new to data visualisation.

Plotly

Plotly[9] is a data visualisation library for creating interactive charts and graphs. It is available in several programming languages, including Python, R, and JavaScript. One of the main benefits of using Plotly is that it can generate visually appealing and interactive charts and plots with relatively little code.

Some of the features of Plotly include:

- A wide variety of chart types, including line plots, scatter plots, bar charts, pie charts, and more.
- The ability to add custom hover text and annotations to charts.
- The ability to zoom, pan, and hover over data points to see more detailed information.
- The ability to create interactive dashboards and reports.
- The ability to integrate with other libraries, such as NumPy and Pandas, for data manipulation and analysis.

[9] *Plotly*: `https://plotly.com/`

Plotly is often used for visualisations for scientific publications and presentations and for creating dashboards and reports for businesses and organisations. It is a valuable tool for anyone who needs to create professional-quality charts and graphs.

Tableau

Tableau [26] is a powerful data visualisation and business intelligence software that enables users to create interactive charts, dashboards, and reports easily. It is designed to help users quickly and easily visualise and understand their data to make informed decisions.

Some of the features of Tableau include

- A wide variety of chart types, including bar charts, line charts, scatter plots, maps, and more.
- Connecting to various data sources, including Excel, CSV, and databases.
- Manipulating and analysing data using drag-and-drop functionality and built-in calculations.
- Creating interactive dashboards and reports that users can share with others.
- Customising the appearance of charts and dashboards using various formatting options.

Businesses, organisations, and individuals often use Tableau to create visualisations for presentations, reports, and dashboards. It is a popular tool among data analysts, business analysts, and data scientists.

React-vis

React-vis[10] is a JavaScript data visualisation library for creating interactive charts and graphs. It is built on top of the React library, a popular library for building user interfaces in JavaScript.

Some of the features of React-vis include:

- A wide variety of chart types, including bar charts, line charts, scatter plots, and more
- Customising the appearance of charts using a variety of styling options
- Adding custom hover text and annotations to charts

[10] *React-vis*: `https://uber.github.io/react-vis/`

- The ability to zoom, pan, and hover over data points to see more detailed information
- Integration with other libraries, such as D3, for data manipulation and analysis

React-vis is often used for creating visualisations for web applications and websites. It is a valuable tool for creating interactive and visually appealing charts and graphs in JavaScript.

Approaches Comparison

This section presents a comprehensive overview of the data visualisation frameworks discussed in the preceding sections. In addition, a table (Table 2.2) is also provided, including the framework's name, key features and advantages. This summary aims to provide the reader with a clear and concise understanding of the available options for data visualisation. Furthermore, it should aid in making an informed decision regarding the appropriate framework for the particular use case.

Name	Features & Advantages
D3.js	Dynamic data binding; support for multiple data types, such as CSV or JSON; large and growing ecosystem with multiple plugins and libraries available. The main reasons why this framework was chosen were its **diverse capabilities of creating different types of charts** and the **ability to implement it with other frameworks**, in the case of this project, React (see Section 5.2 for an explanation of the choice of this framework).
Grafana	Customisable dashboards, allowing various visualisation options, including graphs, tables, and gauges; massive amount of plugins and community support; ability to import and export dashboards as JSON files. This platform was not selected for the implementation of the data visualisation platform as it **could not create dashboards using parallel coordinates or Sankey diagrams**, despite having plugins for the database and message queue utilised in the system (Cassandra and RabbitMQ).
Plotly	Flexible, and easy to use, having the ability to support different programming languages, such as Python and JavaScript. This platform was not selected due to its **inadequate performance in managing and processing substantial amounts of data**.
Tableau	Easy to use, providing an intuitive drag-and-drop interface; seamless integration with different data source types, such as Excel, CSV or SQL. The primary factor that led to the decision not to adopt this platform was its **relatively high cost compared to alternative frameworks**.

React-vis	Flexible and customisable, having a wide range of visualisation types, including line charts, scatter plots, and area charts; high-performance, able to handle huge amounts of data. The primary factors that led to the decision not to adopt this library were the **limited selection of chart types available** and its **requirement for use within a React environment**. While the platform is being developed using React, a visualisation framework change would also be necessary if issues arise or a decision is made to transition away from React.

Table 2.2: Data visualisation framework summary

2.4 Summary

This section aims to summarise the background and related work chapter and presents a synopsis of the articles read and analysed as part of this research project. Table 2.3 lists the articles, briefly overview their main points and findings and shows how valid certain documents were for this project. This summary aims to provide a concise overview of the key insights and conclusions drawn from the articles and any notable differences or similarities between them by presenting this information clearly and organised.

It is expected that this chapter will provide the reader with a strong foundation of theoretical knowledge that will enable them to understand the context of the subsequent chapters.

Description	Keywords	Year
Reputation System [30] Reputation systems enable users to assess the trustworthiness and reliability of other users, organisations, or systems to make informed decisions about interactions. Challenges in design include ensuring accuracy and fairness while addressing privacy, security, and scalability. This document has helped to give us an overview of reputation systems and how we can build trust through them.	Reputation, Entities, Feedback, Trust, Behaviour	2000

Reputation System: A survey and taxonomy [12] Design and implementation of reputation systems must consider challenges such as accuracy, fairness, privacy, security, and scalability. Evaluation methods and metrics have been developed to assess their performance and effectiveness. This paper helped us to see how reputation systems are used in real-life examples (e.g. eBay and Stackoverflow) and the different dimensions we can use for this type of system, such as the examples in this project, entities, governance and data ageing.	Reputation System, Survey. Taxonomy, Feedback, Dimensions, Trustor, Trustee, Recommender	2015
Trust and Reputation Systems [14] Trust and reputation systems facilitate online interactions by allowing users to assess the trustworthiness and reliability of others. Centralised systems use a single authority to manage reputation information, while decentralised systems use P2P networks. These systems must consider accuracy, reliability, privacy, security, and scalability. This document helped us to get an overview of how reputation system architecture works, allowing us to see the advantages and disadvantages of each one and choose the most suitable one for this project.	Reputation, Trust, Centralised, Distributed, Peer-to-Peer	2007
Trust and Reputation Management [20] Trust and reputation management systems aim to provide a way for users to evaluate the trustworthiness and reliability of other users, organisations, or systems to facilitate online interactions and transactions. Factors affecting trust and reputation include the quality of products or services, the system's performance, and user behaviour.	Rankings, Trust, Reputation, Attack-Resilient	2010
Reputation Measurement for Online Services Based on Dominance Relationships [9] Dominance graph represents relationships between users as nodes and edges, with reputation determined by the position in the graph. Factors that affect dominance relationships include product or service quality, performance, and behaviour. This article helped us understand the dominance model and how it can be used in reputation systems.	Reputation, Dominance Relationships, Ranking, Directed Acyclic Graph, Online Services	2021

A Privacy System to Consult Public Institutions Records [16] Reputation and the notion of trust are directly related as an entity can build trust based on another's reputation. Trusting an entity in an online service is risky because no past interactions exist, and the reputation aims to reduce this risk. This master's thesis helped us learn about different reputation models and choose the most appropriate one for this project - the Alpha-Beta model.	Reputation Systems, Trust, Risk, Personal Identifiable Information, Privacy	2020
Visualising Transaction Context in Trust and Reputation System [34] The proposed approach uses a visual interface to display transaction context, including relationships, history, and reputation ratings. It offers improved understanding and accuracy for trust and reputation systems. This paper helped us choose the most suitable components to implement in the visualisation platform, such as parallel coordinate charts and brushing.	Reputation, Transaction Context, Visual Analytics, Visualisation, Parallel Coordinates	2014
Visual framework for big data in d3.js [5] The framework has three components: visual data model, representation, and interaction. It represents data structure and relationships, visualises data, and handles large data sets. It is flexible and customisable for various applications. This document helped us understand how D3.js, the project's chosen tool for the graphical implementation of the visualisation platform, would handle large amounts of data.	D3.js, Visualisation, Data Warehouse	2014
Interactive Visual Analytics on Big Data: Tableau vs D3.js [26] Comparison of Tableau and D3.js when used extensively in data analysis. Different weaknesses and strengths of each framework, including their capabilities, performance, and usability.	Big Data, Big Data Visualisation, Visualisation Tools, Tableau, D3.js.	2016

Table 2.3: Related work summary

Chapter 3

Research Objectives & Approach

This chapter provides a clear and concise description of the research objectives and the approaches we will use to achieve these objectives. First, Section 3.1 outlines the dissertation's specific goals or objectives. Then, in Section 3.2, we discuss the plans to complete each goal and measures taken to handle obstacles encountered.

3.1 Objectives

The primary objective of this dissertation is to investigate and understand the factors contributing to the reputation of the various components/services in the ARCADIAN-IoT project. We implement several methods and approaches to achieve this objective, including streaming reputation information and developing a platform for visualising reputation data.

Streaming reputation information is essential to our research, allowing us to track and monitor various entities' reputations continuously. We can efficiently process and analyse the incoming reputation data in real time using data streams. This enables us to identify trends and patterns and calculate each event's reputation score.

In addition to streaming the reputation data, we will also implement a platform for visualising the reputation data. This platform will allow us to present the data clearly and meaningfully, enabling us to identify critical trends and patterns that may not be immediately apparent from raw data alone through graphical analysis. The platform will also provide valuable insights into the factors influencing the reputation of the entities we are studying, helping us better understand the underlying drivers of reputation.

To reach our main objective, we defined the following goals:

1. **Research reputation measurement methods**

   Chapter 2, specifically Section 2.2.5, documents how we can determine a reputation value for entities in a system and how the system can compare

them to determine which has the highest reputation.

2. **Research how users can visualise the reputation data and through which frameworks**

   According to Section 2.3, it is possible to visually represent the reputation of a specific system using a specified method. Therefore, we can use several frameworks to implement the data visualisation platform to achieve this objective.

3. **Receive events from heterogeneous components with distinct functions**

   To effectively receive the events generated by the various partners involved in the ARCADIAN-IoT project, we have designed the system to use a RabbitMQ exchange for each component. This approach allows the system to efficiently process and manage the incoming event data from each partner, ensuring that it can track each partner's reputation and status within the ARCADIAN-IoT project.

4. **Analyse the data received to calculate the reputation score**

   Upon receiving the data from the partners involved in the project, we utilise data streams to calculate the reputation score for each event received. To accurately determine the reputation score, we must identify the relevant information that needs input into the model, such as whether the event has a positive or negative rating. Once the system has calculated the reputation score, using one or several reputation models based on the entity type, we store all of the reputation information in the database for future reference and analysis. Archiving the reputation information allows us to track the reputation of each event over time and make informed decisions about how to respond to it.

5. **Implement the data visualisation platform**

   We must access the database and retrieve the required reputation information to design the data visualisation platform. Getting the essential reputation information allows us to comprehensively understand the data we are working with and make informed decisions about visualising it meaningfully and effectively. For more information on how we design the data visualisation platform, please refer to Chapter 4, where we describe our approach and the specific tools and techniques we use.

3.2 Approach

This section presents an overview of the methodology employed, the decisions made during the development process, any challenges encountered and the approach taken to address them.

3.2.1 Research & Development Methodology

The purpose of this section is to outline the vision and approach we used for researching and developing a management framework, including the methods and systems we use.

Firstly, understanding state-of-the-art reputation systems, methods for calculating entity reputation within a system, and visualising reputation data was accomplished to gain insights into potential features and popular functionalities for our work and implementation techniques. Then, with the use of search engines such as ResearchGate, IEEE Xplore, and Mendeley, we searched for articles, books and journal publications related to reputation systems (e.g., "reputation score calculation") and how to visualise their data (e.g., "reputation system visualisation").

We thoroughly evaluated each researched framework to determine the most suitable framework for implementing the data visualisation platform. This evaluation included reviewing the documentation of each framework to assess its feasibility for creating the desired platform and viewing tutorials that demonstrated how to implement simpler platforms that were similar to the chosen venue, albeit less complex. This analysis identified the best framework for implementing the data visualisation platform.

After this documentation, we proposed a ranking system to define priorities in the requirements to implement the reputation system and its data visualisation platform. This ranking followed the MoSCoW method to assist us in finding the top requirements that we prioritise. The requirements list was compiled by gathering information from previous artefacts, e.g., the ARCADIAN-IoT project deliveries.

After careful consideration, we decided the development methodology for this project would be an **iterative waterfall model**, combining the structure and predictability of the traditional waterfall methodology with the flexibility and adaptability of an iterative process. The project is divided into minor, distinct phases, allowing for modification and adjustment based on feedback and results. This approach enables us to respond to changing circumstances while maintaining a clear roadmap for the project.

3.2.2 Planning

This section aims to present the plan and activities performed during the first semester and document the following steps to take in the second semester.

We created a Gantt Diagram (Figures 3.1, 3.2, and 3.3) to aid us in defining deadlines and visualising the project's timeline. Each time slot allows us to allocate a certain amount of time towards a specific objective without causing the project to fail.

For the **first semester**, the planning involved usual tasks for a dissertation and specific tasks related to the project.

1. **Dissertation Plan Rewriting**: The dissertation plan was revised to include the development of a data visualisation platform.

2. **Reputation Systems Research**: We researched and gathered background information on reputation systems, their various architectures, and methods for measuring members' reputations within a system.

3. **Data Visualisation Research**: We conducted research and obtained background knowledge on data visualisation, methods for displaying the reputation of system members, and various frameworks we can use to achieve this.

4. **Alpha-Beta Model Testing**: Testing the alpha-beta model to check whether this model fits the project and can calculate the reputation of the different entities when receiving new events.

5. **Project Events Analysis**: Analysis of the events to verify which parameters received by the project partners are relevant to calculating the reputation of an entity.

6. **RabbitMQ Testing**: Testing of RabbitMQ to know how information between producer and consumer is sent and how streams can send reputation information.

7. **Mockups**: The development of mockups for the data visualisation platform.

8. **Functional Requirements Gathering**: Gathering functional requirements for the reputation system and the data visualisation platform.

9. **Non-Functional Requirements Gathering**: Gathering non-functional requirements for the reputation system and the data visualisation platform.

10. **Use Cases**: Realisation of the use cases in which we could use the reputation system in real-life circumstances.

11. **Risk Analysis**: Analysis of possible risks for the project, creating a mitigation plan for the ones with higher severity.

12. **Preliminary Architecture**: Perform and document architectural diagrams for a prior architecture.

13. **Intermediate Report Writing**: Aggregation of all the information gathered to write the intermediate report.

14. **Intermediate Defence Preparation**: Preparation of the intermediate defence with a presentation of the essential contents.

15. **Reception of Partners' Events**: Implementation for receiving events from different project partners. The performance of this task extends into the second semester.

Figure 3.1 visually depicts the interactions between each first-semester milestone. The colour-coding system identifies the type of task, with purple representing planning tasks, yellow representing background and related work, orange representing modelling tasks, red representing requirements elicitation, green representing the project architecture, grey representing report writing, and blue representing implementation tasks.

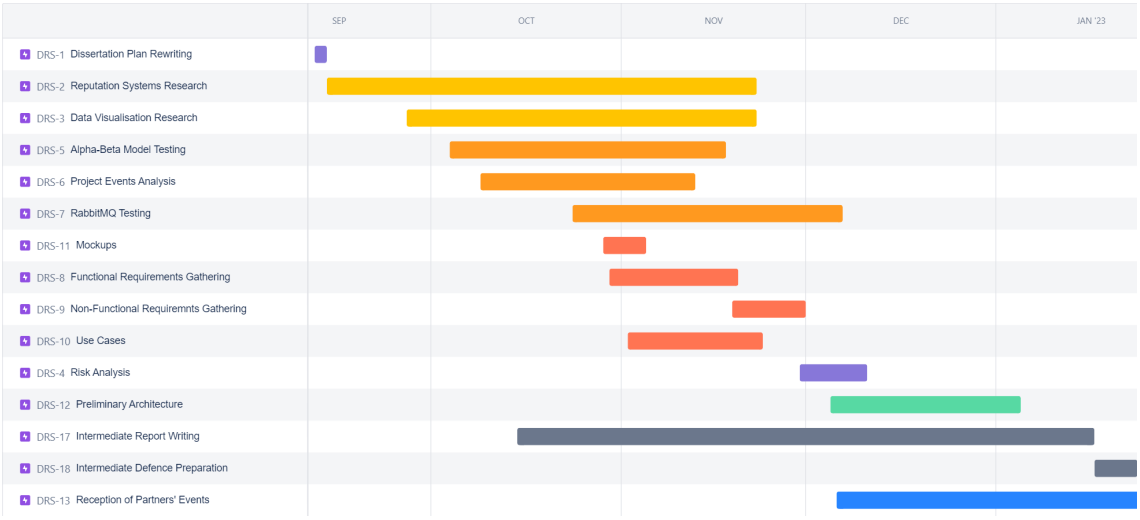


Figure 3.1: Gantt Diagram of the first semester

In the **second semester**, the planning focuses on implementing the reputation system and the data visualisation platform, following the iterative waterfall model. The reputation system and data visualisation platform are integral components of the project, and careful planning and execution ensure their successful implementation.

1. **Document Corrections**: Corrections of the intermediate report based on the feedback provided in the intermediate defence.

2. **Reception of Partners' Events**: As mentioned in the first-semester planning, the aim is to implement how to receive events from different project partners.

3. **Analysis of Reputation Data & Reputation Calculation**: After receiving events from the project partners, we can analyse the event's data and check which is vital for the following steps. When we gather all the essential information for the reputation calculation, the score can be calculated according to the entity in the data and stored in the database.

4. **Platform Frontend Development**: Implementation of the frontend of the data visualisation platform.

5. **Receive Events From Database and Display**: When the frontend finishes and the database has information on the system entity's reputation, the data visualisation platform displays this information.

6. **Quality Attributes Testing**: After implementing the essential requirements of the reputation system and the data visualisation platform, the quality attributes presented in Section 4.3 are tested to verify that they meet the proposed conditions.

7. **Final Report Writing**: Aggregate all the information gathered to write the final report. This task also includes the parallel **writing of the scientific article** mentioned previously in Section 1.2.

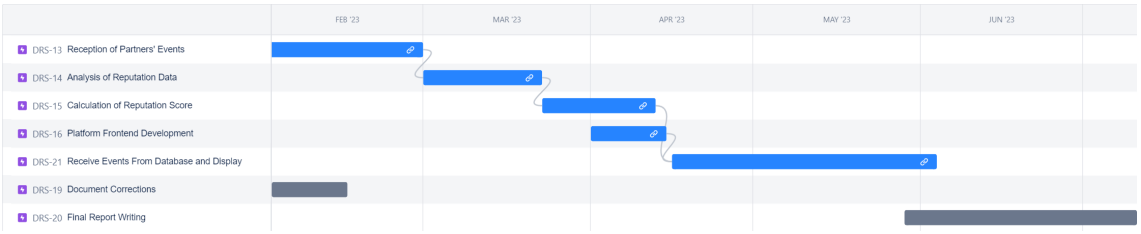
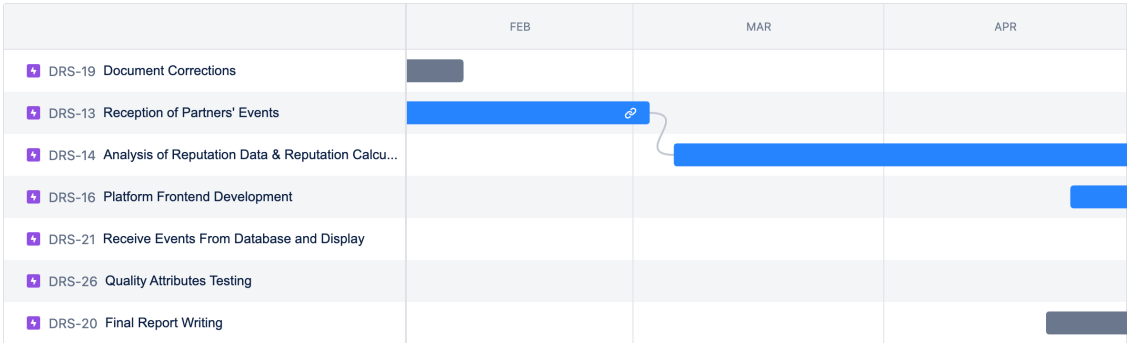
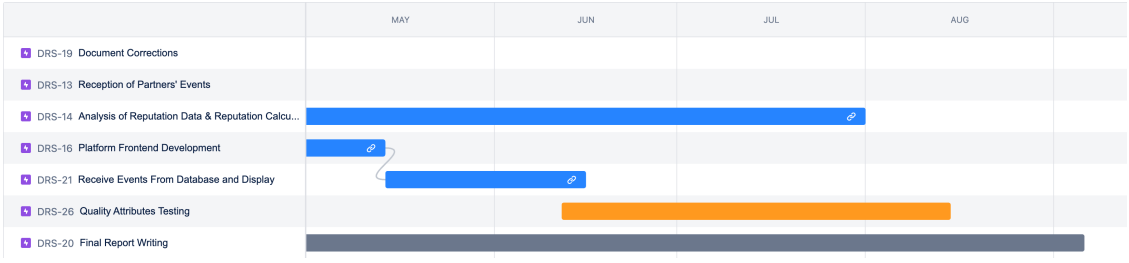


Figure 3.2: Gantt Diagram expected for the second semester

Figure 3.2 visually depicts the expected interactions between each second-semester milestone. The colour-coding system, like in the first-semester figure, identifies the type of task, with grey representing report writing and blue designating implementation tasks.



(a) First three months



(b) Remainder four months

Figure 3.3: Gantt Diagram of the second semester

As depicted in Figure 3.3, the Gantt Chart of the second semester of the project is presented, wherein notable disparities about task deadlines become evident. The divergence in task timelines is attributed to the manifestation of certain risks,

elucidated comprehensively in Section 3.2.3. The colour-coding system is similar to the one in Figure 3.2, only adding the tests of the qualities attributes in orange.

3.2.3 Risks

In this section, we present the identified risks for the project and provide a mitigation plan for those with higher severity. By identifying and proactively addressing risks, we can minimise their potential impact and increase the likelihood of project success. For the risks with high seriousness, we outline the identified risks and the steps we take to mitigate their effect on the project.

Table 3.2.3 presents the options for risk classification [31]. The formula used to determine the risk severity is described below in Equation 3.1 below.

$$Severity = Impact \cdot Likelihood \tag{3.1}$$

Attributes	Rating	Description
Impact	1	A risk with a low impact that would have minimal or minor consequences if it were to occur.
	2	A risk with a medium impact that could have a moderate effect on a project if it were to happen.
	3	A risk with a high impact that could have a significant or severe effect on a project if it were to happen.
Likelihood	1	A risk with a low probability and unlikely to happen.
	2	A risk with a medium probability with a moderate chance of happening.
	3	A risk with a high likelihood that has a high probability of occurring.
Severity	1-2	A low severity risk with a low possibility of happening and would have minimal or minor consequences if it did happen. It may not significantly impact the project's timeline, budget, or overall success.
	3-5	A risk with medium severity that has a moderate chance of occurring and could have a moderate impact on the project. It may require more attention and resources than a low-severity risk but less than a high-severity risk. This could include impacts on the project timeline, budget, or overall success.
	6-9	A risk with high severity that has a high likelihood of occurring and could have a significant or severe impact on the project if it did happen. Addressing and mitigating the potential impact may require considerable attention and resources. The high severity impact can potentially jeopardise the project timeline, budget, or overall success. It is crucial to promptly identify and address high-severity risks to minimise their potential impact on the project.

Table 3.1: Risk classification caption

ID	Description	Impact	Likelihood	Severity
R1	Development takes more than expected, which may affect the quality or content of the final delivery	3	2	6
R2	Elements of the team will become demotivated or burned out, being less productive	2	1	2
R3	The primary author isn't an excellent frontend developer, and the platform isn't intuitive enough	1	2	2
R4	The primary author does not have a background in the different software used on the project, such as RabbitMQ and Java Spark	2	2	4
R5	The project partners don't give the needed information for the data streaming in the required time, thus delaying the analysis of data	3	1	3
R6	The framework development may not meet the required specifications due to poor architectural choices.	3	2	6

Table 3.2: Risk Identification

Table 3.2.3 presents the identified project risks, with those with a higher severity indicated in red. The mitigation plan for these risks is defined as follows:

- R1: If the development takes longer than expected, the mitigation plan for this risk relies on the requirement listing prioritisation using the MoSCoW method, see Section 4.4. Using this classification, we should not consider the less prioritised needs and focus on those with a must-have category.
- R6: If the architectural choices were poor (e.g., RabbitMQ is not scalable enough), we should thoroughly review the options to identify any potential issues or areas for improvement, seeking advice from more experienced professionals. Another plan we can consider is implementing a prototyping testing phase to identify any problems or areas for improvement before proceeding with full development.

Occurred Risks

For the identified risks, **R1**, **R4** and **R5** occurred, but we only had a mitigation plan for R1.

Concerning risk **R1**, the prescribed mitigation plan was diligently executed. In this context, an emphasis was placed on prioritising exigent requirements, specifically within the ambit of the reputation system. Of particular significance were the functions of event reception and reputation calculation. The deliberate decision to extend the dissertation delivery timeline to September gave us an extended period to refine some requirements that had not been previously prioritised. Moreover, it identified and rectified issues pertinent to the reputation system and the data visualisation platform.

Concerning risk **R4**, its materialisation transpired during the implementation phase. Pertinent uncertainties emerged surrounding the feasibility of realising specific functionalities via Java Spark. For instance, there was a notable inquiry into how information derived from RabbitMQ could be effectively stored in Cassandra after data analysis. This difficulty was further compounded by concerns relating to security, particularly the integration of RabbitMQ.

In response to these challenges, a comprehensive and rigorous investigation was conducted, directed at the domains where the issues were manifest. As a result of this intensive research endeavour, a substantial proportion of the predicaments encountered were systematically addressed and subsequently resolved.

Regarding risk **R5**, its occurrence was manifested during the intermediary phase of implementation, namely in analysing events and calculating reputation. Some data was missing from exchanges defined in the system, but there was no information from the project partners about what data would be sent to these respective exchanges. In some cases, new exchanges emerged that had not been initially defined.

To address this challenge, the solution was not to be dependent on sending the information from the missing exchanges, as, fortunately, the system already had support for several exchanges, and some partners sent the information as requested. After analysing the already known exchanges, if there was some information on the missing exchanges' data, then support for these was made in the system.

Chapter 4

Requirements Elicitation

This chapter describes the requirements that need to be considered for the data visualisation platform and reputation system. These include functional requirements, described in Section 4.2, which specify the specific actions and capabilities that the platform and system should be able to perform, and non-functional requirements, described in Section 4.3, which concern the overall quality and performance of the platform and system.

The section then proceeds to list the requirements using the MoSCoW method in Section 4.4, which allows them to be prioritised according to their importance and relevance to the project. Next, the use cases, based on utilisation scenarios presented by an ARCADIAN-IoT deliverable, and mockups of the platform are presented in Sections 4.5 and 4.6, respectively, providing a visual representation of how users will use them and how they will look and feel to the user.

To provide a clear and comprehensive understanding of the reputation system's functional requirements and use cases, Section 4.1 presents various fields to illustrate the diverse range of contexts and scenarios in which these components will be used.

4.1 Domains' Description

This section aims to describe overall stories related to the IoT domains to anticipate and introduce the context for the functional requirements of the reputation system and the use cases specification. With its focus on business applications, this description clearly explains the solutions' functionalities and the security, trust, and privacy management challenges they must address [36].

4.1.1 Domain A: Emergency and vigilance using drones and IoT

This domain focuses on using IoT devices, specifically drones, in citizen-centred urban vigilance services.

A high-level scenario can feature a young lady, Ana, going home after dinner with friends. Using the ARCADIAN-IoT Drone Guard Angel (DGA) app on her smartphone, Ana requests vigilance services to escort her home. The service is available in her city. To be registered and recognised by the DGA, Ana has supplied some personal data in the registration phase, like name, address and photos. To request a service, she must provide the initial and final locations to guarantee the service is available in both spots.

Figure 4.1: DGA entities involved

After receiving the service request with Ana's data, such as her location and identification, a drone parker in a specific place in the neighbourhood lifts off and arrives near her. The first thing it does is to validate the user through multiple criteria, which include the recognition of Ana's smartphone and her physical characteristics. After successfully identifying her, the drone is ready to guard Ana back home.

Ana starts walking home, and the DGA is following her, aware of the surroundings to detect any threat signal, such as rapid movements towards Ana, high-speed vehicles or objects. Suppose the drone sees something abnormal, like a robbery attempt. In that case, it can start an appropriate manoeuvre to scare or demotivate the robbers (blinking lights, emitting sounds, etc.) while it calls for rescue. A medical team is called to the place if the drone detects any injuries. While the rescue team is on its way, some details can already be sent and collected by the camera, microphone and appropriate sensors, like GPS, to give precise locations and provide the incident characteristics.

4.1.2 Domain B: Medical IoT

An example of a Medical IoT (MIoT) scenario is as follows: Maria, a 5-year-old girl, had her tumour removed in Ecuador, and she is currently receiving treatment in Madrid. It is a very rare cerebral sarcoma with a poor prognosis associated with DICER, a rare genetic disorder predisposing individuals to multiple cancer types. Pediatric radio-oncology uses proton medical devices that generate

intensive radiotherapy during the required sessions by sending large amounts of proton to the brain tumour. As a result, almost all patients receive radiotherapy and chemotherapy, but each undergoes different treatments and receives personal treatment.

Considering the demanding volume of treatment sessions, reducing the number of consulting sessions to assess the patient's well-being is very beneficial. For this purpose, a telemonitoring system is well accepted by the medical staff, a team of doctors and nurses, and the patients. Both see the solution as more comfortable and able to automatically provide an evolutionary record of the patient's status and potentially get the medical staff's attention to relevant readings.

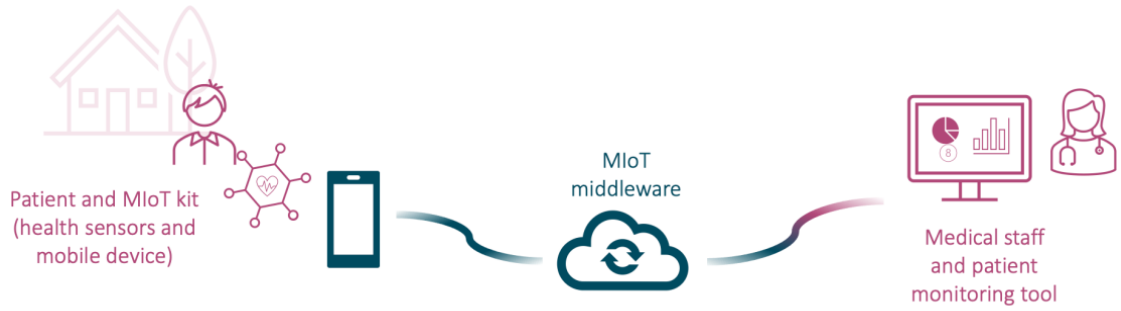


Figure 4.2: MIoT entities involved

To be effective, MIoT needs to be able to monitor patients considering a treatment protocol, such as reading frequency, medication, and other medical recommendations. Therefore, it needs to collect, store, and present the evolution of vital signs in the cardiac area, the heart rate, temperature, SpO_2 (oxygen saturation in the blood) and blood pressure, captured with the medical sensors and timely provide alerts for medical decision support. In addition to these parameters, it should be possible for the patient to enter perceived symptoms in a mobile app, such as levels of fatigue, sweat, diarrhoea, or others that can describe symptom intensity.

The solution for satisfying the requirement will utilise an MIoT kit, which includes medical sensors and a smartphone. The hospital provides this kit to patients. The answer will also have an MIoT middleware service to distribute patient data and securely generate health alerts. A monitoring tool for the medical staff to check the patient's well-being and alerts and change the monitoring protocol.

4.2 Functional Requirements

This section outlines the functional requirements for the reputation system and the data visualisation platform. Each requirement will include a user story, the related domains, the scope, any preconditions the system must meet, any postconditions must be satisfied, and the success scenario must be achieved. We can

ensure that our system meets the necessary functionalities and operates effectively by detailing these requirements clearly and thoroughly.

Reputation System Functional Requirements

RS1. *Receive events from components of the project partners*

- **User Story**: As an authorised member, I want to seamlessly receive events for entities from the components of the project partners in the reputation system so that I can have a comprehensive view of the entities' events.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The project partners can send their entities' events to the reputation system.
- **Requirement post-conditions**: The reputation system has access to all the events' information.
- **Success Scenario**:
  (a) The components from project partners are provided with a message queue endpoint (RabbitMQ) to submit events related to the entities they are responsible for.
  (b) The partners can authenticate themselves using credentials with a connection through exchanges.
  (c) The partners submit events to the reputation system in a specified format (JSON objects) containing information about the event and the entity it pertains to.
  (d) The reputation system processes the incoming events and updates the entities' reputation scores accordingly.
  (e) The partners can handle the errors or issues that may occur during the event submission process, and they are notified of any errors and the reason behind them.

RS2. *Select essential information to be stored*

- **User Story**: As an authorised member, I want the reputation system to automatically select and store the essential information from the events received in the database to have a clear and concise view of the entity's reputation.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The reputation system must have received information about the events from the different components.
- **Requirement post-conditions**: N/A
- **Success Scenario**:

(a) Project partners send events with various data.
(b) The reputation system receives the events from the partners.
(c) The reputation system selects only the necessary information to save.
(d) The reputation system stores the previously selected information in the database.

RS3. *Calculate the entity's reputation score*

- **User Story**: As an authorised member, I want the reputation system to update an entity's reputation score upon receiving a new event so that I always have the most up-to-date reputation information. This calculation should use the most appropriate reputation model for a specific entity. This way, I'll be able to quickly understand the impact of new events on the entity's reputation and make informed decisions based on the most current data.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The reputation system has obtained information regarding the events from the various partners and has registered the corresponding entities.
- **Requirement post-conditions**: The reputation system will store the reputation information of the entity that has calculated the reputation score in the database.
- **Success Scenario**:
  (a) The reputation system queries the database to check if the entity already exists through the ID or name.
  (b) If the entity does not exist, it initialises the values required for reputation calculation.
  (c) Reputation score is calculated/updated based on the received event rating using the reputation measurement model chosen for the project.
  (d) Storage of the new reputation data in the database.

RS4. *Trustable Storage Mechanism*

- **User Story**: As a project partner, I want the reputation system to have robust and trustworthy mechanisms for storing the reputation of entities and their associated events so that it guarantees no information leaks.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: A trustable storage mechanism is in place and provides some Application Programming Interface (API) to enable the storage of reputation information.
- **Requirement post-conditions**: N/A

- **Success Scenario**:
  (a) Choosing a secure storage solution that meets industry data encryption and security standards.
  (b) Implementing access controls ensures that only authorised personnel can access and update the stored data.
  (c) Regularly backing up the stored data to ensure that it can be recovered in the event of a disaster or data loss.
  (d) Continuously monitoring the storage system for any security breaches or unauthorised access attempts and taking appropriate action to address any issues.
  (e) Conducting regular security audits and penetration testing to identify and address any vulnerabilities in the storage system.
  (f) Keeping the storage system and its software updated with the latest security patches and upgrades.

Data Visualisation Platform Functional Requirements

VP1. *Homepage Lists the Different Entities*

- **User Story**: As a platform user, I want to see a list of all the entities from which the homepage receives reputation information to understand where the reputation information is coming from easily.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The data streams with the reputation information must be working for this information to appear on the platform
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) Creating a homepage that displays a list of entities, including the ID/name, the entity type and the current reputation score for each entity.
  (b) Retrieving the reputation information for each entity from the reputation system's database and displaying it on the homepage in a clear and easy-to-understand format.
  (c) Implementing a pagination mechanism allows users to view a certain number of entities at a time on the homepage and navigate through the pages.
  (d) Continuously monitoring and updating the list of entities and their reputation information to ensure that it remains accurate and up-to-date.

VP2. *Filtering of Entities*

- **User Story**: As a user of the platform, I want to be able to see a list of all the entities from which the homepage receives reputation information and to filter them by reputation score, date of events received in the reputation system and search for a specific type of entity, so that I can easily understand where the reputation information is coming from and quickly identify and assess the entities reputations that are most relevant to me. Additionally, I want to be able to search for a specific type of entity so that I can easily find the entities that are most relevant to me.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The data streams with the reputation information must work for this information to appear on the platform.
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) Implementing a filtering system on the homepage that allows users to sort the entities by reputation score (highest to lowest or lowest to highest) or by date of events received in the reputation system (most recent or oldest).
  (b) Developing a search bar on the homepage that allows users to search for a specific entity type.
  (c) Ensuring that the filtering and search options are user-friendly and easy to use.
  (d) Continuously monitoring and updating the filtering and search functionality to ensure that it remains accurate and reliable.

VP3. *Entity Page with the Entity's History*

- **User Story**: As a user of the platform, I want to be able to view detailed reputation information for each entity on its page so that I can fully understand the entity's reputation. This should include information on the type of events that have affected the entity's reputation, the variation of the reputation over time, and the current reputation score.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The data streams with the reputation information must work for this information to appear on the platform, and the entity must interact with the system
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) Creating an entity's page for each entity in the reputation system that displays the entity's ID/name, type, and current reputation score.

(b) Retrieving the reputation information for the entity from the reputation system's database, including the type of events that have affected the entity's reputation, the variation of reputation over time, and the current reputation score.

(c) Developing a user-friendly interface that displays reputation information in a clear and easy-to-understand format using a parallel coordinate chart.

(d) Continuously monitoring and updating the entity's page to ensure the reputation information remains accurate and up-to-date.

VP4. *Entity Page with the Events Additional Information*

- **User Story**: As a platform user, I want to view a list of justifications for all the events received in the reputation system for each entity on its page to understand the reasons behind its reputation. Each explanation should include details such as the event's severity, the type of action taken, and any sub-actions involved.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The data streams with the reputation information must work for this information to appear on the platform, and the entity must have had interaction with the system with negative events
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) Retrieving the reputation information for the entity from the reputation system's database, including the events that have affected the entity's reputation, with their justifications, such as the degree of severity, type of action, sub-action, and other relevant information.
  (b) Developing a user-friendly interface that displays the reputation information in a clear and easy-to-understand format, including a list of the events that have affected the entity's reputation and its justifications.
  (c) Continuously monitoring and updating the entity's page to ensure the reputation information remains accurate and up-to-date.

VP5. *Entity Page with the Reputation History*

- **User Story**: As a platform user, I want to view the reputation score history for each entity on its page to see the changes in the entity's reputation over time. It should display the reputation score for each month, with the ability to zoom in and see the reputation score for each day of the selected month.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services

- **Requirement preconditions**: The data streams with the reputation information must work for this information to appear on the platform, and the entity must have interacted with the system.
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) The user navigates to the page of a specific entity on the website.
  (b) Retrieving the reputation information for the entity from the reputation system's database, including the historical reputation score data for each month.
  (c) On the page, the user sees a graph that displays the reputation score of the entity for each month in the past year.
  (d) Upon clicking on a month, the graph updates to display the reputation score of the entity on a day-by-day basis for that specific month.
  (e) The user can view the reputation score history for the entity on a granular level and easily identify patterns or trends in the data.
  (f) The user is satisfied with the level of detail provided and can make informed decisions based on the reputation score history.

VP6. *Statistics of the Reputation System*

- **User Story**: I want to view the reputation system's statistics on a single page as a platform user. Among these statistics, I would like to view the total number of entities in the system, total events processed, and information regarding each type of entity, such as the average, minimum and maximum reputation and the number of each type.
- **Related Domains**: Domains A and B
- **Requirements Scope**: Persons, IoT devices, Applications/Services
- **Requirement preconditions**: The system must be populated with several entities and events to present their statistics.
- **Requirement post-conditions**: N/A
- **Success Scenario**:
  (a) The user navigates to the page with the reputation system's statistics.
  (b) The platform consults the database and provides all the needed statistics for the web page.
  (c) The user can view all the statistics of the reputation system and is satisfied with the details provided by the web page.

4.3 Non-Functional Requirements

This section examines the quality attributes encompassed by the reputation system and the data visualisation platform, providing a methodology for evaluating these requirements.

Reliability The reputation system can be more reliable and withstand failures, providing a more consistent and trustworthy service to its users to ensure that all entities are processed or stored if there has been a problem with the reputation service. Simulations of system failures can evaluate the system's reliability to ensure that all events are processed correctly.

Scalability The system needs to be scalable to receive data from the different components of the project partners, each with numerous entities and events. The system's scalability can be evaluated by determining the number of messages processed per second by the message queue.

Security The reputation system must handle high traffic and data volume, maintain confidentiality, integrity and availability, and only allow authorised access to the database to ensure accurate and trustworthy reputation data is consistently available for query and transmission. The system's security can be assessed by evaluating the effectiveness of the implemented encryption mechanisms.

Usability The reputation system data visualisation platform should have a simple and intuitive interface that is easy to understand for all users. The usability of the data visualisation platform can be evaluated by administering a survey to potential users to assess their perception of the platform's intuitiveness and simplicity.

4.4 Requirements Listing

In this section, we use the MoSCoW method[1] to prioritise our project's functional requirements, explained in Section 4.2. This method allows us to classify the requirements into four categories:

- ***Must Have***: The requirement is non-negotiable. It must be present in the final product to not compromise the success of the product.
- ***Should Have***: The requirement is non-vital but dramatically increases the product value.
- ***Could Have***: These would be a "nice-to-have" requirement, but they are not necessary to the core functionalities of the product. Leaving it out only has a negligible impact on the product's success.
- ***Won't Have***: Indicates requirements beyond the project's scope. If included, this requirement contributes to the complexity by increasing the project scope.

[1] *MoSCoW Method*: `https://www.productplan.com/glossary/moscow-prioritization/`

Using the MoSCoW method, we can communicate the priorities of the requirements and ensure that the project stays focused on the most critical tasks. It also helps us make informed decisions about what requirements can be deferred or omitted if necessary. This allows us to effectively manage the project scope and ensure that we deliver a high-quality product that meets the needs of all stakeholders.

Table 4.1 and 4.2 represent the functional requirements listing using the MoSCoW method for the reputation system (RS) and the data visualisation platform (VP), respectively.

ID	**Name**	**Description**	**MoSCoW Scale**
RS1	Receive events from the components of the project partners	The reputation system needs to receive the events of the entities from the components of the project partners in the reputation system.	Must have
RS2	Select essential information to be stored	The reputation system selects the essential information from the events received to be stored in the database.	Should have
RS3	Calculate the entity's reputation score	Upon receiving a new event from an entity, the reputation system must update that entity's reputation score.	Must have
RS4	Trustable storage mechanism	The reputation system requires trustable mechanisms to store the reputation of entities and their events.	Should have

Table 4.1: Reputation System Requirements Listing

ID	**Name**	**Description**	**MoSCoW Scale**
VP1	Homepage Lists the Different Entities	The Homepage must list all the entities from which it gets the reputation information.	Must have
VP2	Filtering of Entities	Filter the different entities on the homepage and search for a particular entity type.	Could have
VP3	Entity Page with the Entity's History	Entity pages display various reputation dimensions.	Must have
VP4	Entity Page with the Events Additional Information	Entity pages presenting a list with the justification for the events in the reputation system.	Could have
VP5	Entity Page with the Reputation History	Entity pages should display a graph with the monthly reputation score history	Should have

VP6	Statistics of the Reputation System	Statistics page should display all the relevant information of the entities and events in the reputation system.	Should have

Table 4.2: Data Visualisation Platform Requirements Listing

4.5 Use Cases

In the use cases section, we present the different use cases for our project based on the domains we identified in the previous section. A use case describes a set of actions that a system performs to achieve a specific goal. It outlines a user's steps to accomplish a particular task and identifies the system's requirements to complete it successfully.

By presenting our project's use cases, we can clearly understand how users utilise the system and what it is expected to do. This helps stakeholders better understand the project's scope and the requirements that need to be met. It also helps to identify any potential issues or challenges that may arise during the development process.

The use cases in this section were reached based on previously identified real-world scenarios in Section 4.1. This allows us to provide a comprehensive overview of how users use the system and the different types of users it serves. Furthermore, by presenting the use cases in this way, we can ensure that all stakeholders clearly understand the project and can make informed decisions about its development.

In each use case, we include the following elements: a name for the use case, ID, a list of the actors involved, a description of the actions that take place in the use case, a list of preconditions that must be met for the use case to be initiated, a list of post-conditions that describe the state of the system after the use case is completed, a list of the entities of the system that are present in the use case, and an example of the reputation system's behaviour in specific events. The given use cases are based on an ARCADIAN-IoT deliverable [36], and the main contribution of this work is demonstrated through examples that illustrate the behaviour of reputation in specific events.

4.5.1 Use Cases based on Domain A

Use Case #1: Person registration at DGA service

- **ID**: A1
- **Actors**: Citizen/Person to guard
- **Story**

To register for the service, the user installs a compliant mobile app on their smartphone and provides the necessary data (including biometric information) for registration. The service offers the user trustworthiness and sets up security, privacy monitoring, and recovery mechanisms in case of an incident. The user's device has hardened encryption, integrity attestation mechanisms, and a decentralised identifier or verifiable credential. The framework also ensures that the user controls where their data is used.

- **Preconditions**

  1. DGA services are compliant with ARCADIAN-IoT
  2. The user has a smartphone with eSIM, and the app is installed
  3. Monitor user, personal device, and third-party service interactions to trigger any security action needed and update the trusted knowledge. This monitorisation may include dynamic reputation and authorisation changes of the parties involved.

- **Post-conditions**

  1. In the system, the user is registered. It has at least three robust identification mechanisms configured, one of them decentralised, and the Root of Trust (RoT) on their device has information for providing the personal data encrypted. The user can securely log in and start using the third-party services, the DGA services, with their sensitive data privacy ensured.
  2. The third party has the necessary data for providing its service.

- **Entities**: Person, IoT device and Services

- **Example of reputation's behaviour in events**

  - Person Registration
    - When a specific user registers himself in the service, the system gives the user a positive event.
    - When a specific user installs the app and registers himself in the DGA Service, providing all the fields in the form, a positive event shall be given to the user, the service and the device the app was installed.
    - If the user provides miscellaneous information in the form fields, a negative event shall be given to the user.
  - Person registration - biometrics
    - Upon obtaining consent from the user to utilise FaceID recognition, a positive event will be given to the user, the service, and the device.
    - When a user selects multiple images for successful FaceID recognition, a positive event will be given to the user and the device.
    - When a user provides a valid password for authentication, the person and the device receive positive events.

Use Case #2: Person authentication at DGA service

- **ID**: A2
- **Actors**: Citizen/Person to guard
- **Story**

  After registering, users must authenticate themselves with multiple robust identity mechanisms to access the DGA services, such as Two-Factor Authentication and Time-Based One-Time Password (TOTP). The app will use the identification mechanisms of new Self-Sovereign Identity (SSI) and cellular network credentials. Trust models informed by monitoring and Cyber Threat Intelligence (CTI) components will be in place to allow or deny the user's authentication.

- **Preconditions**

  1. Use case A1.
  2. Behaviour monitoring and CTI components monitor the interactions of the user, personal device and third-party service to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes of the parties involved.

- **Post-conditions**

  1. The user can log in to the DGA and request a service.

- **Entities**: Person, IoT device and Services
- **Example of reputation's behaviour in events**
  - Person Authentication
    * Upon successful authentication of a person in the service, a positive event shall be given to the user.
    * A negative event is generated if a person cannot authenticate in the DGA service after a specified number of attempts.
    * A negative event is generated if persons can authenticate but the DGA service is unavailable at their current location.

Use Case #3: Person retrieving and editing personal data

- **ID**: A3
- **Actors**: Citizen/Person to guard
- **Story**

  The end-user can retrieve and edit their data using the DGA mobile app. DGA services validate the user, device and app identity and trustworthiness. If the entities are authorised and trustworthy, the data is retrieved,

encrypted and decrypted at the device, with private cryptographic material kept secure at the hardware level for editing. After the intended edition, it is encrypted with RoT information and sent to the DGA services. It maintains encryption until the user authorises access (self-aware data privacy).

- **Preconditions**

  1. Use cases A1 and A2.
  2. Behaviour monitoring and CTI components monitor the interactions of the user, personal device and third-party service to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes of the parties involved.

- **Post-conditions**

  1. Updated personal data encrypted and stored in DGA services

- **Entities**: Person, IoT device and Services

- **Example of reputation's behaviour in events**

  - Person retrieves personal data
    - Upon requesting reputation information of the user, device and the reputation, allowing the editing and secure updating of the DGA app information, a positive event is generated and associated with the person and device.
  - Person edits personal data
    - If the phone has an outdated version of the DGA which does not support robust encryption mechanisms that protect against various attacks, such as Man-in-the-Middle (MITM) attacks, and, therefore, does not provide secure sending, a negative event is associated with the device, including the mobile app.
    - Upon the DGA app receiving the information and successfully editing it without any errors during the update process in the DGA service, a positive event shall be associated with the person, service, and device.

Use Case #4: User requesting a DGA service

- **ID**: A4

- **Actors**: Citizen/Person to guard

- **Story**

  When the end-user requests a drone using the DGA mobile app, the app sends the necessary personal data encrypted with RoT information to the DGA service. After verifying the user's identity and the trustworthiness of

the site, app, and device, DGA selects a trustworthy drone from the available options based on its reputation information. The selected drone's identity data is retrieved and attested to assure its trustworthiness before granting access to the user's data. The DGA service then shares the encrypted data with a trustworthy drone, which can decrypt it using RoT information to meet the person and attest to its identity. The person is informed of the drone's location and identification and may be given the drone's tag for visual identification upon its arrival at the service location. The person is also made aware of the self-aware privacy measures in place.

- **Preconditions**

  1. Use cases A1 and A2.
  2. Behaviour monitoring and CTI components monitor the interactions of the user, personal device and third-party service to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes of the parties involved.

- **Post-conditions**

  1. Use case A5.

- **Entities**: Person, IoT device and Services

- **Example of reputation's behaviour in events**

  - Person requests DGA service
    * If a user requests a drone and provides accurate information such as precise location, a positive event is associated with the person and device.
    * Upon the DGA service verifying a user's identity and reputation, a positive event is associated with the service.
    * If the user deliberately places themselves in a zone with poor coverage, leading to issues with accurate location determination for the DGA app, and despite the app warning the user they do not change their location, a negative event is associated with the user.
    * If the user's mobile device, where the service is requested, experiences location-related issues such as GPS malfunction, a negative event is assigned to the device.
    * Upon the DGA service dispatching drones according to their reputation, a positive event is associated with the drones.
    * If a drone flies according to its planned route, a positive event is associated with the drone.

Use Case #5: DGA service

- **ID**: A5

- **Actors**: Citizen/Person to guard
- **Story**

  After receiving the necessary data, the selected drone needs to meet and identify the person who requested it and begin the vigilance service. The person is informed about the drone's reputation and how to perform the required biometric identification. When the person is near the drone, a mutual authentication process occurs between the person, their device, and the drone. All private data exchanged is encrypted and only accessible by the targeted devices. The identification process involves three simultaneous identifications of the person. The service begins if the process is successful and the drone follows the person. Any user data sent by the drone to DGA services, including from emergency or rescue situations, is encrypted and only accessible by entities authorised by the user. When the service ends, all user data is deleted from the drone, and any data needed by DGA services about the user or the service is encrypted. The user is informed about which data is kept in the service and can choose to have it deleted.

- **Preconditions**

  1. Use case A4.
  2. Behaviour monitoring and CTI components monitor the interactions of the user, personal device and third-party service to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes of the parties involved.
  3. Processes of anonymisation of people near the user are in place.

- **Post-conditions**

  1. If the service ends successfully, the drone returns to the base with no other action needed.
  2. If the service doesn't end successfully for some, the behaviour monitoring and the CTI try to infer the reason and act accordingly. DGA services perform the corrective measure considered needed (e.g., an operator calls the user, or the services send another drone to the last known location to continue the service or assess the situation).

- **Entities**: Person, IoT device and Services
- **Example of reputation's behaviour in events**

  - Drone in DGA service
    - When a drone arrives at the location, and the user acknowledges the drone's arrival, a positive event shall be given to all entities.
    - When a drone identifies the user through the DGA app, a positive event shall be given to all entities.

    - When a person is not present in the location where they requested the DGA service and the drone informs the DGA service that no person has been found, a negative event shall be given to the person.
    - When a person receives a notification of drone arrival in the DGA app and does not move to be identified within a certain time, a negative event shall be given to the person and device.
    - When a drone is not in the correct location, but the user is in the correct location, a negative event shall be given to the service and the drone.
  - Drone in DGA service - recognise the person
    - When a drone fails to identify a user due to using old photos or other photos that do not allow for good identification, such as those with masks or clothing covering the face, a negative event shall be given to the person.
    - When a drone sends a message to the DGA service requesting the user to be in the Line of Sight (LOS), for example, to move away from trees, and the user does not move after several warnings, a negative event shall be given to the person.
    - When the drone and the user's device with the DGA app perform mutual authentication, a positive event shall be given to all entities.
    - When the drone and the user's device with the DGA app fail to perform mutual authentication due to timeout and several attempts, such as if the user disconnects the interface providing the connection to the DGA service, a negative event shall be given to the device.
    - When the user arrives at the final location and reports it according to the DGA service policies, and the DGA service registers at the end of the mission, a positive event shall be given to all entities.
    - When the user arrives at the final location but does not report it to the DGA service and the drone returns to the station after a certain time, a negative event shall be given to the person.
    - When the user rates the service provided by the drone, a positive event shall be given to the service, device, and drone.
    - When the drone receives the notification of the end of the mission and performs the deletion of data that is not required, a positive event shall be given to the drone.

Use Case #6: Drone security or privacy incident

- **ID**: A6
- **Actors**: Attacker(s)
- **Story**

A system has been implemented to monitor and detect potential threats to protect against security and privacy incidents involving IoT devices like drones. This system uses a CTI tool and a federated Artificial Intelligence (AI) approach to train an AI model on distributed information while maintaining data privacy and informing the device's self-protection mechanisms. Data is encrypted to protect end-user privacy and undergoes a security assessment before being sent to the device. A dynamic reputation system determines the device's trustworthiness based on factors such as previous behaviour. If a security incident is detected, the device's internet access is restricted to only those services necessary for recovery. If the device is operational and cooperative, it will take action to recover from the incident according to pre-defined procedures. The CTI tool also shares threat information with other networks to increase awareness.

- **Preconditions**
  1. The drone is compliant with ARCADIAN-IoT
  2. Related framework components (e.g., reputation system, authorisation, self-protection, self-recovery) are operational
- **Post-conditions**
  1. If the device is operational (e.g., didn't get damaged), not stolen, and cooperative, there has been a mitigation of the incident, and its security and privacy are restored.
  2. Anonymised data/trained models about the incident are shared with Computer Security Incident Response Team (CSIRT) and Computer Emergency Response Team (CERT)
- **Entities**: IoT device and Services
- **Example of reputation's behaviour in events**
  - Drone is stolen
    - When the user reports that the drone did not arrive at the planned time and the DGA service cannot retrieve the drone's location, a negative event shall be given to the service and drone.
    - When the drone stops providing status messages, including location, to the DGA service after a certain period of time, a negative event shall be given to the drone.
    - When the drone reports strange behaviour, such as information from the accelerometer or other data sensors that may indicate a malfunction or the impact of atmospheric conditions, a positive event shall be given to the service and drone.
    - When the drone reports strange behaviour, such as information from other data sensors that may indicate malfunction or deviation from the planned route, a positive event shall be given to the service, and a negative event shall be given to the drone.
  - Drone is stolen - the recovery process

* If the person who attempts to capture the drone is the user, a negative event shall be given to the person, or the reputation may be reset.
* When the drone reports malfunction triggers an alert to initiate recovery, and the recovery is completed, leading to the restoration of the drone, a positive event shall be given to the drone.
* When the drone reports malfunction triggers an alert to initiate recovery, and recovery is attempted but does not lead to a successful restore due to the drone being captured illegally, a negative event shall be given to the drone.

Use Case #7: Personal device security or privacy incident

- **ID**: A7
- **Actors**: Attacker(s)
- **Story**

  To protect against security and privacy incidents involving personal devices that access DGA services, a system has been created to collect and interpret information about potential threats using a behaviour monitoring component and a CTI tool. To protect end-user privacy, sensitive data on the device is encrypted, and a dynamic reputation system is in place to determine the device's and app's trustworthiness based on factors such as its behaviour. If a security incident is detected, the device's internet access is restricted to only those services necessary for recovery. If the device is operational and cooperative, it will take action to recover from the incident according to pre-defined procedures. If the self-recovery processes are successful, the device's hardware and software will be restored to compliance, including credentials recovery. The CTI tool also shares threat information with other networks to increase awareness. Human intervention is minimised during the recovery process.

- **Preconditions**

  1. Persona device is compliant and integrated with the framework component.
  2. Related framework components (e.g., CTI, reputation system, authorisation, self-protection, self-recovery) are operational.

- **Post-conditions**

  1. If the device is operating (e.g., didn't get damaged) and not stolen, the incident is mitigated, and its security and privacy are restored.
  2. Threat information in the form of trained models is shared with CSIRT and CERT networks to propagate threat awareness.

- **Entities**: Person, IoT device and Services

- **Example of reputation's behaviour in events**
  - Personal device security
    * When the user informs the DGA service via the DGA service website that their device has been stolen upon arriving home, the device's reputation shall be reset.
    * A negative event shall be given to the person and device for failing to inform the DGA service about the loss of the device with the DGA app, resulting in another user attempting to use the service with multiple failed login attempts from outside the location of previous missions.
    * The user acknowledges the recovery of their device with the DGA app, but the device has not attested properly, leading to a negative event for the device.

4.5.2 Use Cases based on Domain B

Use Case #1: MIoT kit delivery - Patient registration and authentication

- **ID**: B1
- **Actors**: Patient and Medical Professional
- **Story**

  A patient at a hospital is given an MIoT kit, including a smartphone with a specific app installed and medical sensors, to be monitored at home. The authorised medical professional asks the patient to register using the MIoT app on the smartphone, free of any personal data from previous patients. The app presents information about the service's data security and privacy procedures, and the security reputation of the MIoT entities is shown to the patient. If the patient agrees, a personal "RoT" (eSIM) is generated and provisioned to the smartphone, and cryptographic material is generated and sent securely to the device RoT. The patient then fills out a form with personal data, part of which is used to create their SSI, and this data is encrypted and submitted to the MIoT services. The SSI is backed up in a permissioned blockchain and stored in the device RoT or a secure ID wallet. The patient is informed that their new network credentials, associated with the eSIM/RoT, will be used as a second secure identification/authentication mechanism. The medical staff then confirms the delivery of the equipment to the patient and provides any necessary explanations about its use.

- **Preconditions**

  1. MIoT app and services are compliant.
  2. Smartphone provided in the MIoT kit has embedded Universal Integrated Circuit Card (eUICC).
  3. Medical staff is registered in MIoT monitoring services with authentication and identification compliant

4. Health sensors are securely paired with the smartphone (components act in the smartphone for security)

- **Post-conditions**

1. Patient registration and authentication
2. Behaviour monitoring and CTI components start overseeing the device behaviour to trigger any necessary security actions and update the related trust knowledge (which may include reputation and authorisation changes).
3. The patient can be made aware of where their data will start being sent, authorisation since this moment specific doctors

- **Entities**: Person/IoT/Apps Services
- **Example of reputation's behaviour in events**
  - MIoT kit delivery - patient registration and authentication
    * The user completes the registration process for the DGA service, indicating a positive event for both the user/patient and the service.
    * The patient completes the registration process and is authenticated, resulting in a positive event for the patient, service and device.
    * The patient's authentication attempt is unsuccessful, leading to a negative event for the patient and the device used.

Use Case #2: MIoT capturing and sending vital signs and perceived health status

- **ID**: B2
- **Actors**: Patient
- **Story**

  The patient at home has sensors placed on their body and synced with a smartphone, and the MIoT app starts collecting data from the sensors. If necessary or according to the health monitoring plan, the patient also reports their perceived health status through the app. The app on the smartphone encrypts the data from the sensors and the patient's perceived health status with RoT (eSIM) information and sends it to MIoT middleware services. If there is no connectivity, the device stores the encrypted data locally and sends it to MIoT services when communication is restored. With the patient's permission, MIoT middleware forwards the data to compliant hospital monitoring tools, which can decrypt the data with the patient's consent. If the smartphone is turned off, the patient must re-authenticate in MIoT services, and the app starts collecting sensor data again. A re-authentication process may also be applied for security purposes.

- **Preconditions**

  1. Use case B1.
  2. Sensors are well placed (the app can help by informing when data needs to be received).
  3. Behaviour monitoring and CTI components oversee the interactions of the IoT device (the IoT gateway) and services involved to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes for the parties involved.

- **Post-conditions**

  1. The MIoT app transmits encrypted data to the MIoT middleware, which can be forwarded and encrypted to authorised IoT monitoring tools.

- **Entities**: Person/IoT/Apps Services

- **Example of reputation's behaviour in events**

  - MIoT capturing and sending vital signs and perceived health status
    * The encrypted data is successfully transmitted to the hospital, representing a positive event for the service and device.
    * The individual completes the re-authentication process, which is a positive event for both the patient and the device.
    * The individual fails to complete the re-authentication process, resulting in a negative event for both the patient and the device.

Use Case #3: Personal data processing towards health alarm triggering

- **ID**: B3

- **Actors**: Patient

- **Story**

  This use case involves processing health data in the cloud to detect and trigger alarms in a hospital monitoring tool. The data, which is encrypted when it is received from the patient's MIoT smartphone, needs to be decrypted to detect any alarm conditions. The patient must authorise the processing of their data for this purpose, and if they do, cryptographic material to interpret the data is provided to the processing unit. If the patient revokes authorisation, new cryptographic material is generated and distributed but not offered to the team. To maintain patient privacy, decryption techniques only decrypt the data payload and encrypt the patient's identity. The processing unit detects alarm conditions based on medical protocols or patterns learned from other anonymous patients, and these alarms are encrypted and merged with the patient's encrypted identification. With the patient's authorisation, MIoT middleware forwards the encrypted data, including

the warnings, to compliant hospital monitoring tools, which must have strong identity and authentication mechanisms, allow security behaviour monitoring, and authenticate in MIoT services to receive the cryptographic material needed to decrypt the data. Decryption by medical staff only occurs with the patient's authorisation.

- **Preconditions**
  1. Use case B2.
  2. Behaviour monitoring and CTI components oversee the interactions of the services involved to trigger any necessary security actions and update the trusted knowledge. This may include dynamic reputation and authorisation changes for the parties involved.
- **Post-conditions**
  1. When relevant, patient health alarms are triggered and sent to the hospital.
- **Entities**: Services
- **Example of reputation's behaviour in events**
  - Personal data processing towards health alarm triggering
    * The patient consents to processing their data, which represents a positive event for the service.
    * Anomalous behaviour, such as a false health alarm, is reported for the mobile device, which is a negative event for the device.
    * Threats, like unauthorised access and Denial of Service attacks, are reported regarding the MIoT service, representing a negative event for the service.

Use Case #4: Monitor a patient and update the patient monitoring protocol

- **ID**: B4
- **Actors**: Medical Professional
- **Story**

  Patient monitoring is a key use case for MIoT, and the system is designed to monitor patients efficiently and securely while at home. Medical professionals log in to the MIoT hospital platform with strong multifactor authentication. MIoT services validate the user and the app's identity and assess its reliability to grant access to the services. The medical professional can select a patient to whom they have access or request access to a new patient. They may also have a dashboard for monitoring multiple patients with authorised access to their data. Patient data is encrypted until requested by an authorised medical professional, who can decrypt it with cryptographic material provided by self-aware data privacy and hardened encryption. If

the medical professional wants to change a patient's monitoring protocol, they request that MIoT services send commands to the patient's MIoT app. The MIoT hospital platform decrypts data sent by the patient app. It allows the medical professional to edit it before sending it back, encrypted, to the MIoT app on the patient's smartphone. The smartphone must be on, connected, and the patient securely authenticated with multiple factors in MIoT middleware to receive the new commands, decrypted with RoT (eSIM) information and executed if successful or discarded if not.

- **Preconditions**

  1. Use case B1.
  2. Medical professionals, the user of the MIoT hospital platform registered with an identification compliant with personal identification.
  3. Medical professional authorised by a patient (or more) to decrypt his health data for well-being monitoring purposes.
  4. The MIoT hospital platform is provisioned with material for encrypting the commands sent.
  5. Devices RoT (eSIM) provisioned with material for decrypting the commands sent.
  6. Behaviour monitoring and CTI components monitor the interactions of the entities involved to trigger any security actions deemed necessary and update the trusted knowledge. This may include dynamic reputation and authorisation changes for the service.

- **Post-conditions**

  1. Authorised medical staff can monitor a patient's health and update the medical protocol.
  2. In the case of a medical protocol change, the MIoT kit has new information to update the medical monitoring protocol.

- **Entities**: Person/IoT/Apps Services

- **Example of reputation's behaviour in events**

  - Monitoring a patient and updates to the patient monitoring protocol
    * A medical professional successfully authenticates on the MIoT hospital platform, representing a positive event for the medical staff, service and the MIoT hospital platform.
    * A medical professional successfully accesses patient data through the MIoT hospital platform, representing a positive event for the medical staff, service, and the MIoT hospital platform.
    * A medical professional is unable to access patient data due to an error, specifically, the MIoT hospital platform is unable to decrypt the data. This represents a negative event for the MIoT hospital platform.
    * A medical professional is trying to access the data of a patient for which it does not have permission, representing a negative event for the medical staff.

Use Case #5: Patient MIoT devices security or privacy incident

- **ID**: B5
- **Actors**: Attacker(s)
- **Story**

  In this scenario, a security or privacy incident involves a patient's MIoT devices (smartphone/app and sensors). The system is designed to detect, protect, and recover from such incidents through the use of components like behaviour monitoring, flow monitoring, and a CTI tool, as well as a self-protection component that infers threats based on data received and applies rules to protect MIoT devices and the IoT network. To protect the patient's private information, the data collected by the smartphone is encrypted and only sent to compliant, authorised, and trustworthy medical third parties. The patient's identity is composed of multiple factors, including network credentials stored in the secure hardware element (eSIM/eUICC) and an SSI stored securely in the secure element of the RoT or ID wallet. In the event of a security incident, the smartphone's trustworthiness reputation is updated, and access authorisation enforcement is also updated to allow the MIoT app to only access network services for recovery, not services that may provide or request private data or cryptographic material. If the device/app is operational, it takes self-recovery and self-healing actions to recover from the incident, potentially requiring access to ARCADIAN-IoT services. If the device is not operational, a recovery process is initiated, potentially requiring the patient to replace the device and restore their identity and data.

- **Preconditions**

  1. MIoT devices are compliant (being therefore integrated with the framework components).
  2. Related framework components (e.g., reputation system, authorisation, self-protection, self-recovery) are optional.

- **Post-conditions**

  1. If the device is operational (e.g., not damaged) and not stolen, the incident is mitigated, and privacy is restored with reduced human intervention.
  2. Threat information in the form of trained models is shared with CSIRT and CERT.

- **Entities**: Person and IoT device
- **Example of reputation's behaviour in events**
  - Patient MIoT devices security or privacy incident
    - A security incident, such as unauthorised access, is detected, representing a negative event for the patient and the device.

    * The self-recovery and self-healing processes are successfully implemented, representing positive events for both the patient and the device, culminating in the negative event previously given.

Use Case #6: MIoT Cloud services security or privacy incident

- **ID**: B6
- **Actors**: Attacker(s)
- **Story**

  This use case describes a process for handling security or privacy incidents involving MIoT Cloud services, such as a data processing unit for detecting and triggering health alarm conditions. The service is prepared to detect and respond to incidents and protect private data and identities. The process includes using components for incident detection, protection, and recovery, a self-protection component that infers threats and applies protection rules, and a dynamic reputation system that assesses the trustworthiness of services. The service identity is protected with decentralised identifiers and attestation to prevent impersonation. In the event of a security incident, the service's reputation is updated, and access to certain services may be restricted. The service can then take actions to recover from the incident, including self-recovery and self-healing processes, and can recover decentralised identifiers from the ARCADIAN-IoT blockchain component. The self-healing component includes a decision manager and resource inventory to guide recovery. The CTI tool shares relevant threat information with CSIRT and CERT networks throughout the process.

- **Preconditions**

  1. MIoT Cloud services are compliant (being therefore integrated with the framework components).
  2. Related framework components (e.g., behaviour monitoring, flow monitoring, reputation system, authorisation, self-protection, and self-recovery) are operational.

- **Post-conditions**

  1. The incident is mitigated, and its security and privacy are restored, with minimum or no human intervention.
  2. Threat information in the form of trained models is shared with CSIRT and CERT.

- **Entities**: Services
- **Example of reputation's behaviour in events**

  – MIoT services security or privacy incident

  - A security incident is detected, representing a negative service event.
  - The self-recovery and self-healing processes are successfully implemented, representing a positive event for the service.

Use Case #7: Medical third-party security or privacy incident

- **ID**: B7
- **Actors**: Attacker(s)
- **Story**

  This use case involves a medical service that receives data from MIoT devices to monitor patients' health. The service is designed to protect patient's personal information by keeping the data encrypted until it is requested by a trustworthy, authorised medical professional who has been properly authenticated. The service also includes various components for detecting, protecting against, and recovering from security and privacy incidents, such as unauthorised access to data or manipulation of service functionality. These components include behaviour and flow monitoring, a CTI tool, and a dynamic reputation system that determines the trustworthiness of medical services based on various factors. To protect the service's identity, each service is assigned a decentralised identifier and undergoes attestation to prevent impersonation. If a security incident is detected, the service's reputation is updated, access authorisation is enforced, and self-recovery and self-healing procedures are initiated to restore the service to a compliant state. The self-healing component includes a decision manager and resource inventory to determine how to repair the service without human intervention. The CTI tool also shares threat information with other networks to increase awareness.

- **Preconditions**

  1. The third-party medical platform is compliant (being therefore integrated with the framework components).
  2. Related framework components (e.g., behaviour monitoring, flow monitoring, reputation system, authorisation, self-protection, and self-recovery) are optional.

- **Post-conditions**

  1. The incident is mitigated, and its security and privacy are restored, with minimum or no human intervention.
  2. Threat information in the form of trained models (not the actual data) is shared with CSIRT and CERT.

- **Entities**: Services and person
- **Example of reputation's behaviour in events**

- Medical third-party security or privacy incident
  - A security incident is detected, representing a negative event for the service and the MIoT hospital platform.
  - The self-recovery and self-healing processes are successfully implemented, representing a positive event for both the service and the MIoT hospital platform.

4.6 Data Visualisation Platform Mockups

This section presents the various designs and layouts developed for the platform. We intend for these designs and layouts to provide a comprehensive and user-friendly interface for viewing and analysing data related to the reputation system. The reputation system is the crucial component of our proposed approach, designing the data visualisation platform in an intuitive, straightforward, and easy way.

The mockups presented in this section aim to show how the platform will look and function. The mockups were done using the Figma[2] platform.

The data visualisation platform consists of a homepage, a page for each entity in the system and a statistics page for the reputation system. In the following Subsections, 4.6.1, 4.6.2 and 4.6.3, we detail each of these pages, examining their features and functions. The homepage provides an overview of the reputation system as a whole, the entity page focuses on the reputation of individual entities within the system, while the statistics page offers all the essential statistics for the reputation system.

4.6.1 Homepage

As mentioned, the homepage aims to provide an overview of the reputation system and all the entities included. After logging in to the website, we can see a welcome screen for the data visualisation platform, here named "Reputation System Analyser", as shown in Figure 4.3.

Next, if we scroll down, there is a list of all the entities incorporated in the system. Each has a name, type (e.g. person, device and service), and reputation score in the system, as seen in Figure 4.4. It is also possible on this page to search for a particular entity or filter the entities based on some factors: the oldest or most recent date the system received an event from the entity in the system and the highest or lowest reputation score inside the system.

[2] *Figma*: `https://www.figma.com/`

Figure 4.3: Data Visualisation Platform Homepage

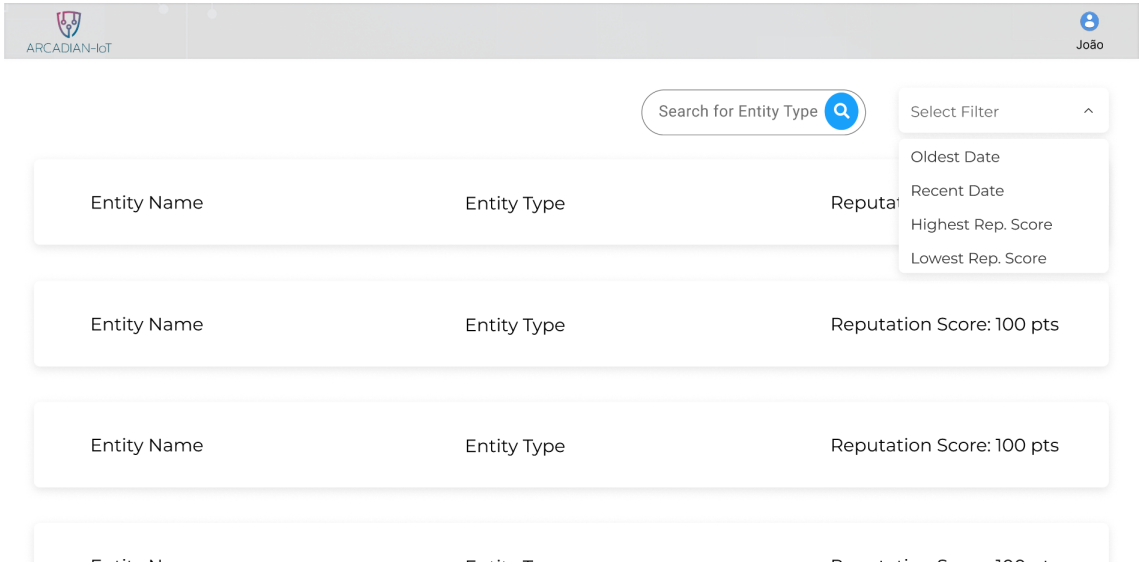


Figure 4.4: List of entities in the system, search and filtering

Selecting an entity from the list on the homepage takes the user to the page of that specific entity.

4.6.2 Entity Page

Users are presented with the entity's name, type and reputation score when entering the page selected on the homepage. Below it, we can see the parallel coordinates graphic of all the entity's interactions throughout the reputation system, including the rating, date and action type (e.g., login, alert, transaction, etc.) of each event, as seen in Figure 4.5.

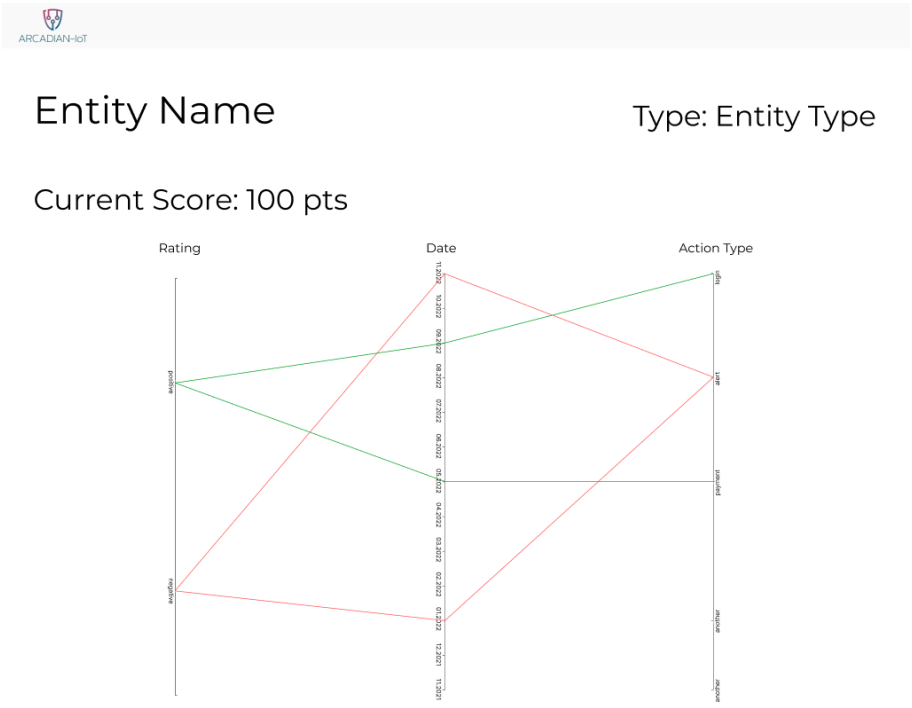


Figure 4.5: Entity Page - Parallel Coordinates

If the entity has any events that require further explanation of the ratings received, there is a section explaining these ratings given to the event, seen in Figure 4.6. The platform incorporates this explanation in a list of events with the following information: event's type, additional information about the event (e.g., if the event is alert, additional information can be a Distributed Denial of Service (DDoS)) and the severity of the adverse event inside the system. It should be noted that within the list provided, events in green signify positive events, while red signifies negative events, as received by the system for the specific entity in question.

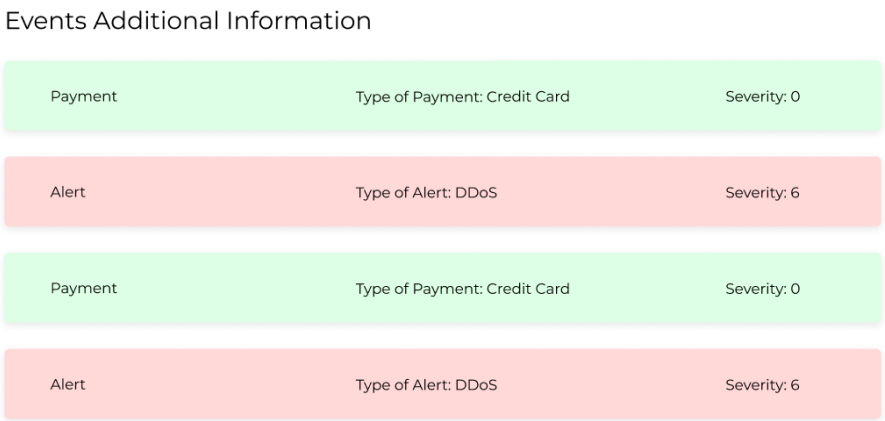


Figure 4.6: Entity Page - Events Additional Information

Finally, at the end of the page, there is a graphic with the reputation score history of the entity since its first interaction with the reputation system, as illustrated in

Figure 4.7.

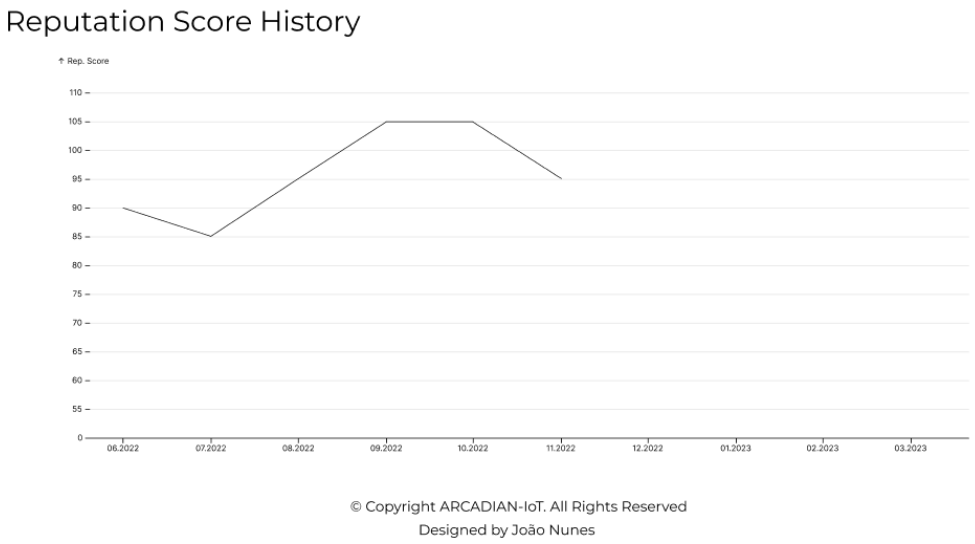


Figure 4.7: Entity Page - Reputation Score History

4.6.3 Statistics Page

As for the statistics page, the platform users are presented with the total number of entities in the system and the total number of events processed by the reputation system.

Next, information is presented on each of the different types of entities. This information highlights the number of entities in the system of each type (i.e., service, person) and the mean, minimum and maximum reputation of that type of entity.

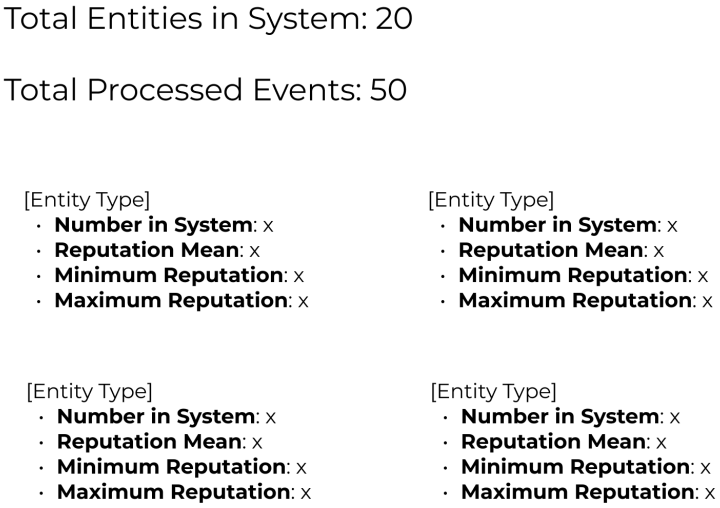


Figure 4.8: Statistics of the Reputation System Page

Chapter 5

Project Architecture

In this chapter, we delve into the design and structure of our reputation system through architectural diagrams. These diagrams provide a visual representation of the various components of our system and how they interact and function together to enable the reputation system. By presenting this detailed overview of the system's architecture, we aim to clearly understand how the reputation system works and how we can use it to improve the quality and trustworthiness of online interactions. Furthermore, this chapter provides a comprehensive understanding of the reputation system's architecture in a preliminary phase, meaning that some changes will occur based on feedback and the development process of the final product.

We illustrate the architectural diagrams based on the C4 Model[1] developed by Simon Brown. The C4 Model is a hierarchical model that provides a standardised way to visualise and communicate the structure and design of software systems. It is a flexible model representing many architectures, from small to large, complex systems. Using the C4 Model to create our diagrams, we aim to clearly and consistently represent the reputation system's architecture that facilitates understanding and communication among stakeholders.

The C4 Model consists of four diagrams that we can use to represent the architecture of a software system. These levels are:

1. **C1 - Context Diagram** (Section 5.1): This high-level diagram shows the relationships between the system and its external actors or stakeholders. It provides a broad overview of the system and its environment, including its boundaries, interfaces, and dependencies.
2. **C2 - Container Diagram** (Section 5.2): This diagram represents the system's logical organisation into containers, defined as deployable units that can run on a host. Containers seek to describe components, services, or modules that make up the system.
3. **C3 - Component Diagram** (Section 5.3): This diagram provides a detailed view of the components and their relationships within a container. It shows

[1] *C4 Model*: `https://c4model.com/`

how the components are organised, communicate, and interact with external systems or services.

4. **C4 - Code Diagram**: This is the most detailed level of the C4 Model, showing the classes, their attributes, and the relationships between them. Class diagrams can represent a system's internal structure and design at a granular level.

5.1 C1 Context Diagram

The Context Diagram, represented by C1, provides a high-level overview of the actors and software systems involved in the architecture.

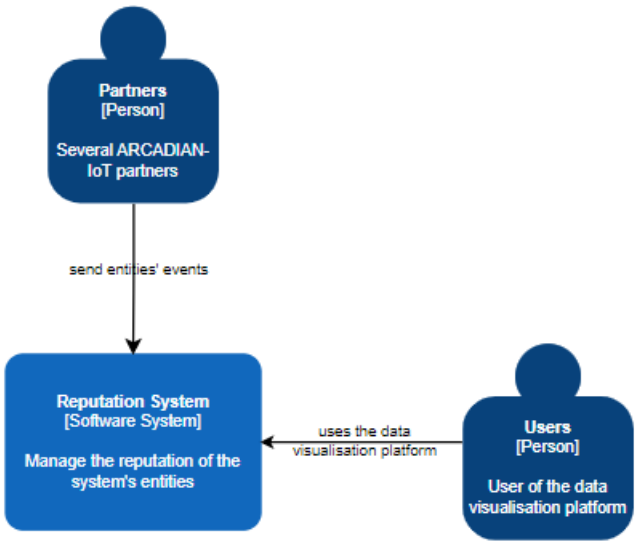


Figure 5.1: C1 architectural diagram - Context

As depicted in Figure 5.1, the Context Diagram for this project includes the following elements:

- **Partners [Person]**: The Reputation System relies on input from several project partners, who provide essential information regarding the events of the entities that these are responsible for.
- **Users [Person]**: Represent all the users that can visit the data visualisation platform to learn about critical information of every entity in the Reputation System.
- **Reputation System [Software System]**: Represents the entire reputation system and its different categories that manage the reputation of every entity it includes.

5.2 C2 Container Diagram

The Container Diagram, represented by C2, illustrates the relationships and dependencies between the various software elements in the architecture.

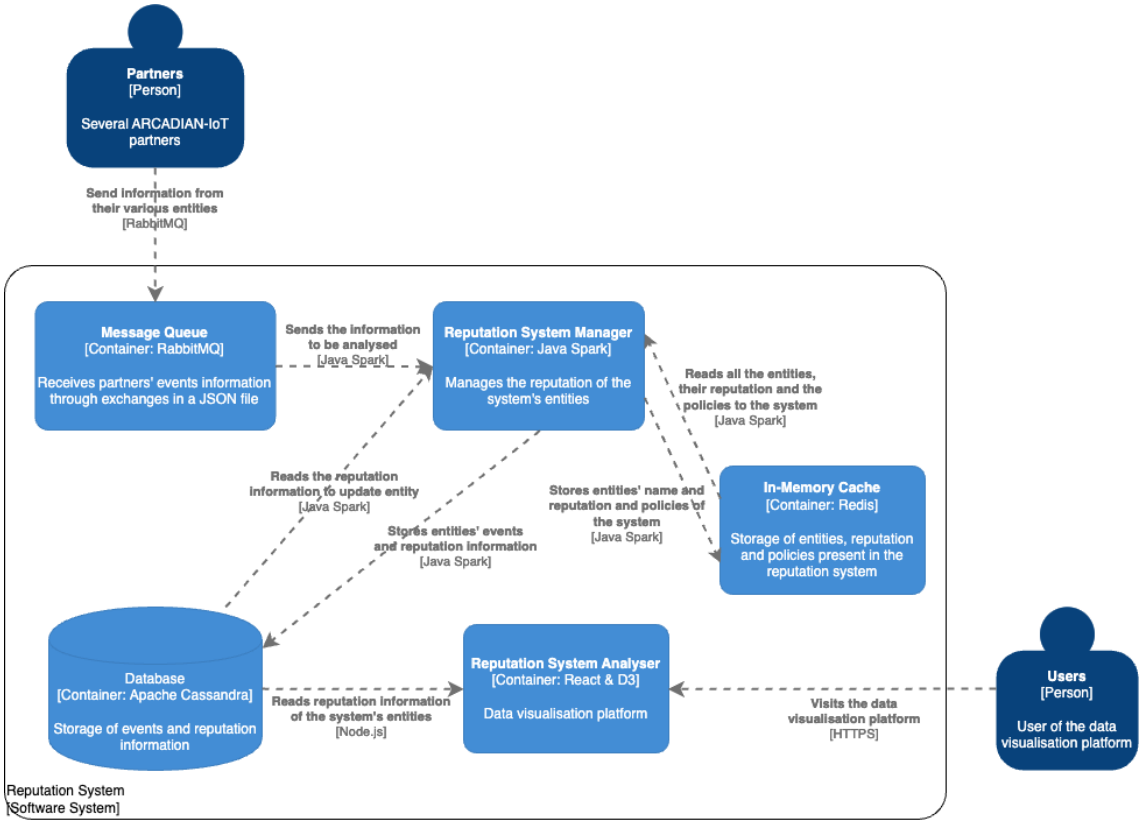


Figure 5.2: C2 architectural diagram - Container

As shown in Figure 5.2, the Container Diagram for this project includes the following features:

- **Message Queue [Container]**: Using RabbitMQ[2], the message queue receives the events sent by the project's partners through different exchanges in JavaScript Object Notation (JSON) format. The project partners authenticate themselves using their RabbitMQ exchange credentials (username and password). Each project partner will connect to a different message queue exchange.

- **Reputation System Manager [Container]**: With the Java Spark framework[3], the system will receive information from the message queue and analyse it, managing each entity's reputation, such as updating its reputation score, and storing the critical information in the database and the in-memory cache.

[2] *RabbitMQ*: `https://www.rabbitmq.com/`
[3] *Java Spark*: `https://sparkjava.com/`

- **Database [Container]**: Stores all the events and reputation information from every entity in the system. When the system receives an entity's event, it is necessary to store four essential pieces of information for the data visualisation platform:

  - *Event type (positive or negative)* - It is necessary for both the platform and the reputation system to update the reputation value based on whether the received event is positive or negative.
  - *Timestamp* - Saved so that on the platform, it is possible to view the date the reputation system received a particular event.
  - *Action type* - Stored so that the platform can display the action a particular entity did for the event.
  - *Origin Exchange* - It is important to know the exchange from which the event was sent because the system can receive events of a certain entity from different exchanges.

  Regarding the reputation information, the database needs the following topics to calculate the reputation score of each entity:

  - *Entity's ID* - It is necessary to have it in the database to have a comparison factor when more events are received through RabbitMQ from the same entity to be able to update the reputation value.
  - *Type of entity* - Stored mainly to facilitate the identification of the entity on the entity data visualisation platform.
  - *Mean* - Essential value, as the reputation value directly depends on this value.
  - *Reputation values* - It is also helpful to save the history of all the entity's reputation values in the database so that the visualisation platform has a graphic with the reputation history of each entity.
  - *Reputation Model* - Represents which model was used to calculate the reputation score, e.g., if the model was the Alpha-Beta or Alpha-Beta with a severity factor.

- **Reputation System Analyser [Container]**: Implemented in React with D3.js for designing the visualisation of the reputation data. This analyser is the data visualisation platform that reads all the needed reputation information from the database and displays it on different web pages. Authorised users of the reputation system visit this platform using Hypertext Transfer Protocol Secure (HTTPS). React was chosen for this project due to its intuitiveness and familiarity.

- **In-Memory Cache [Container]**: The entities' names and reputation information, specifically the alpha and beta values, along with the reputation policies existing within the reputation system, is stored utilising Redis[4], an in-memory cache. This utilisation facilitates reimporting all entities and

[4] *Redis*: `https://redis.io/`

policies in case of a reputation system failure. Redis was chosen for this functionality because it is a swift tool for optimised reading and writing data in memory.

5.3 C3 Component Diagram

The C3 Component Diagram illustrates the relationships and dependencies between the various software components in the architecture.

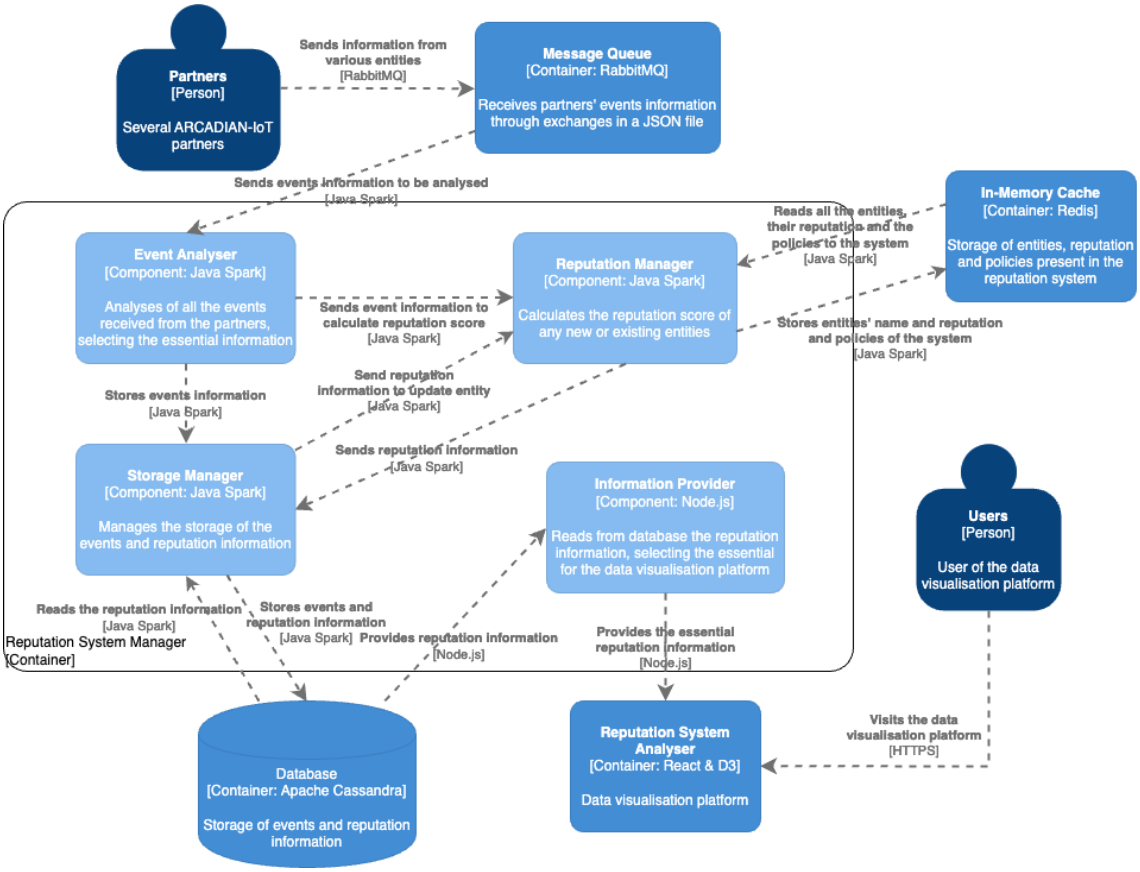


Figure 5.3: C3 architectural diagram - Component

As shown in Figure 5.3, the Component Diagram for this project includes the following details:

- **Event Analyser [Component]**: Implemented in Java Spark, the event analyser receives the events from the message queue and analyses them to send the essential reputation information to the Reputation Manager, sending the event data to the Storage Manager.
- **Reputation Manager [Component]**: Implemented using Java Spark, is responsible for receiving reputation information and calculating the reputation score for each entity using the reputation models chosen for the calculation (i.e., the Alpha-Beta Model). This component sends the estimated

reputation score to the Storage Manager for storage and future reference. It sends the entity name, reputation, and policies affecting the system to the in-memory cache. The Reputation Manager plays a crucial role in the overall reputation system, as it is responsible for accurately and efficiently calculating reputation scores based on the received reputation information.

- **Storage Manager [Component]**: Implemented using Java Spark, it interacts with the database to store and retrieve all relevant information from the events and reputation data. It provides a secure and efficient means of storing and accessing the data points used to calculate reputation scores and inform system decisions. By utilising Java Spark, the Storage Manager can manage the storage and retrieval of large volumes of data, ensuring the smooth operation of the reputation system. Upon consultation with the database, the Storage Manager shall also be utilised to confirm the existence of an entity within the reputation system to furnish the Reputation Manager with the current information of the specified entity to be updated upon receipt of a new event.

- **Information Provider [Component]**: Implemented using Node.js, serves as the backend of the data visualisation platform. It retrieves reputation information from the database and provides it to the Reputation System Analyser container for display. Using Node.js, the Information Provider can retrieve and transmit large volumes of data, ensuring the Reputation System Analyser can promptly display relevant information to users. Node.js was chosen for this functionality due to its ease of learning, scalability and the ability to handle several requests simultaneously.

Chapter 6

Implementation

This chapter delves into the technical details of our reputation system, including components such as the reputation calculation mode, data visualisation platform, and others. We discuss the database architecture and security measures to protect sensitive data.

In the second semester, this chapter emphasises the development of the reputation system and the data visualisation platform. By providing thorough explanations, it is expected that readers will gain a comprehensive understanding of the reputation system's mechanics.

6.1 Alpha-Beta Model

As previously said in Section 2.2.5, the model chosen by the project's team was the Alpha-Beta Model due to its ease of implementation and the fact that it is a good model for any entity the system receives. We propose implementing the Reputation Manager component, which is responsible for calculating the reputation score in Java, utilising various Java libraries to facilitate the calculation process. This approach allows us to take advantage of the functionality and efficiency offered by these libraries.

The Reputation Manager has a HashMap attribute, with the key being a string representing the name of the entity and the value being the BetaDistribution information for that entity. The system uses this information to calculate the reputation of each entity using the Alpha-Beta Model, which utilises a beta probability density function. Apache Commons' BetaDistribution library[1] assists with these calculations, for example, by finding the numerical value of the mean of this distribution. A static attribute, known as the ageing factor, is also proposed at 0.5 since we wanted past events to have some effect on the reputation calculation but not an excessive impact.

If a new entity introduces itself to the system, the Reputation Manager initialises the HashMap attribute for that entity with an initial alpha and beta value of 1.0.

[1] *BetaDistribution*: `https://bit.ly/3CeR8zH`

We give these initial values because when an entity is new to the system, we do not have information about its reputation. As a result, we must assign it an initial reputation value that reflects this need for more information. By setting the alpha and beta values to 1.0, we give the entity a reputation value equally likely to be either positive or negative, reflecting our uncertainty about its actual reputation.

6.1.1 Obtain Reputation Score

We base the reputation score for an entity on its numerical mean, which represents the average of all ratings received by the entity. To obtain this value, we call the *getNumericalMean()* method from the BetaDistribution class. This method calculates the numerical mean of the entity by taking into account the total number of ratings received and the sum of all ratings. Using this method, we can accurately determine the reputation score for each entity based on its numerical mean. It is important to remember that the ageing factor impacts the calculation of reputation through the prior values of alpha or beta, depending on the type of event.

6.1.2 Reputation Score Update

Updating the reputation score involves several input parameters, including the stream's name that contains the relevant information and an anomaly flag indicating whether a negative event (anomaly=true) or positive event (anomaly=false) has occurred. It is important to note that an ageing factor directly influences the updated reputation value, which considers the previous value when calculating the new value. The constant AGING_FACTOR represents the ageing factor between 0.0 and 1.0. In other words, the more recent the event, the more heavily it weighs in calculating the updated reputation score.

6.2 Project Partners Events Reception

Following the information presented in Chapter 5, it has been determined that the reputation system shall facilitate the reception of events from various project partners through a message queue. Specifically, the reputation system will employ the RabbitMQ tool for this purpose.

Under this system, the partners involved shall assume the role of producers, while the reputation system shall act as the consumer for all messages transmitted. The content of these messages may vary depending on the partner in question. Each may possess unique insights and information about the entities to be introduced or those existing within the reputation system. To facilitate the transmission of these messages, each partner shall send a respective message to a RabbitMQ exchange, which serves as a routing intermediary for a separate queue.

Each partner shall be held responsible for transmitting a JSON object containing

all pertinent information relating to the relevant entity. The reputation system shall filter the information contained within this object as necessary to facilitate the calculation of the entity's reputation, as discussed in Section 6.3. The reputation system shall receive the messages as a byte array, which the system shall subsequently convert to a JSON object for use in the reputation calculation.

6.3 Reputation Data Analysis & Calculation

Upon analysis of the received messages subject to filtering and the corresponding reputation calculation, the system converts the data from an array of bytes to a JSON object. The filtering process depends on the exchange from which the message originates, as each partner, functioning as a producer, transmits a JSON object containing specific information deemed relevant for updating the reputation value of an entity. Upon filtering the message by the exchange of origin, the entity is suitably recorded in a ReputationManager object utilising the HashMap outlined in section 6.1. This HashMap comprehends all entities within the system, thereby facilitating the computation of reputation values.

For illustrative purposes, consider the following JSON object received from the *middleware_exchange*, which serves as an instance of an event when a device successfully establishes a connection with the service:

```
{
    "Message: {
        "ClientId": "q9f3qplqzq",
        "Username": "q9f3qplqzq",
        "Protocol": "V311",
        "TimestampUTC": "2023-06-14T19:55:14.2098888Z"
    },
    "ArcadianId": "api.box2m.io:b666ca65-0faa-4e8b-a4bb",
    "Type": "Connected"
}
```

Listing 6.1: JSON object example of event received in *middleware_exchange*

In this particular case, the relevant information required to perform the reputation calculation is the entity identifier (i.e., the ArcadianId) and the verification that the device has connected to the service (as denoted by the "Type": "Connected" attribute). The reputation calculation can be carried out considering the successful connection of the device as a positive event. Therefore, the reputation is updated based on the pre-existing information stored in the HashMap of the ReputationManager type object.

Upon updating the entity's reputation, the updated information regarding the entity's reputation is transmitted to an exchange (*reputation_updates*) to enable the partners to access the updated information concerning the relevant entity from which the message was sent. The transmitted data consists of the entity's ID and former and current scores, as indicated in the JSON object below.

```
{
    "entityID": "api.box2m.io:b666ca65-0faa-4e8b-a4bb"
    "previousScore": 0.5
    "currentScore": 0.6
}
```

Listing 6.2: JSON object example of reputation information being sent to *reputation_updates* exchange

Following the transmission of reputation data to the *reputation_updates* exchange, the critical reputation data will require storage in a table within the Cassandra database for later use in the data visualisation platform. Specifically, the data that the system will store in the *entity_reputation* table of the Cassandra database will comprise the alpha, beta, old and current value of the entity reputation, as well as the entity ID (i.e., *ArcadianId*), the entity type, the type of each event (positive or negative), and the model utilised to calculate the entity reputation.

Figure 6.1 represents a flow diagram summarising how the reputation system behaves with the different messages, creating, analysing and calculating the reputation of the different entities introduced into the system.

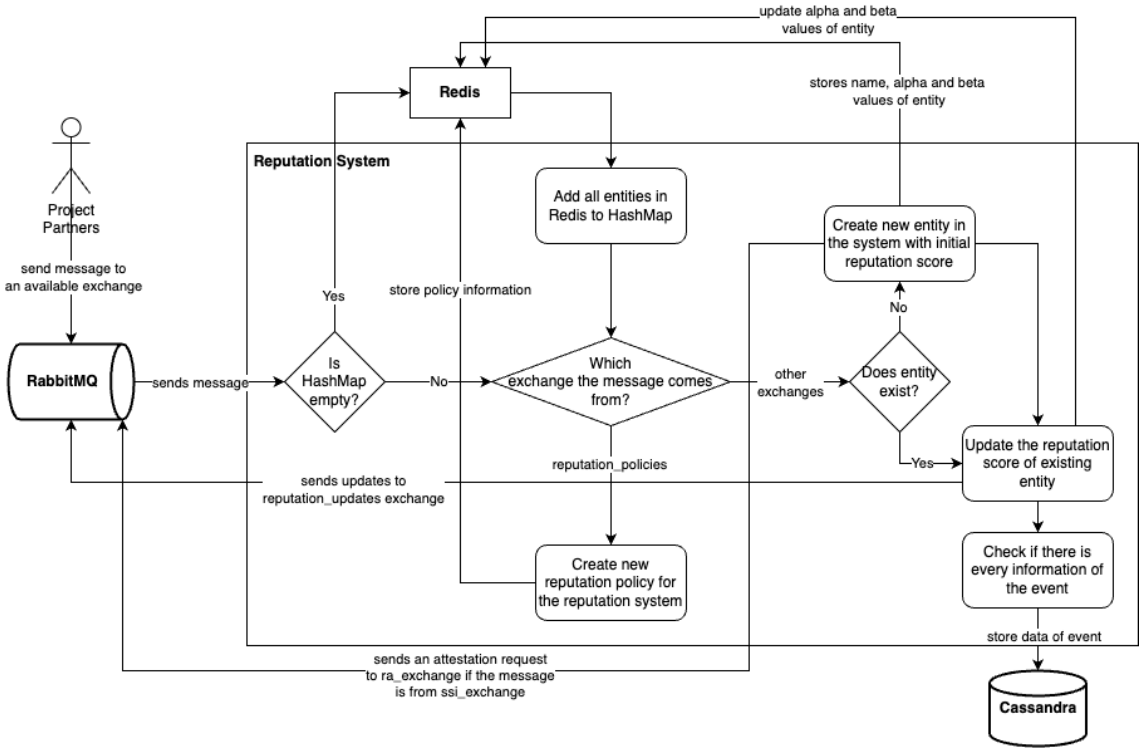


Figure 6.1: Flow diagram of the data analysis and calculation in the reputation system

Table 6.1 presents a list of all the exchanges that the system can receive messages from as well as the characteristics of each one, namely, which type of events are sent to a particular exchange, the reputation model that the system used to calculate the reputation of the entities and the type of entities are sent. Appendix B

presents more information about all the exchanges.

Name	Event Type	Reputation Model	Entity Type
nfm_exchange	Positive & Negative	Alpha-Beta & Severity Alpha-Beta	Not Specified
dbm_exchange	Negative	Alpha-Beta	Devices
ra_exchange	Positive & Negative	Alpha-Beta	Devices
naz_exchange	Positive & Negative	Alpha-Beta	Not Specified
bio_exchange	Positive & Negative	Alpha-Beta	Not Specified
ssi_exchange	Initialisation of Entity	None	Any
middleware_exchange	Positive & Negative	Alpha-Beta	Not Specified
sadp_exchange	Positive	Alpha-Beta	Devices

Table 6.1: Summary table for the important information from each exchange

6.4 Policies Used for Reputation

In conjunction with the reputation system and the methodology for determining entity reputation, an additional feature, namely the policy system, has been implemented. This system allows for including one or more policies within an entity. The primary objective of these policies is to establish a framework of predefined rules and regulations for the operation of the reputation system. This framework enhances overall consistency, ensures compliance with legal requirements, bolsters security measures, and promotes the adoption of best practices.

The reputation system's mechanism of incorporating these policies is akin to handling events about distinct entities accomplished through RabbitMQ exchanges. Specifically, the reputation_policies exchange serves as the conduit for transmitting such policies. To illustrate, an exemplar JSON object that initiates a policy creation within the system can be exemplified as follows:

```
{
    "createdBy": 1,
    "aiotIDs": ["api.box2m.io:b666ca65-0faa-4e8b-a4bb"],
    "infoSIM": "None",
    "action": 2,
    "actionRatio": 10,
    "minReputationScore": 0.2,
    "maxReputationScore": 1,
    "description": "test policy",
    "id": 4,
```

```
    "direction": "both",
    "event": "CREATE",
}
```

Listing 6.3: JSON object example of creation of a policy being sent to *reputation_policies* exchange

The policies above are stored and preserved for subsequent retrieval and association with diverse entities through an in-memory cache, namely Redis, employed within the framework of this reputation system. Among the elements encompassed within the received JSON object, the following aspects merit particular attention:

- *aiotIDs*: An array with IDs of the domain targeted by the policy that must include the AIoT identifiers;
- *infoSIM*: information to SIM that have the possible values of None, Self-Protection or Self-Recovery;
- *action*: integer value that action type value that can be either deny (1), throttle (2) and accept (3);
- *actionRatio*: integer with the ratio to reduce bandwidth. For instance, to reduce 10% of bandwidth;
- *minReputationScore*: float value representing the minimum reputation score that an entity must have to be affected by that policy;
- *maxReputationScore*: float value representing the maximum reputation score an entity must have to be affected by the policy.

The process of linking the information of a policy to a specific entity occurs upon receiving an event from a designated exchange. Following the completion of the reputation calculation, this association is established. To achieve this, a comparison is made between the entity's data and the policies stored in Redis. Specifically, the process verifies if the entity's ID is present within the array derived from the IDs impacted by the policy and if the reputation score falls within the specified range of minimum and maximum values. Provided that the entity fulfils all the conditions stipulated by the policy, its data is appended to the message destined for transmission to the reputation_updates exchange. To illustrate, an exemplar of the message dispatched to the exchange is as follows:

```
{
    "currentScore":0.6,
    "previousScore": 0.5,
    "entityID": "api.box2m.io:b666ca65-0faa-4e8b-a4bb",
    "infoSIM": "None",
    "action": 2,
    "actionRatio": 10
}
```

Listing 6.4: JSON object example of reputation information being sent to *reputation_updates* exchange with policy information

6.5 Data Visualisation Platform Development

This section comprehensively explores the implementation aspects of our data visualisation platform. The success of any data visualisation platform lies in its ability to effectively present complex information in an intuitive and visually appealing manner. To achieve this, we focus on two crucial components: frontend and backend implementation.

In the frontend implementation (Subsection 6.5.1), we discuss the strategies employed to create an engaging and user-friendly interface that enables users to interact with the data effortlessly.

Additionally, in the backend implementation (Subsection 6.5.2), we delve into the underlying architecture responsible for processing and organising the data, ensuring seamless and efficient display on the platform.

6.5.1 Frontend Implementation

In implementing the frontend for the visualisation platform, various visualisation tools integrated with React were utilised to optimise the user experience and enhance the platform's intuitiveness. By leveraging these tools, diligent efforts were made to ensure an interactive and user-friendly interface for seamless navigation and efficient data exploration. In this Section, we will delve into the most crucial frontend components of the platform, and figures of the entire platform interface are presented in Appendix C.

To realise the implementation of the table encompassing all entities, along with their corresponding type and reputation value, the *react-data-table-component*[2] was employed, as depicted in Figure 6.2. This component facilitated the organisation of entity information into distinct columns, including name, type, and current reputation value. Notably, it offered the ability to sort the entities based on these three values and provided the functionality to search within the table by entity name or type. Such features contributed to an enhanced user experience and facilitated efficient data exploration within the table.

[2] *react-data-table-component*: `https://react-data-table-component.netlify.app/`

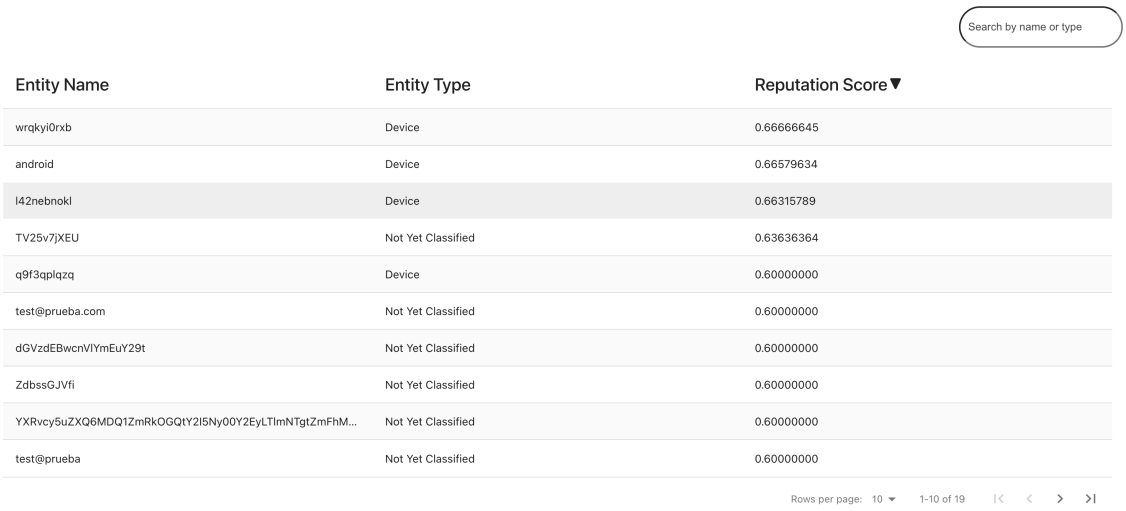


Figure 6.2: Table with all the entities in the reputation system

The implementation process of the parallel coordinate graph was accomplished by leveraging the capabilities of D3.js. This implementation defined three distinct axes: Rating, Action Type, and Date, with each axis representing the associated information from individual events of a certain entity. To enhance visual clarity and facilitate data exploration, brushing techniques were developed for each axis using D3.js, as seen in Figure 6.3. These brushing techniques allowed users to interactively select and focus on specific data ranges, rendering certain trends about an entity more intuitively observable. Using D3.js and incorporating brushing functionalities contributed to the graph's effectiveness in conveying complex data relationships and patterns through the polylines connecting the different axes.

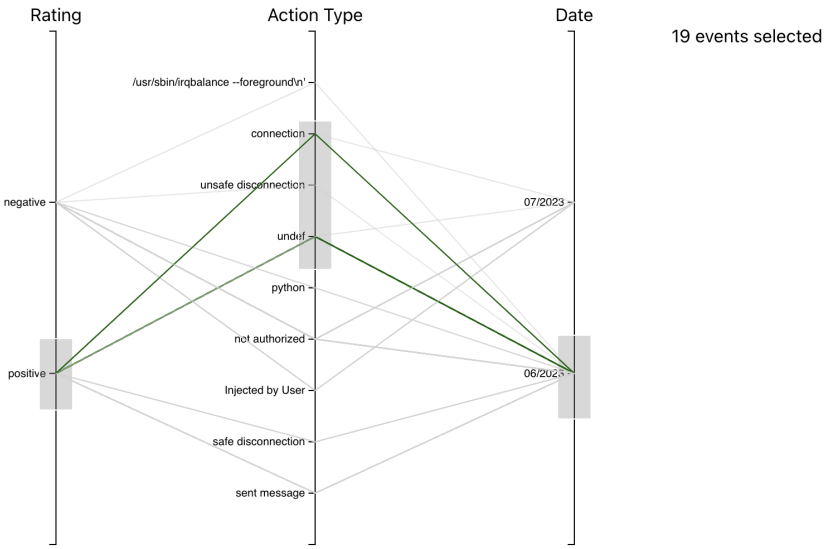


Figure 6.3: Parallel coordinate chart with brushed events

Analogous to the implementation approach adopted for the table containing all entities within the reputation system, the *react-data-table-component* was also cho-

sen for implementing the table with supplementary information about an entity's events. In this functionality, each row corresponds to an individual event. At the same time, the columns encompass the type of action, the model utilised to update the entity's reputation, the origin exchange from which the event emanated, and the date of occurrence, as noted in Figure 6.4. Notably, the date column was thoughtfully sorted to enable users to discern the chronology of events, facilitating the identification of the oldest or most recent occurrences. Additionally, users were empowered to perform event searches based on specific dates, further enhancing their ability to access relevant data within the table.

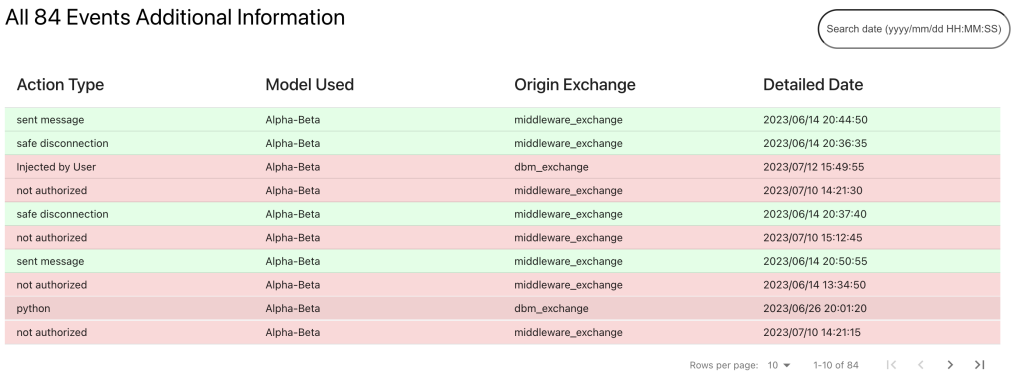


Figure 6.4: Additional information of an entity's events

To achieve the visualisation of the reputation score history since an entity's inception into the system, the D3.js library was used. The graph construction entails a continuous line delineating the evolution of the reputation score over time, with the x-axis denoting the date and the y-axis representing the reputation score, as depicted in Figure 6.5. Each event is represented as a distinct point on the line. Upon hovering over a particular point, the user gains access to detailed information, including the precise reputation score and the exact date of the corresponding event. This interactive feature facilitates a comprehensive understanding of the entity's reputation trajectory, offering users an intuitive means to explore and analyse historical data effectively.

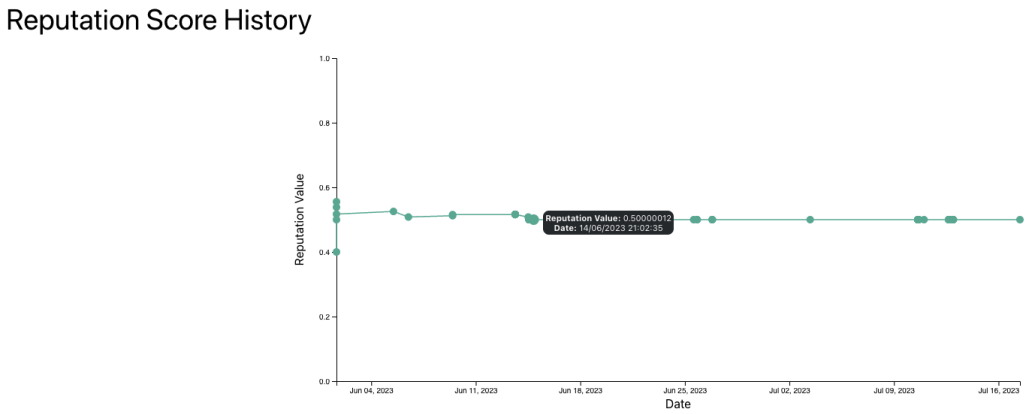


Figure 6.5: Chart with the reputation history of an entity in the reputation system

The development and integration of the statistics page within the reputation system are represented by creating a straightforward webpage, which presents comprehensive data about the total count of entities and events received by the system up to the present time, along with specific details concerning each entity category, as seen in Figure 6.6. Said information comprises the number of entities belonging to a given type and the mean, minimum, and maximum reputation scores associated with entities of the corresponding type.

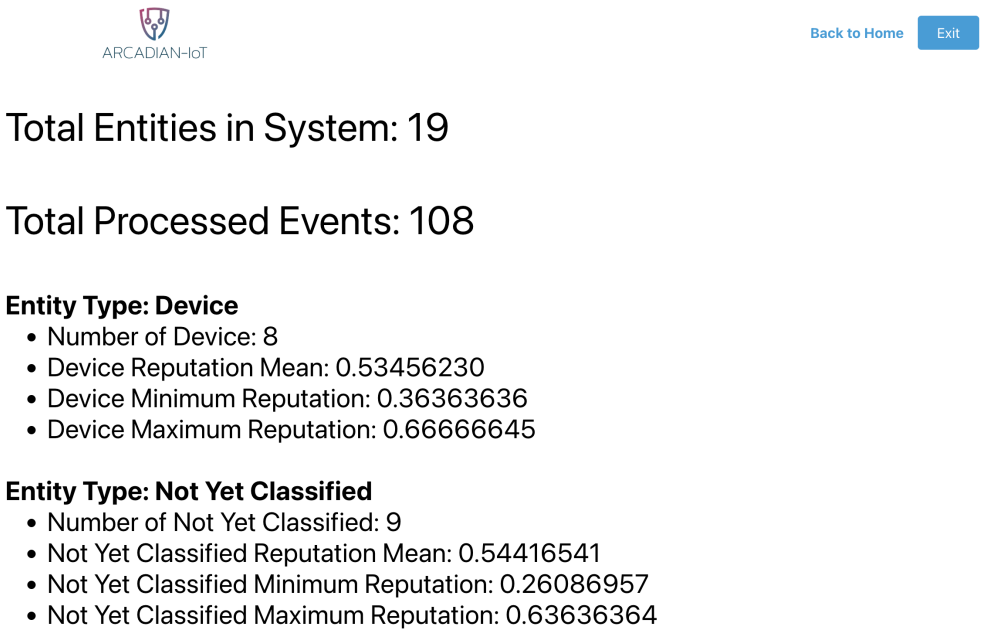


Figure 6.6: Page with the statistics of the reputation system

6.5.2 Display of Information in Platform

Concerning the backend of the data visualisation platform, various functionalities were employed to acquire the desired information from the platform. These functionalities encompassed both authentication procedures and data querying operations about the reputation system.

In terms of user authentication, an implementation utilising Firebase[3] authentication was employed. This tool facilitated the restriction of platform access to specific individuals, such as partners and various project members, while ensuring the segregation of paths for users lacking login privileges.

Regarding data querying operations directed towards the Cassandra database, multiple queries were executed using Node to acquire the requisite information for each visualisation component.

To construct the table containing all entities within the reputation system on the homepage, it was imperative to gather the ID, type, and current reputation value for each respective entity, which were then presented within the table.

[3] *Firebase*: `https://firebase.google.com/`

To acquire the comprehensive information required for presenting the parallel coordinate graph, it was essential to retrieve the rating, date, and type of action associated with each event from the database. Subsequently, the date was parsed into month and year format to facilitate its presentation within the graph.

To populate the table containing additional information for each event, it was imperative to retrieve specific data from the database from each entity. This included the type of action, the reputation calculation model utilised, the originating exchange, and the date of occurrence for each event associated with a particular entity. Furthermore, the event reception date was parsed to provide precise timestamp details for enhanced visualisation, encompassing year, month, day, hour, minute, and second.

The date and reputation value of each event about a specific organisation were gathered to construct the reputation history graph. Subsequently, these events were sorted chronologically based on their dates to streamline their presentation within the chart.

Each entity's type and current reputation value were obtained to compile the statistics information. A comprehensive count of all events and entities in the database was also conducted. Subsequently, the entities were filtered to calculate each entity type's mean, maximum, and minimum reputation values. This process facilitated the generation of meaningful insights regarding reputation metrics across different entity categories. In the simplified code example 6.5.2, we can see how the different entity types' minimum, maximum and mean reputation values can be calculated. Note that the *total_events*, *unique_types* and *all_entities* variables, although static in the example, result from analyses of the platform's database.

```
var total_events = 20; // total number of events processed
var unique_types = ['Person', 'Device'];
var all_entities = [
  {
      name: 'joaonunes',
      type: 'Person',
      reputation_score: 0.65
  },
  {
      name: 'mariasilva',
      type: 'Person',
      reputation_score: 0.4
  },
  {
      name: 'drone01',
      type: 'Device',
      reputation_score: 0.58
  },
  {
      name: 'iPhone',
      type: 'Device',
      reputation_score: 0.8
  }
];
var info = []; // array with information to display in platform

```

```
// Calculate min and max
for (var i = 0; i < unique_types.length; i++) {
    var dic = {}  // dictionary with info for each type
    dic["type"] = unique_types[i];
    dic["number"] = 0;
    dic["mean"] = 0;
    dic["min"] = 1;
    dic["max"] = 0;
    for (var j = 0; j < all_entities.length; j++) {
        if (all_entities[j]["type"] === unique_types[i]) {
            dic["number"]++;
            dic["mean"] += all_entities[j]["reputation_score"];
            if (dic["min"] > all_entities[j]["reputation_score"])
                dic["min"] = all_entities[j]["reputation_score"];
            if (dic["max"] < all_entities[j]["reputation_score"])
                dic["max"] = all_entities[j]["reputation_score"];
        }
    }
    info.push(dic);
}
// Calculate mean
for (var i = 0; i < info.length; i++) {
    info[i]["mean"] /= info[i]["number"];
}
var n = {"entities": unique_types.length, "events": total_events};
info = [n, ...info];
/*
    REQUEST OUTPUT
    info = [
        { entities: 2, events: 20 },
        {
            type: 'Person',
            number: 2,
            mean: 0.525,
            min: 0.4,
            max: 0.65
        },
        {
            type: 'Device',
            number: 2,
            mean: 0.69,
            min: 0.58,
            max: 0.8
        }
    ]
*/
```

Listing 6.5: Calculation of the statistical values of the reputation system

Chapter 7

Feature Testing & Validation

In the following chapter, we focus on testing and validating the features implemented in this project, previously referenced in Chapter 6, developed as part of this research. Testing and validation are essential in the software development process, as it helps ensure that the software is of high quality and meets the users' needs.

To conduct the testing and validation, we design a comprehensive set of test cases covering various scenarios and use cases. These test cases include input and expected output data, which we use to verify that the software or system functions correctly. We also look for any defects or issues with the software or system and document these if they are found.

7.1 Alpha-Beta Model Testing

This section aims to explain the analysis of the alpha-beta model, previously introduced in Chapter 2 and the implementation described in Chapter 6. This model assists in the calculation of the reputation of a particular entity after the reception of a new event, affecting the entity's reputation positively or negatively.

7.1.1 Testing the Model

A small script was made in Java to verify if the model is adequate to calculate and update the reputation value to test the alpha-beta model. With this, several tests were conducted where the value of the ageing factor was varied, a low, medium and high value, to see the influence of this factor in the reputation calculation. Next, random events (positive and negative) were created to determine the trend of alpha and beta values. Finally, with this information, the results were added to a Comma-Separated Values (CSV) file to be able to evaluate the alpha, beta, and mean values.

The tests done to verify the model were as follows:

- Ten negative events with an ageing factor of 0.5, to have an intermediate value of the effect of past events in the calculation of reputation;
- Ten negative events with an ageing factor of 0.2, a low value, meaning that the influence of the previous events is retained to a greater extent;
- Ten negative events with an ageing factor of 0.8, a high value, meaning that the previous value heavily influences the new value;
- Ten positive events with an ageing factor of 0.5;
- Ten positive events with an ageing factor of 0.2;
- Ten positive events with an ageing factor of 0.8;
- Four positive and three negative events with an ageing factor of 0.5;
- 20 random events with an ageing factor of 0.5.

7.1.2 Analysis of the Model

After testing the model and analysing the data, a few conclusions were reached.

Firstly, it was realised that, upon creation, the value of alpha and beta is always 1.0, with alpha representing positive events and beta representing negative events. When a positive or negative event is received, the growth of alpha and beta is logarithmic, respectively, despite the ageing factor value, as seen in Figure 7.1.

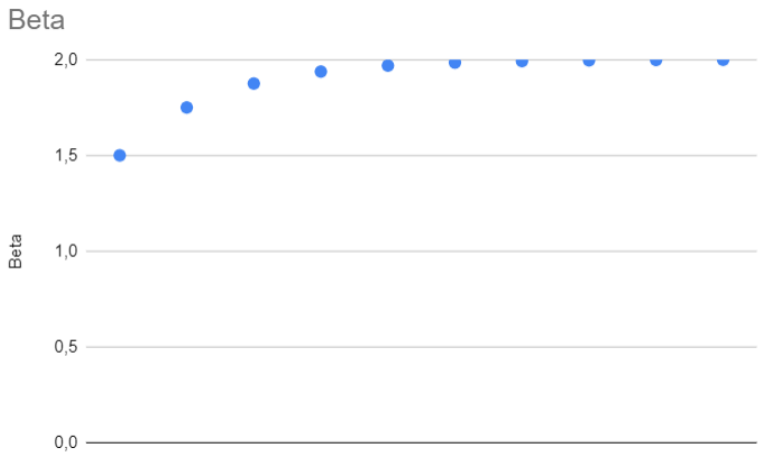


Figure 7.1: Logarithmic rise of the beta value of an entity

In terms of the ageing factor, this value affects alpha and beta values, meaning that growth is always the value of the age factor from the previous development. So, for example, if the ageing factor is 0.5, first it grows 0.5, then 0.25, then 0.125, and so on.

Finally, the mean goes down when it receives negative events and increases when positive ones are received. When negative events are received, the mean presents values closer to 0 and closer to 1 when positive events are received, as shown in Figure 7.2 with a logarithmic increase when successive positive events are received.

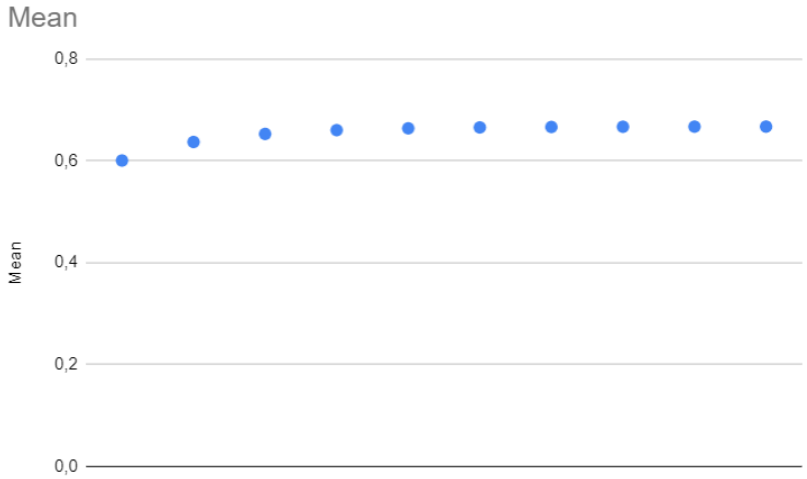


Figure 7.2: Logarithmic growth of mean when received positive events

Further information on the tests made with all graphics and tables are presented in Appendix A.

7.1.3 Conclusions of the Analysis

The alpha-beta model helps to calculate the reputation of entities in a system because it presents advantages, such as having access to the previous reputation value through the ageing factor. However, there should be a differentiation factor for reputation calculation, especially in negative events. For example, a person failing to log in on a system should not have the same impact as an attack attempt such as an SQL Injection, which, despite being both negative events, should not have the same effect on reputation calculation.

7.1.4 Changes Made to the Model

As stated in the preceding section, the binary assessment utilised in event processing and the consequent calculation of reputation is deemed inadequate owing to varying degrees of negative occurrences, some of which may have a more pronounced effect than others.

Therefore, the reputation update procedure of the Alpha-Beta model (referenced in 6.1.2) has been modified to address the unequal assessment of negative events. This technique will include a severity factor as an extra parameter, which must be included in the message transmitted by project partners, enabling the update of

the respective entity's reputation. The approach is similar to the previous method of updating reputation scores, except for updating the beta value in cases of negative events with a severity value instead of 1, as evidenced by equation 7.1.

$$\beta_i = (\beta_{i-1} * A) + S, \tag{7.1}$$

where A represents the ageing factor value between 0.0 and 1.0, and S is an integer value between 1 and 3 representing the event's severity.

The severity range between 1 and 3 was based on Atlassian's incident impact measures [4]. According to these measures, there can be three levels of severity for a given incident, where an incident with severity 1 is the most severe and an event with severity 3 is the least. For this model, we consider this measure, but it is applied in a reverse way, i.e. negative events with severity 3 are considered incidents with a very high impact (e.g., stolen phone) and negative events with severity 1 are considered incidents with a lower impact (e.g., incorrect password), similar to the negative events calculated through the Alpha-Beta model.

It is important to note that this particular methodology will solely be utilised to revise the standing of an entity if the event partners submit the message data indicating the event's severity. Without any indication of severity, the previously described approach should be employed without factoring in severity.

7.2 RabbitMQ & Reputation Information Testing

Following the testing section of the alpha-beta model, Section 7.1, it was necessary to test RabbitMQ and check how it was possible to take advantage of its features to send an entity's reputation information to calculate the reputation score later.

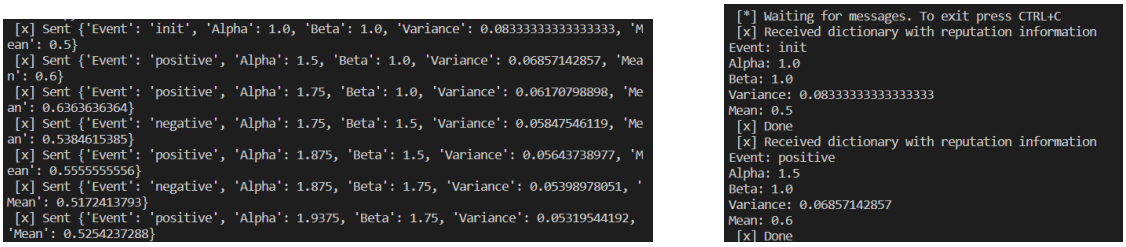
In a producer-consumer relationship, the producer is responsible for generating and sending data, while the consumer is responsible for receiving and processing that data. In the context of RabbitMQ, the producer is a Python script that reads a CSV file and sends JSON objects to the consumer using the Pika library[1]. In a dictionary, these JSON objects contain information such as the event, alpha, beta, variance, and mean associated with the respective values, each representing an entity's event.

In this case, the consumer receives the JSON objects and analyses their information. Using this information, the consumer can calculate a reputation score. This score is calculated based on the data collected in the JSON objects and is used to assess the reliability or trustworthiness of a particular event or source of information.

The producer-consumer relationship in RabbitMQ enables high flexibility and scalability in data processing. By decoupling the production and consumption of data, it is possible to add additional consumers as needed to handle data pro-

[1] *Pika*: `https://pika.readthedocs.io/en/stable/`

cessing. In this project, adding new consumers is viable if the partners provide a wide range of event information. This would allow each consumer to analyse the events from each partner and determine the best method for calculating the reputation score.



(a) Producer (b) Consumer

Figure 7.3: Producer sending JSON objects to Consumer in RabbitMQ

7.3 Quality Attributes

This section thoroughly examines the various tests conducted to evaluate the quality attributes of the software system under examination. Each subsection concentrates on a particular quality attribute and delves into the testing methodologies and techniques employed to guarantee the desired performance level of the reputation system. The reputation system underwent tests to assess its reliability, scalability, and security, while the data visualisation platform was subjected to security and usability evaluations.

7.3.1 Reliability

A comprehensive set of tests was undertaken to ascertain the reliability of the reputation system. Among these tests, a fundamental procedure entails temporarily ceasing the reputation system's operations, as Figure 7.4 depicts the number of connected clients to Redis. This simulation aims to mimic a scenario wherein the system experiences an offline state or encounters a period of dormancy. Throughout this designated period, the reputation information about the entities is diligently and securely stored within Redis, ensuring the preservation of data integrity and confidentiality.

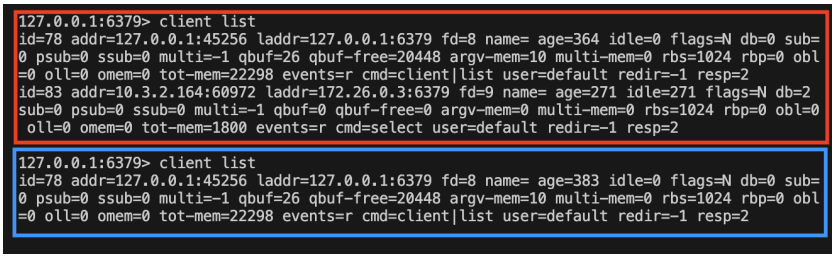


Figure 7.4: Clients connected to Redis before (red) and after (blue) system shutdown

Upon restarting the reputation system, a critical procedure is executed wherein all entities are systematically imported back into the designated HashMap that houses the repository of reputation information. This meticulous process guarantees the preservation of data integrity and mitigates the risk of data loss that may have occurred during the system shutdown phase. The system diligently reads the stored data from Redis, meticulously verifying and validating the accuracy of each reputation record. The seamless restoration of reputation records can be observed through the reputation system command line interface, as exemplified in Figure 7.5. By implementing such a robust mechanism, the reputation system ensures the reliable restoration of reputation data, instilling confidence in the system's ability to deliver accurate and consistent information.

By subjecting the reputation system to these tests, it can be guaranteed to withstand outages, recover, and resume normal operations without problems. These tests demonstrate that the system can remain stable and robust, preserve data integrity and provide consistent and reliable information to its users.

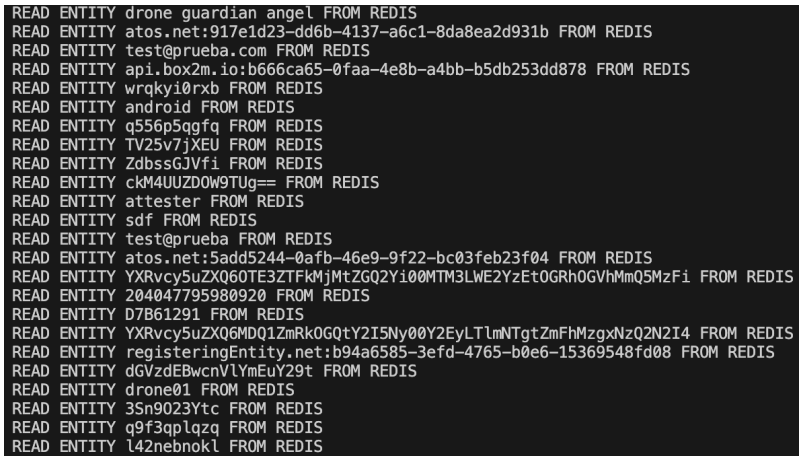


Figure 7.5: System importing back the entities from Redis

7.3.2 Scalability

A series of diverse experiments were conducted regarding scalability assessments of the reputation system. Initially, an evaluation assessed the system's response to a substantial influx of events emanating from multiple exchanges. The primary objective was to ascertain the system's capacity to efficiently process these incoming messages within a temporal threshold of one second, concurrently identifying the system's bottleneck. Subsequently, a comprehensive examination was executed to gauge how the reputation system computes reputation scores across distinct reputation models that have been implemented. This evaluation investigated the variability of reputation outcomes across the different models operating within an identical environment.

For the first set of tests, a producer was created in Python that generated messages to be sent to four different RabbitMQ exchanges, in this case, the middleware_exchange, dbm_exchange, bio_exchange and naz_exchange. In the reputation system, each message's processing time was counted, and the total time it

took the system to process all the messages sent per test was checked. The time it took the system to read all entities and their reputation data in Redis was also counted to assess possible bottlenecks.

Table 7.1 shows the important information generated in the events of each of the exchanges for each test, showing the number of events generated per exchange, whether the exchange introduces new entities into the system whether the events generated are random or deterministic and what type they are.

	middleware	**dbm**	**bio**	**naz**
Number of Events	25	25	25	25
Adds New Entities	No	Yes	No	No
Generated Events	Random	Deterministic	Random	Random
Type of Generated Events	Positive or Negative	Negative	Positive or Negative	Positive or Negative

Table 7.1: Information on the messages generated for each exchange in one test

The way these tests were carried out was first to restart the reputation system to import the data from Redis. Then 25 messages were sent from each exchange simultaneously, and the time it took the system to process the messages was counted in milliseconds. This process was done five times with the same parameters in each test, causing the number of entities present in Redis and the system's HashMap to increase, and the processing time is expected to increase in the next iteration.

	Test #1	**Test #2**	**Test #3**	**Test #4**	**Test #5**
Total Processing Time	438	538	528	643	731
Redis Processing Time	62	90	111	128	131
Processing Time Without Redis	376	448	417	515	600
Processing Event Time Mean	4.38	5.38	5.28	6.43	7.31
Processing Event Time Mean Without Redis	3.76	4.48	4.17	5.15	6

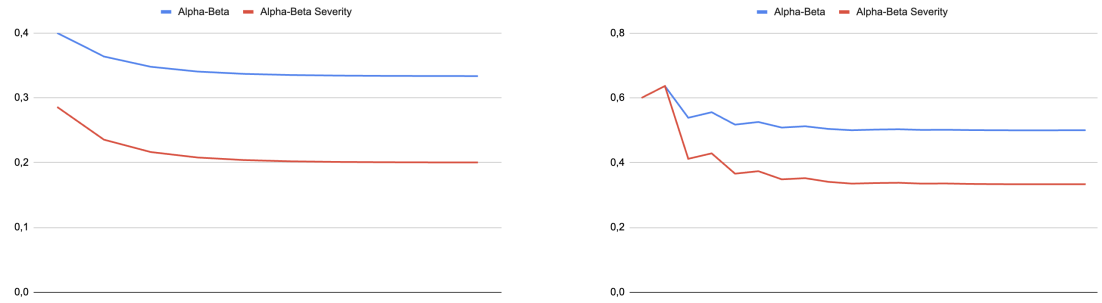
Table 7.2: Time measurements in the different tests, in milliseconds

As delineated in Table 7.2, it becomes apparent that the system has demonstrated the capacity to efficiently process all events within a punctual timeframe, precisely, within a duration of less than one second. Another inference that can be drawn pertains to the bottleneck inherent in retrieving entities through reading and subsequent reintegration into the system. This is exacerbated by the intrinsic characteristics of Redis, wherein an increase in the number of stored entities

directly correlates with an elongation in the processing interval. It is worth acknowledging that additional variables, such as querying the system's HashMap, wield a consequential impact on the temporal aspects of processing. Consequently, the processing duration devoid of Redis queries does not maintain a state of absolute constancy.

In the context of the second series of examinations, conducted to assess the impact of distinct reputation models, specifically the Alpha-Beta model and the Alpha-Beta model incorporating Severity considerations, a series of tests akin to those delineated in Section 7.1 were executed. The aim was to discern the fluctuations in reputation across the various models under scrutiny.

As illustrated in Figure 7.6, it is discernible that the severity model imposes a markedly augmented penalty concerning negative events. In the specific context of the tests conducted, events designated as negative with a severity factor of 2 were considered.



(a) 10 negative events

(b) 20 random events

Figure 7.6: Comparison of an entity's reputation growth according to the different models

The instance portrayed in Figure 7.6b offers an illustrative case wherein half of the events have a positive connotation while the remaining half embodies a negative one. By the Alpha-Beta model's anticipatory outcome, such an equilibrium would typically culminate in a reputation value of 0.5. Nonetheless, integrating a severity factor introduces a significant divergence, whereby negative events acquire a substantially amplified influence compared to their positive counterparts. Consequently, the reputation derived from a severity-modulated perspective is less than 0.5.

A comparative analysis of these two reputation models can also be conducted by testing if a particular entity is trustworthy or not, depending on the different reputation models used to calculate its reputation.

Let us examine an illustrative scenario in which the system presents a sequence of events consisting of 4 positive occurrences followed by two negative incidents originating from a specific entity. It is noteworthy that the negative occurrences have different severity values. As seen in Figure 7.7, it becomes evident that the system's imposition of penalties against the Alpha-Beta model incorporating Severity is significantly augmented. Furthermore, in alignment with the anal-

ysis of trustworthy entities, and under the premise that entities boasting reputation scores surpassing 0.5 are deemed trustworthy, while those below 0.5 are classified as untrustworthy, the entity subjected to reputation assessment via the Alpha-Beta model attains a status of trustworthiness (final reputation score of 0.5254). In stark contrast, the other two entities that used the Alpha-Beta model with Severity garnered an untrustworthy reputation status (final reputation score of 0.3735 with severity 2 and 0.2897 with severity 3).

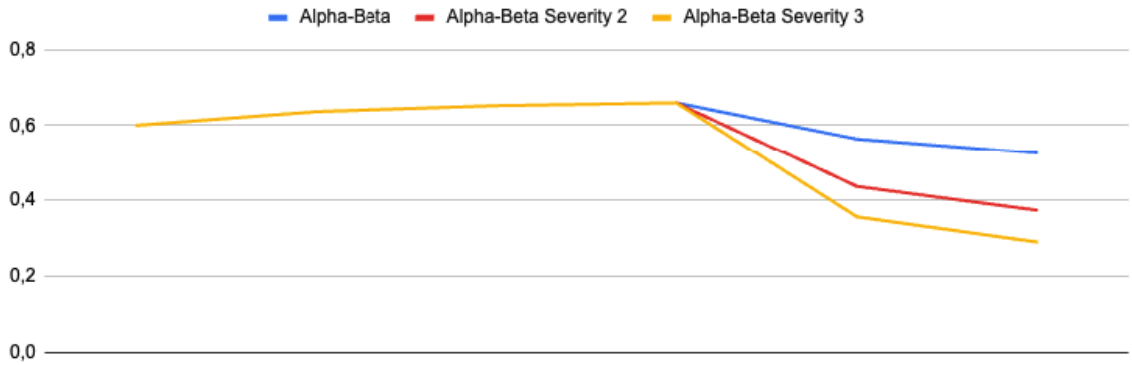


Figure 7.7: Comparison of reputation score with different severity factor

7.3.3 Security

Numerous security mechanisms were employed to uphold a degree of security within both the reputation system and the data visualisation platform.

Regarding the reputation system, it was intricately connected to RabbitMQ, Cassandra, and Redis – esteemed technological components integral to its functioning. A stringent authentication process was enforced to establish a secure connection, requiring the system to authenticate via username and password in each service. In conjunction with the authentication process, certification for accessing the Cassandra and Redis services was effectuated by utilising a valid certificate issued by the Helpdesk of Department of Informatics Engineering (DEI), as depicted in Figure 7.8. Concerning the RabbitMQ certification, despite incorporating provisions for secure communication, the actual testing with the system could not be carried out due to the production environment's non-utilisation of secure connections.

```
admin@AIoT-RS-prod:~/AIoTReputationSystem-main$ openssl x509 -in certs/ssl/aiot-rs-prod.dei.uc.pt.crt -noout -enddate
notAfter=Jun  5 23:59:59 2024 GMT
admin@AIoT-RS-prod:~/AIoTReputationSystem-main$ openssl verify -CAfile certs/ssl/GEANT-CA.crt certs/ssl/aiot-rs-prod.dei.uc.pt.crt
certs/ssl/aiot-rs-prod.dei.uc.pt.crt: OK
```

Figure 7.8: Commands demonstrating the validity of the SSL certificate

Two measures were implemented to enhance the security aspects of the data visualisation platform to enhance its overall security. As described in Section 6.5.2, the initial measure involved the incorporation of Firebase authentication. This method ensures that only users with appropriate permissions can access the platform. The assignment of these permissions is the responsibility of the platform

administrators, who are tasked with creating unique email and password authentications for each project partner seeking access to the platform. These credentials are managed through the Firebase console.

By employing this authentication mechanism, the platform can restrict access to various pages containing sensitive information, thus providing an additional layer of confidentiality. This measure aims to safeguard the data and ensure it remains accessible solely to authorised personnel.

The second pivotal security measure entailed the integration of an Secure Sockets Layer (SSL) certificate to facilitate HTTPS support for the data visualisation platform. The consensus was reached to utilise the same certificate employed for the certification of Cassandra and Redis, as earlier stipulated. Figure 7.9 demonstrates the successful implementation of this security component, accomplished through uncomplicated commands in the deployed platform configuration file.

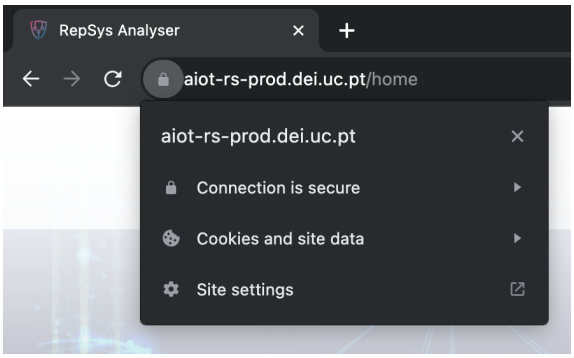


Figure 7.9: Data visualisation platform with HTTPS

The deployment involved configuring the platform to listen to port 443 and incorporating the parameters *ssl_certificate* and *ssl_certificate_key* to identify the certificate and key, respectively. These adjustments were reiterated as necessary to ensure the effective activation of the SSL certificate functionality. By incorporating SSL, the platform's data transmission over the network is encrypted, bolstering the overall security posture and safeguarding sensitive information from potential threats.

7.3.4 Usability

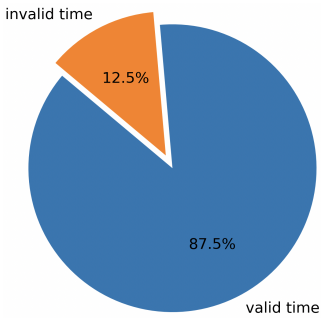
To test the usability of the data visualisation platform, usability tests were carried out on different people to verify the quality of the developed product. The way these tests were carried out was iterative, that is, two to three usability tests were performed, and, according to the problems and suggestions pointed out by the testers, these were changed so that the same issues were not pointed out by the following testers again. This form of testing was used because, according to Nielsen [28], as we test with the same prototype, the level of redundancy will increase because the same problems will be pointed out. In this way, we can also achieve less expenditure of resources for elaborating the tests.

In these tests, a questionnaire was conducted to verify the tester's experience when trying the platform. Initially, it was necessary to know the platform tester's previous knowledge of data visualisation and distributed or reputation systems. Then, the testers were asked to do a set of tasks, for each of which the time taken to do each task was timed and checked if it was done promptly, which was previously estimated. Finally, after completing all the tasks, the testers answered questions expressing their opinions about their platform experience. These questions focused on the navigation and interface of the platform, the interaction and perception of the graphics, and the best and worst aspects of the platform.

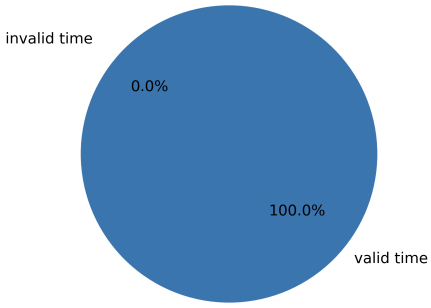
About the tasks executed by the platform testers, it was stipulated that prior login was a prerequisite. The following tasks were proposed for their evaluation and assessment:

- **Task #1**: Find the entity with the lowest reputation on the homepage.
- **Task #2**: On the homepage, search for the entity *api.box2m.io:b666ca65-0faa-4e8b-a4bb-b5db253dd878* and click on it.
- **Task #3**: On the entity page, check how many positive events of the connection type there are in June of 2023.
- **Task #4**: On the entity page, check the event type of the most recent event received.
- **Task #5**: On the entity's page, check the event type of June 5th, 2023 and its reputation value through the reputation history graph.
- **Task #6**: Return to the homepage and look for the statistics of the reputation system and see how many events the system has received to date.

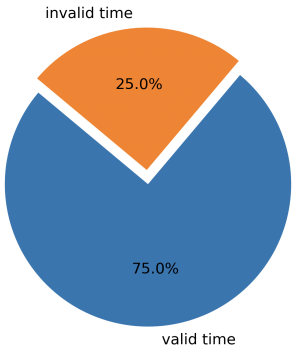
Regarding the conducted tests, the individuals participating in the platform testing encompassed a diverse group, ranging from bachelor's to master's students in Informatics Engineering and Data Science and Engineering. The questions asked before the tasks were carried out showed that the participants had good visualisation knowledge and demonstrated a background in distributed systems. However, in some cases, explaining the concept of reputation systems was necessary. Concerning the outcomes derived from the testers' performance evaluation, as evident from the graphical representations in Figure 7.10, most users completed all assigned tasks within satisfactory timeframes. However, it is worth noting that only a few testers exhibited relatively extended duration in tackling the more intricate tasks, such as Task #3 and #5.



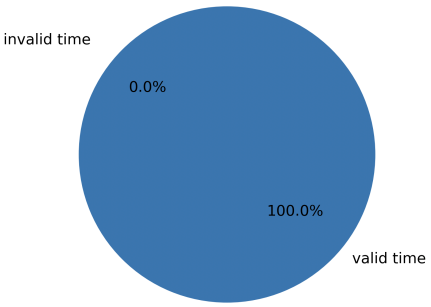
(a) Task #1



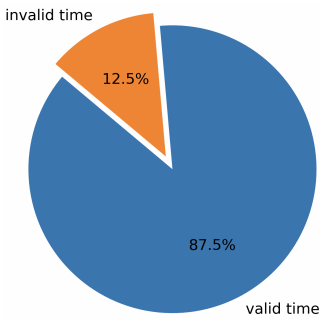
(b) Task #2



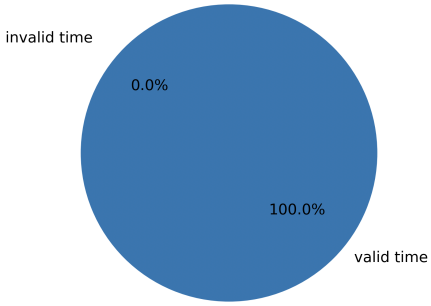
(c) Task #3



(d) Task #4



(e) Task #5



(f) Task #6

Figure 7.10: Valid time taken for each task - blue in valid time and orange invalid

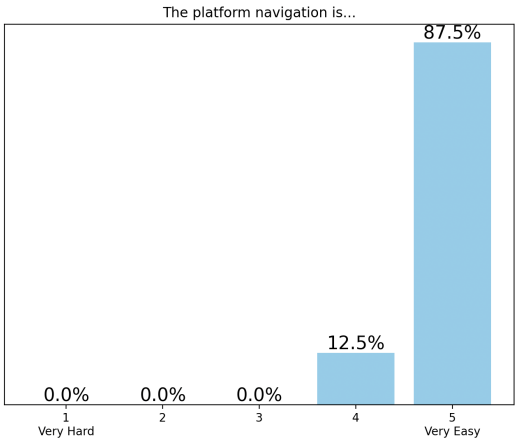
After completing the assigned tasks, the testers were invited to express their viewpoints concerning their platform usage experience. As depicted in Figure 7.11, the feedback about the platform was predominantly positive, particularly concerning aspects such as platform navigation and the interpretation of various graphics.

Lastly, regarding the platform's favourable and unfavourable attributes, the testers have identified several positive aspects worth noting. Specifically, the intuitive nature of the platform was commended, attributed to its systematic division into well-defined sections, each serving a distinct purpose. Additionally, the organisation and efficacy of graph and table filtering functionalities were highlighted as commendable features.

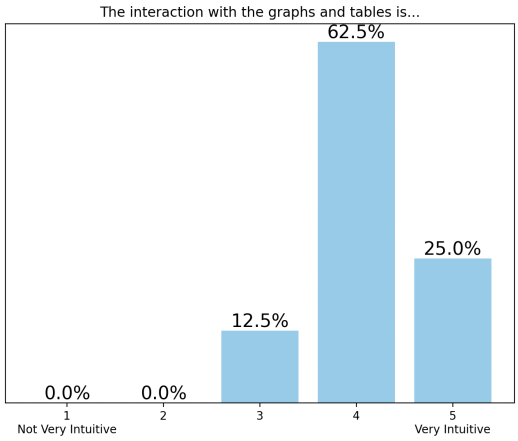
Regarding the negative aspects identified during the assessment, these critical observations underscore the significance of effecting modifications to the plat-

form. The concerns raised by various testers were diverse, necessitating a refined and tailored approach to address each distinct issue comprehensively. Notably, the intuitiveness of the date-filtering functionality on the entity's page emerged as an area warranting improvement. Furthermore, the comprehensibility of the date representation on the reputation history graph was recognised as a point of ambiguity. Lastly, the explicitness of the table sorting mechanism was cited as an area of potential enhancement.

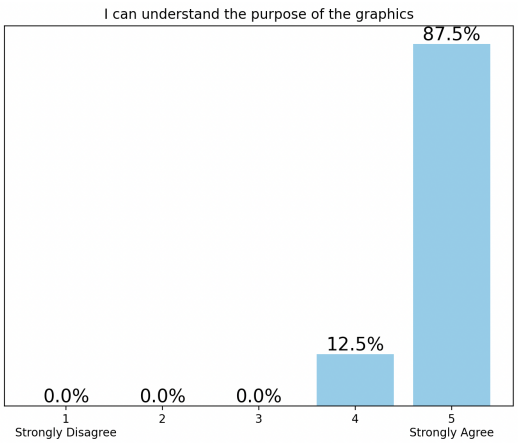
Implementing an iterative usability testing process was pivotal in the platform's refinement journey. By incorporating feedback from successive testers, adjustments were made to tackle the identified problem areas effectively. Consequently, subsequent rounds of testing revealed a discernible improvement, as the new testers no longer highlighted the same issues previously encountered. This iterative approach has proven instrumental in ensuring the platform evolves with user requirements, ultimately enhancing its overall usability and user satisfaction.



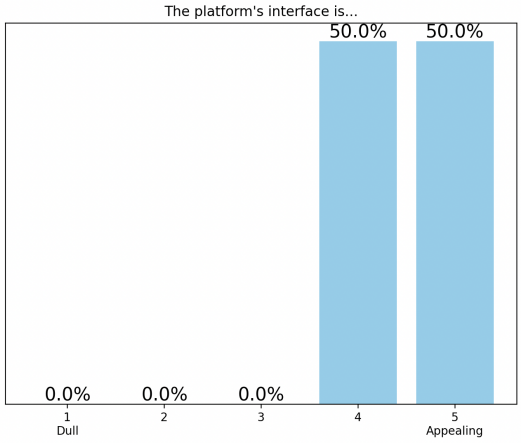
(a) Platform navigation



(b) Graphics and tables interaction



(c) Graphics perception



(d) Platform interface

Figure 7.11: Graphics depicting the opinion of the testers regarding several platform functionalities

Chapter 8

Conclusion

This chapter aims to reflect on the work done during the whole project, presenting the knowledge that the work provided, what was accomplished and the possible difficulties experienced during the year.

Throughout the first semester, we learned various frameworks that were new to us, such as RabbitMQ. In addition, the research we conducted helped us gain a deeper understanding of reputation systems, a topic that we had little knowledge about before the start of the project.

As for the second semester, we had the opportunity to learn how Java Spark works and how it interacts with other tools such as RabbitMQ, a Cassandra database and Redis. There was also the opportunity to broaden our knowledge of web development through tools such as React and Node.js and data visualisation through D3.js, allowing us to implement all the features listed for the data visualisation platform.

As indicated throughout the document, the project encountered delays during the execution phase of the reputation system, primarily attributable to the non-adherence to specified deadlines. Nonetheless, the strategic decision to extend the dissertation submission deadline to September proved instrumental in mitigating these setbacks, culminating in the realisation of a project of enhanced quality and heightened success.

The process of gathering requirements and designing the project architecture relied heavily on research from past ARCADIAN-IoT project deliveries and other projects that utilised similar components. This research allowed us to identify best practices and potential challenges, enabling us to make informed decisions and create a solid foundation for the project.

During the project, we encountered some difficulties. In investigating how the reputation system's architecture would work, it stands out how RabbitMQ could receive events from multiple exchanges. To address this challenge, we conducted thorough research on examples of how a multithreaded model could read from various exchanges. This required considerable time and effort but ultimately allowed us to successfully implement this aspect of the project.

During the implementation phase, notable challenges within the context of the reputation system pertain to specific functionalities of Java Spark, particularly concerning tasks such as comprehensive data storage for events within the database. Likewise, security-related concerns, encompassing the generation of certificates for utilised tools, constitute an additional facet of these difficulties.

Concerning the complexities encountered within the data visualisation platform, impediments were predominantly attributed to limited familiarity with the chosen frameworks. This encompasses challenges in harnessing D3.js for the implementation of graphical representations and tabular displays, as well as navigating the employment of Node.js for seamless interaction with the Cassandra database.

8.1 Future Work

This dissertation presents a wide range of future work for both the reputation system and the data visualisation platform.

Regarding the reputation system, a pivotal area of prospective endeavour resides in the forthcoming realisation of an optimised system upon the culmination of the ARCADIAN-IoT project. For instance, a project partner initiates entities by messaging *ssi_exchange*, while other exchanges manage events for these entities, updating their reputation through event analysis.

An additional aspect for future work within the scope of the reputation system involves incorporating a broader spectrum of reputation models into the existing framework, such as the dominance model. Given that this constitutes a central facet of innovation within the project, it will be essential to filter what kind of model will be used to calculate the reputation of a particular entity. For instance, the consideration might extend towards adopting a dominance model for persons or integrating an Alpha-Beta model with Severity considerations for devices.

A notably ambitious trajectory for the prospective evolution of the reputation system entails its potential to facilitate a dynamic selection mechanism for the constituent elements employed in computing reputation scores, as communicated by collaborating project partners. To elaborate, this would encompass the partners sending an initial message detailing the specific factors deemed pertinent for the reputation calculation. The system would subsequently preserve these identified factors, thereby enabling the subsequent measure of the reputation during the transmission of events by the salient factors outlined by the partners within the initial message.

Regarding future work for the data visualisation platform, we can highlight some suggestions brought up by the platform testers during the usability tests. Notably, these suggestions encompass enhancements such as the entity page presenting the date filter of the table with additional information on the events in the form of a calendar. Furthermore, the augmentation of graphical representations on the statistics page emerges as another significant proposition put forth by the testers.

Finally, it is expected that the writing of the scientific article with the same theme as this document will be completed, in which all the information and results obtained on data processing and visualisation in reputation systems will be compiled concisely and informatively.

References

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-grid: a self-organizing structured p2p system. *ACM SiGMOD Record*, 32(3):29–33, 2003.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[3] E. Adar and B. A. Huberman. Free riding on gnutella. *First monday*, 2000.

[4] Atlassian. Understanding incident severity levels. `https://www.atlassian.com/incident-management/kpis/severity-levels`, 2019. Accessed on August 2023.

[5] F. Bao and J. Chen. Visual framework for big data in d3.js. In *2014 Ieee Workshop on Electronics, Computer and Applications*, pages 47–50. IEEE, 2014.

[6] E. Damiani, D. C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 207–216, 2002.

[7] C. Dellarocas. The digitization of word of mouth: Promise and challenges of online feedback mechanisms. *Management science*, 49(10):1407–1424, 2003.

[8] R. Farmer and B. Glass. *Building web reputation systems*. "O'Reilly Media, Inc.", 2010.

[9] X. Fu, K. Yue, L. Liu, Y. Feng, and L. Liu. Reputation measurement for online services based on dominance relationships. *IEEE Transactions on Services Computing*, 14(4):1054–1067, 2018.

[10] Gnutella. Gnutella - a protocol for a revolution. `https://rfc-gnutella.sourceforge.net/`, 2003.

[11] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 127–130. IEEE, 2002.

[12] F. Hendrikx, K. Bubendorfer, and R. Chard. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184–197, 2015.

[13] T. D. Huynh, N. R. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. 2004.

[14] A. Jøsang. Trust and reputation systems. In *Foundations of security analysis and design IV*, pages 209–245. Springer, 2007.

[15] A. Josang and R. Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, volume 5, pages 2502–2511, 2002.

[16] C. A. S. Júnior. A privacy preserving system to consult public institutions records. Master's thesis, University of Coimbra, 2020.

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[18] E. Koutrouli and A. Tsalgatidou. Reputation-based trust systems for p2p applications: design issues and comparison framework. In *International conference on trust, privacy and security in digital business*, pages 152–161. Springer, 2006.

[19] E. Koutrouli and A. Tsalgatidou. Taxonomy of attacks and defense mechanisms in p2p reputation systems—lessons for reputation system designers. *Computer Science Review*, 6(2-3):47–70, 2012.

[20] L. Liu and W. Shi. Trust and reputation management. *IEEE Internet Computing*, 14(5):10–13, 2010.

[21] Z. Malik and A. Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. *The VLDB Journal*, 18(4):885–911, 2009.

[22] S. Marti and H. Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[23] P. Michiardi and R. Molva. Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Advanced communications and multimedia security*, pages 107–121. Springer, 2002.

[24] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th annual Hawaii international conference on system sciences*, pages 2431–2439. IEEE, 2002.

[25] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of reputation in multiagents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 280–287, 2002.

[26] L. Nair, S. Shetty, and S. Shetty. Interactive visual analytics on big data: Tableau vs d3. js. *Journal of e-Learning and Knowledge Society*, 12(4), 2016.

[27] Napster. Napster: Music from every angle. `https://www.napster.com/pt`, Nov 2022.

[28] J. Nielsen. Why you only need to test 5 users. `https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`, 2000. Accessed on July 2023.

[29] PoSeID-on. Protection and control of secured information by means of a privacy enhanced dashboard poseid-on. `https://www.poseidon-h2020.eu/`, Jul 2019.

[30] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[31] M. Richards and N. Ford. *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 2020.

[32] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents*, pages 194–195, 2001.

[33] E. Salituro. *Learn Grafana 7.0: A beginner's guide to getting well versed in analytics, interactive dashboards, and monitoring*. Packt Publishing Ltd, 2020.

[34] J. Sänger and G. Pernul. Visualizing transaction context in trust and reputation systems. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 94–103. IEEE, 2014.

[35] W. Teacy, J. Patel, N. R. Jennings, and M. Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

[36] Truphone. Autonomous trust, security and privacy management framework for iot - d2.2: Use case specification, 2021.

[37] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE transactions on Knowledge and Data Engineering*, 16(7):843–857, 2004.

[38] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms for electronic marketplaces. *Decision support systems*, 29(4):371–388, 2000.

Appendices

Appendix A

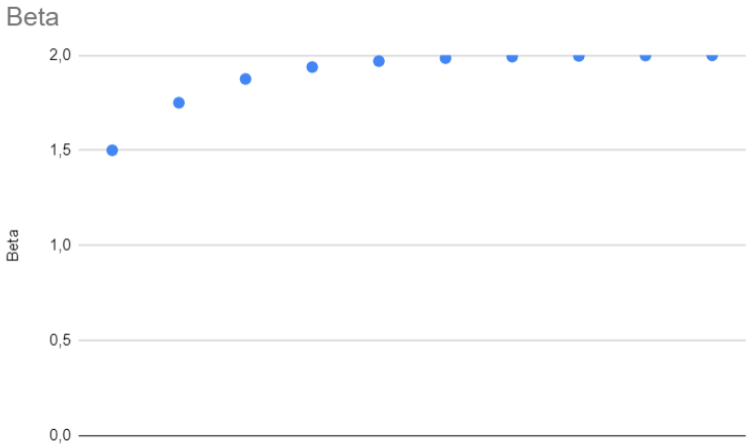
Alpha-Beta Model Validation Tests

This appendix presents the validation tests for implementing the Alpha-Beta Model (explained in Section 7.1). This includes the tables and graphics used to perform the analysis discussed in Subsection 7.1.2 for the alpha, beta, and mean values.

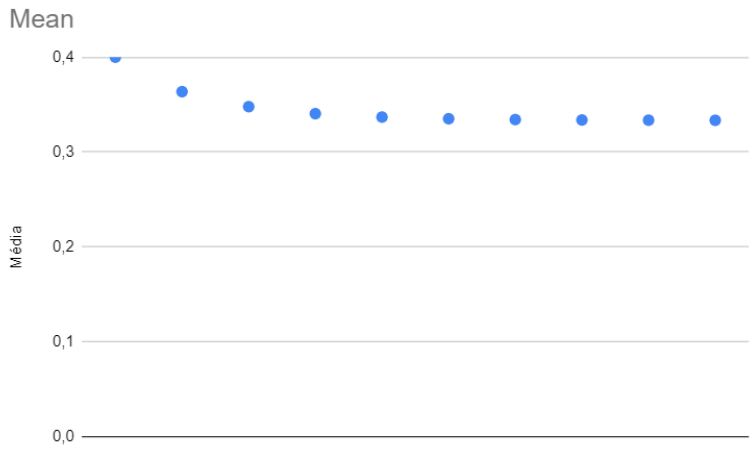
A.1 Ten negative events with an ageing factor of 0.5

Event	Alpha	Beta	Mean
negative	1.0	1.5	0.4
negative	1.0	1.75	0.3636363636
negative	1.0	1.875	0.347826087
negative	1.0	1.9375	0.3404255319
negative	1.0	1.96875	0.3368421053
negative	1.0	1.984375	0.335078534
negative	1.0	1.9921875	0.3342036554
negative	1.0	1.99609375	0.333767927
negative	1.0	1.998046875	0.3335504886
negative	1.0	1.999023438	0.3334418756

Table A.1: 10 negative events with AGING_FACTOR=0.5



(a) Beta



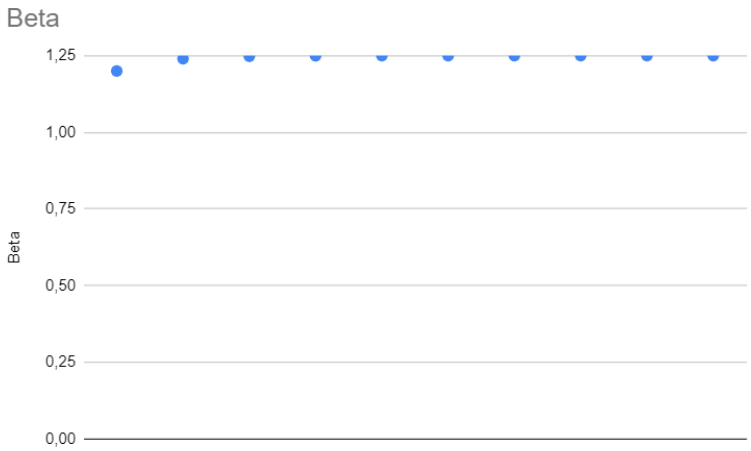
(b) Mean

Figure A.1: Graphics of the model's behaviour with ten negative events with AGING_FACTOR=0.5

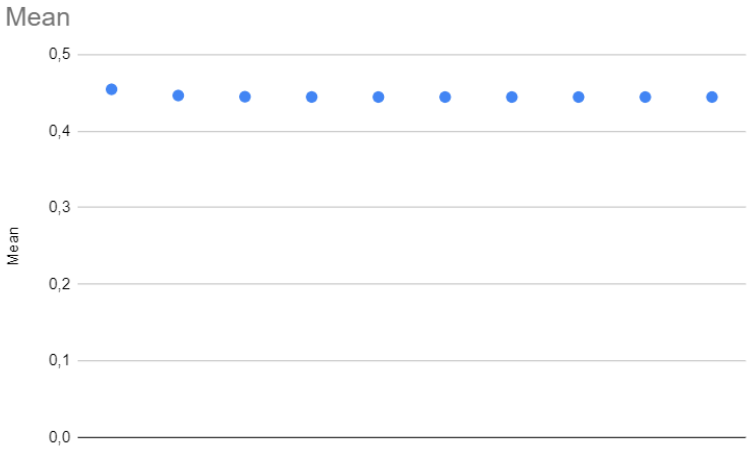
A.2 Ten negative events with an ageing factor of 0.2

Event	Alpha	Beta	Mean
negative	1.0	1.2	0.4545454545
negative	1.0	1.24	0.4464285714
negative	1.0	1.248	0.4448398577
negative	1.0	1.2496	0.4445234708
negative	1.0	1.24992	0.4444602475
negative	1.0	1.249984	0.444447605
negative	1.0	1.2499968	0.4444450765
negative	1.0	1.24999936	0.4444445709
negative	1.0	1.249999872	0.4444444697
negative	1.0	1.249999974	0.4444444495

Table A.2: 10 negative events with AGING_FACTOR=0.2



(a) Beta



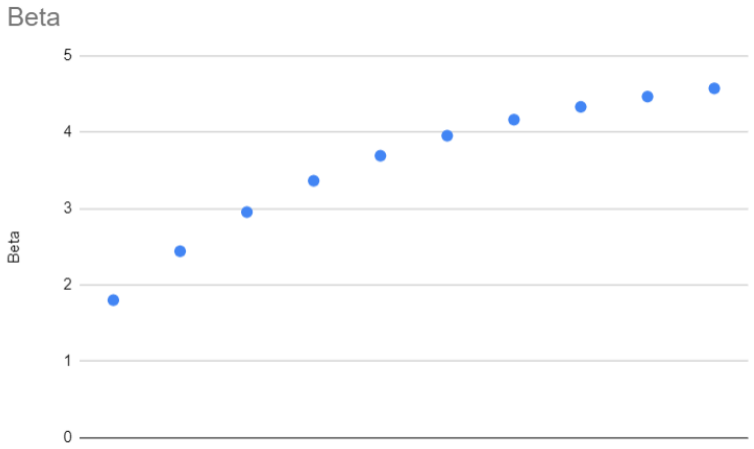
(b) Mean

Figure A.2: Graphics of the model's behaviour with ten negative events with AGING_FACTOR=0.2

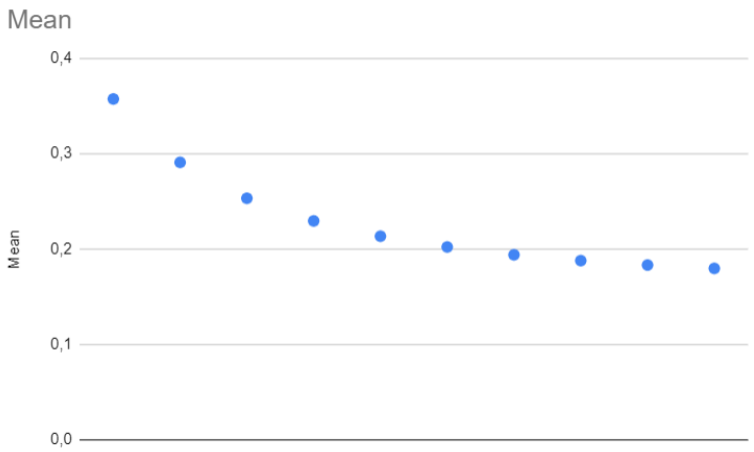
A.3 Ten negative events with an ageing factor of 0.8

Event	Alpha	Beta	Mean
negative	1.0	1.8	0.3571428571
negative	1.0	2.44	0.2906976744
negative	1.0	2.952	0.2530364372
negative	1.0	3.3616	0.229273661
negative	1.0	3.68928	0.2132523543
negative	1.0	3.951424	0.2019621022
negative	1.0	4.1611392	0.1937556732
negative	1.0	4.32891136	0.187655589
negative	1.0	4.463129088	0.1830452812
negative	1.0	4.57050327	0.1795169936

Table A.3: 10 negative events with AGING_FACTOR=0.8



(a) Beta



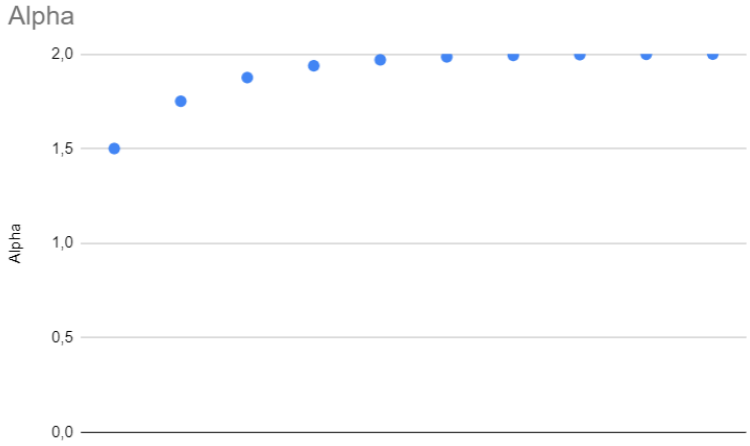
(b) Mean

Figure A.3: Graphics of the model's behaviour with ten negative events with AGING_FACTOR=0.8

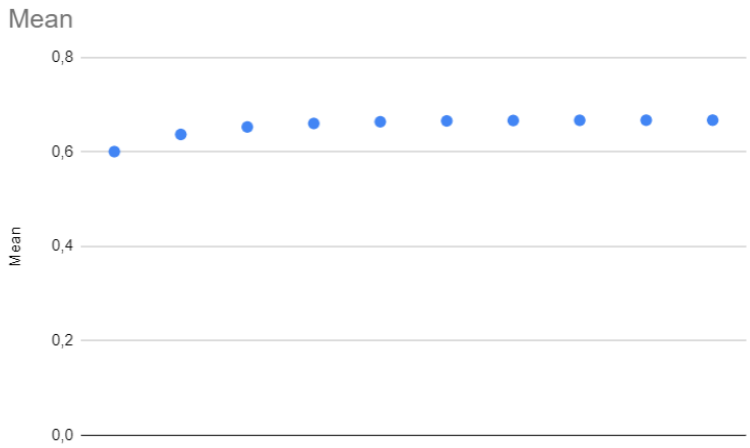
A.4 Ten positive events with an ageing factor of 0.5

Event	Alpha	Beta	Mean
positive	1.5	1.0	0.6
positive	1.75	1.0	0.6363636364
positive	1.875	1.0	0.652173913
positive	1.9375	1.0	0.6595744681
positive	1.96875	1.0	0.6631578947
positive	1.984375	1.0	0.664921466
positive	1.9921875	1.0	0.6657963446
positive	1.99609375	1.0	0.666232073
positive	1.998046875	1.0	0.6664495114
positive	1.999023438	1.0	0.6665581244

Table A.4: 10 positive events with AGING_FACTOR=0.5



(a) Alpha



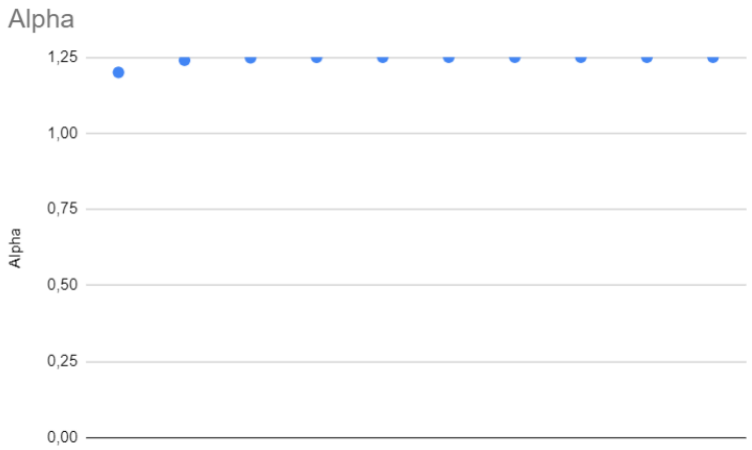
(b) Mean

Figure A.4: Graphics of the model's behaviour with ten positive events with AGING_FACTOR=0.5

A.5 Ten positive events with an ageing factor of 0.2

Event	Alpha	Beta	Mean
positive	1.2	1.0	0.5454545455
positive	1.24	1.0	0.5535714286
positive	1.248	1.0	0.5551601423
positive	1.2496	1.0	0.5554765292
positive	1.24992	1.0	0.5555397525
positive	1.249984	1.0	0.555552395
positive	1.2499968	1.0	0.5555549235
positive	1.24999936	1.0	0.5555554291
positive	1.249999872	1.0	0.5555555303
positive	1.249999974	1.0	0.5555555505

Table A.5: 10 positive events with AGING_FACTOR=0.2



(a) Alpha

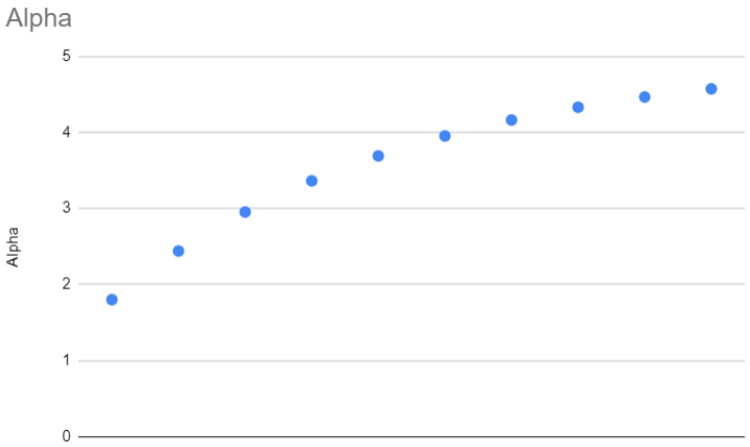
(b) Mean

Figure A.5: Graphics of the model's behaviour with ten positive events with AGING_FACTOR=0.2

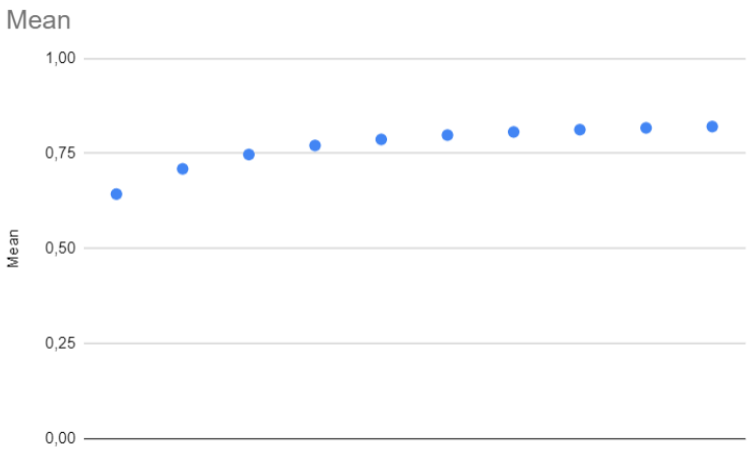
A.6 Ten positive events with an ageing factor of 0.8

Event	Alpha	Beta	Mean
positive	1.8	1.0	0.6428571429
positive	2.44	1.0	0.7093023256
positive	2.952	1.0	0.7469635628
positive	3.3616	1.0	0.770726339
positive	3.68928	1.0	0.7867476457
positive	3.951424	1.0	0.7980378978
positive	4.1611392	1.0	0.8062443268
positive	4.32891136	1.0	0.812344411
positive	4.463129088	1.0	0.8169547188
positive	4.57050327	1.0	0.8204830064

Table A.6: 10 positive events with AGING_FACTOR=0.8



(a) Alpha



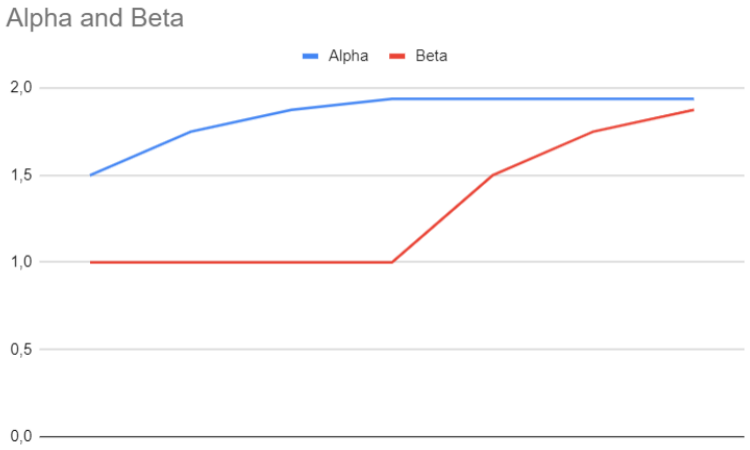
(b) Mean

Figure A.6: Graphics of the model's behaviour with ten positive events with AGING_FACTOR=0.8

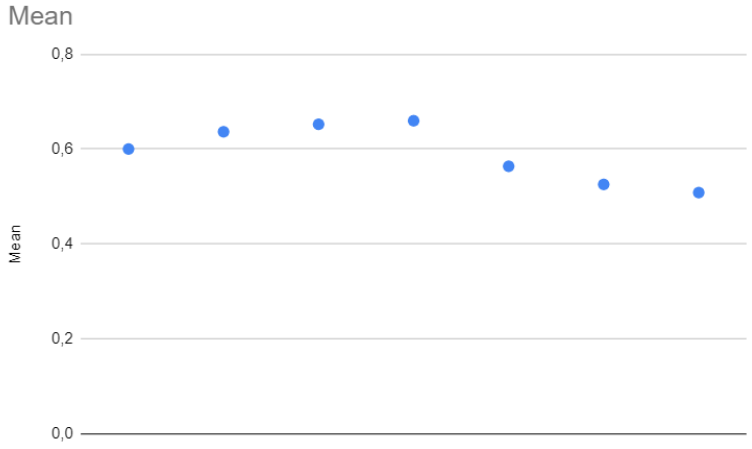
A.7 Four positive and three negative events with an ageing factor of 0.5

Event	Alpha	Beta	Mean
positive	1.5	1.0	0.6
positive	1.75	1.0	0.6363636364
positive	1.875	1.0	0.652173913
positive	1.9375	1.0	0.6595744681
negative	1.9375	1.5	0.5636363636
negative	1.9375	1.75	0.5254237288
negative	1.9375	1.875	0.5081967213

Table A.7: 4 positive and 3 negative events with AGING_FACTOR=0.5



(a) Alpha and Beta



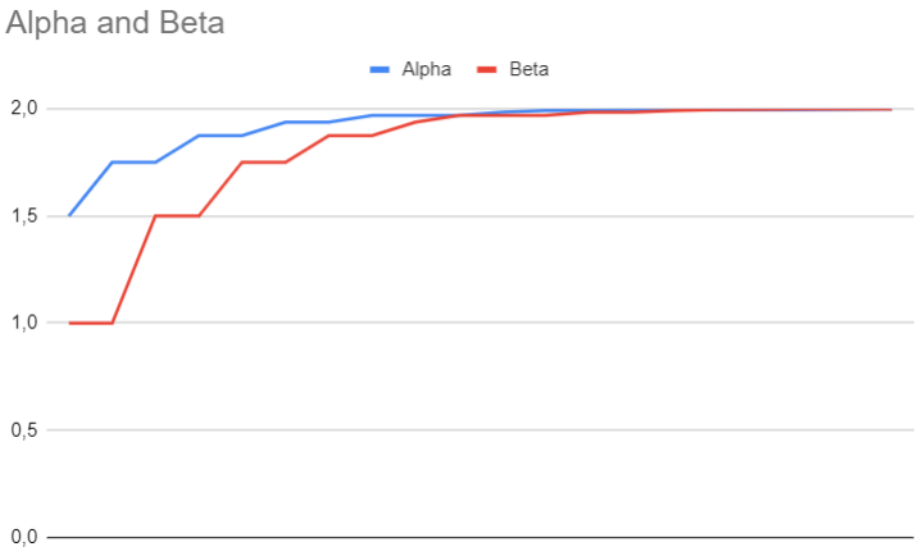
(b) Mean

Figure A.7: Graphics of the model's behaviour with four positive and three negative events with AGING_FACTOR=0.5

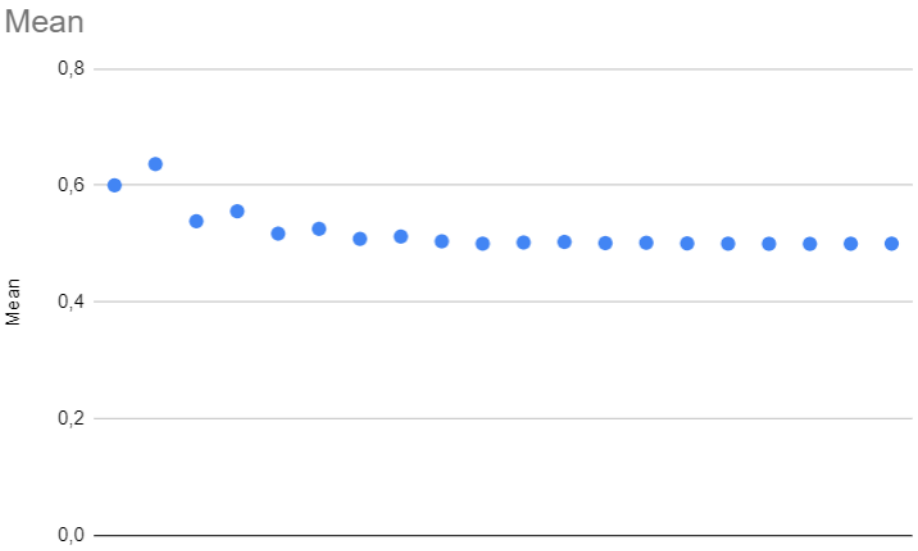
A.8 20 random events with an ageing factor of 0.5

Event	Alpha	Beta	Mean
positive	1.5	1.0	0.6
positive	1.75	1.0	0.6363636364
negative	1.75	1.5	0.5384615385
positive	1.875	1.5	0.5555555556
negative	1.875	1.75	0.5172413793
positive	1.9375	1.75	0.5254237288
negative	1.9375	1.875	0.5081967213
positive	1.96875	1.875	0.512195122
negative	1.96875	1.9375	0.504
negative	1.96875	1.96875	0.5
positive	1.984375	1.96875	0.5019762846
positive	1.9921875	1.96875	0.5029585799
negative	1.9921875	1.984375	0.5009823183
positive	1.99609375	1.984375	0.5014720314
negative	1.99609375	1.9921875	0.500489716
negative	1.99609375	1.99609375	0.5
negative	1.99609375	1.998046875	0.4997555012
negative	1.99609375	1.999023438	0.4996333415
positive	1.998046875	1.999023438	0.4998778402
positive	1.999023438	1.999023438	0.5

Table A.8: 20 random events with AGING_FACTOR=0.5



(a) Alpha and Beta



(b) Mean

Figure A.8: Graphics of the model's behaviour with 20 random events with AGING_FACTOR=0.5

Appendix B

List of Exchanges in the Reputation System

This appendix endeavours to comprehensively present all the information obtained from various exchanges and elucidate the message parsing process. Its primary purpose is to facilitate the reputation calculation and the storage of essential data for the data visualisation platform.

B.1 Network Flow Monitor Exchange


```
{
    "Resources": {
        "flowResourceId": "D4D7BC93",
        "parentResourceId": "E39B1DF2",
        "encapsulationLayer": 2,
        "encapsulationID1": "000029CC",
        "encapsulationID2": "00000001",
        "encapsulationType1": "vxlan",
        "encapsulationType2": "gtp",
        "sense": "INGRESS",
        "outMacSrc": "40:00:00:02:00:03",
        "outMacDst": "40:00:00:02:00:05",
        "srcIP": "10101100010100101101111011111110",
        "dstIP": "10101100010100101101111000000100",
        "outSrcIP": "00001010000000010000000000000110",
        "outDstIP": "00001010000000010000000000001010",
        "l4Proto": "1",
        "tos": "192",
        "resourceAbstractionLayer": "2",
        "resourceId": "A0B91002",
        "resourceType": "FLOW_SAMPLE",
        "state": "ACTIVE",
        "serviceInstanceResourceId": "16168DAF",
```

```
        "reportedTime": 1669221845510
    },
    "Alert": {
        "alertName": "7",
        "alertReasonId": "10000002",
        "alertAssertionType": "NEGATIVE",
        "alertImpact": 2,
        "alertTime": 1669221844127,
        "resourceId": "01CEB72C",
        "resourceType": "ALERT",
        "state": "FIRED",
        "serviceInstanceResourceId": "16168DAF",
        "reportedTime": 1669221845514
    }
}
```

Listing B.1: JSON object example of reputation information being sent from *nfm_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *flowResourceId*: In the reputation system it's the ID of the entity;
- *alertAssertionType*: Being either "INFORMATIVE" or "NEGATIVE", it will decide whether the event is positive or negative, respectively;
- *alertImpact*: Will function as the severity factor to calculate the reputation score using the Alpha-Beta model with a severity factor.

B.2 Device Behaviour Monitor Exchange

```
{
    "timestamp": "1677767901.81823",
    "attack_start_date": 1677767898.168318,
    "occurence_number": 11,
    "sender": "behaviour_monitoring",
    "device_id": "drone01",
    "cause": "Xorg",
    "process_id": 920
}
```

Listing B.2: JSON object example of reputation information being sent from *dbm_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *device_id*: In the reputation system it's the ID of the entity;
- *cause*: It is the type of action of the event.

It is noted that, as agreed with the project partners, the events received from the dbm_exchange will be solely negative events.

B.3 Remote Attestation Exchange

```
{
    "id": "attester",
    "appraisal_result": 1.0,
    "trust_score": 1.0
}
```

Listing B.3: JSON object example of reputation information being sent from *ra_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *id*: In the reputation system, it's the ID of the entity;
- *appraisal_result* & *trust_score*: These two parameters are essential to check if the event is positive or negative. If the *appraisal_result* is lower than 0.8 and the *trust_score* lower than 0.9, then the event is classified as negative, else, it is a positive one. These parameters will also be essential to know the type of action, whether it has "trustable claims" or "low trustable claims".

B.4 Network Authorisation Exchange

```
{
    "imsi": "204047795980920",
    "rule": "allow"
}
```

Listing B.4: JSON object example of reputation information being sent from *naz_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *imsi*: In the reputation system, it's the ID of the entity;
- *rule*: Being either "allow" or "deny", it will decide whether the event is positive or negative, respectively, and it will also be the type of action.

B.5 Biometrics Exchange

```
{
    "message": "User registered correctly",
    "code": 0
}
```

Listing B.5: JSON object example of reputation information being sent from *bio_exchange* exchange

To know the ID of the entity, which is encoded in base64 and the system is responsible for decoding it, and the action that was performed (register, update or delete), these parameters are sent via the message's routing key, which has the following structure:

a.bio.cloud.arcadian_iot_ID.crud.__**CRUD**__.reply.__**AIOT_ID**__

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *code*: With the different values [0..10], the system will evaluate if the event is positive or negative.

  If the CRUD type is **create**:

  - 0: User registered correctly
  - 1: Error processing image - Face not detected
  - 2: Error processing image - More than one face detected
  - 3: Error processing image - Other error
  - 4: ID already in the Database
  - 10: Other error

  If the CRUD type is **update**:

  - 0: User updated correctly
  - 1: Error processing image - Face not detected
  - 2: Error processing image - More than one face detected
  - 3: Error processing image - Other error
  - 4: Error updating user – No user with that ID
  - 5: User updated correctly - No changes made
  - 10: Other error

  If the CRUD type is **delete**:

  - 0: User deleted correctly
  - 1: Failed to delete user - No user with that ID

B.6 Self-Sovereign Identity Exchange

```
{
    "aiotID": "registeringEntity.net:b94a6585-3efd-4765",
    "type": "Service",
    "info": {
        "contactEmail": "test@mail.com",
        "did": "did:test",
        "domainURL": "https://test.com"
    }
}
```

Listing B.6: JSON object example of reputation information being sent from *ssi_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *aiotID*: In the reputation system it's the ID of the entity;
- *type*: The type of entity, being either person, service or device.

It is noted that, as agreed with the project partners, the events received from the *ssi_exchange* will be solely for initialisation purposes. When the reputation system is fully implemented, all system entities will be initialised in this exchange.

When introducing a new entity, the system sends a message to *ra_exchange* containing only the entity ID to make an attestation request.

B.7 Middleware Exchange

```
{
    "Message": {
        "ClientId": "l42nebnokl2",
        "Username": "l42nebnokl2",
        "TimestampUTC": "2023-06-14T19:55:04.5989538Z",
        "Reason": "invalid decryption key"
    },
    "ArcadianId": "api.box2m.io:b666ca65-0faa-4e8b-a4bb",
    "Type": "NotAuthorized"
}
```

Listing B.7: JSON object example of reputation information being sent from *middleware_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *ArcadianId*: In the reputation system it's the ID of the entity;
- *Type*: This type of message will define if the event is either positive or negative, also defining the type of the event.

B.8 Self-Aware Data Privacy Exchange

```
{
    "DeviceId": "Alice's phone",
    "User": "Alice",
    "Method": "POST",
    "Target URL": "/Alice/vitals",
    "HEOp": "encrypt with Alice's policy"
}
```

Listing B.8: JSON object example of reputation information being sent from *sadp_exchange* exchange

From this information sent to the reputation system of this exchange, the important factors for storage are the following:

- *DeviceId*: In the reputation system it's the ID of the entity;
- *HEOp*: It's the type of action of the received event.

It is noted that, as agreed with the project partners, the events received from the *sadp_exchange* will be solely positive events.

Appendix C

Data Visualisation Platform Full UI Presentation

This appendix aims to provide a comprehensive overview of the interface for the data visualisation platform, as described in Section 6.5. The interface is a vital platform component, enabling users to effectively explore and interpret data through visual representations. With a wide range of features, tools, and functionalities, the interface empowers users to interact with the platform's extensive data visualisations, offering valuable insights and facilitating data-driven decision-making.

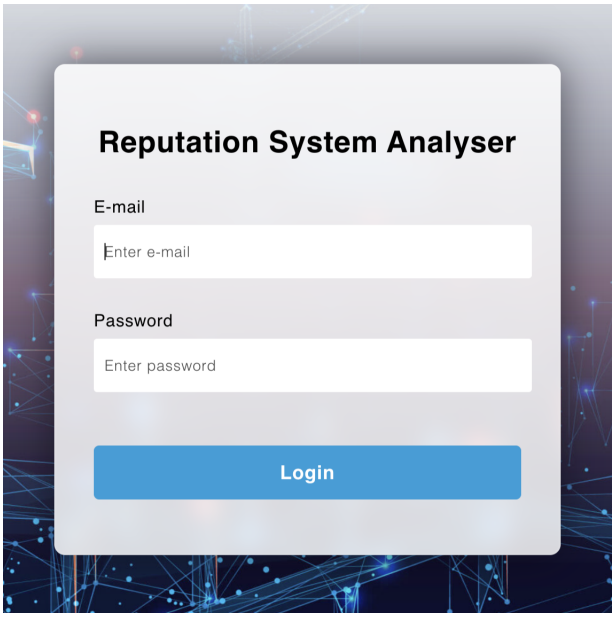


Figure C.1: Login page

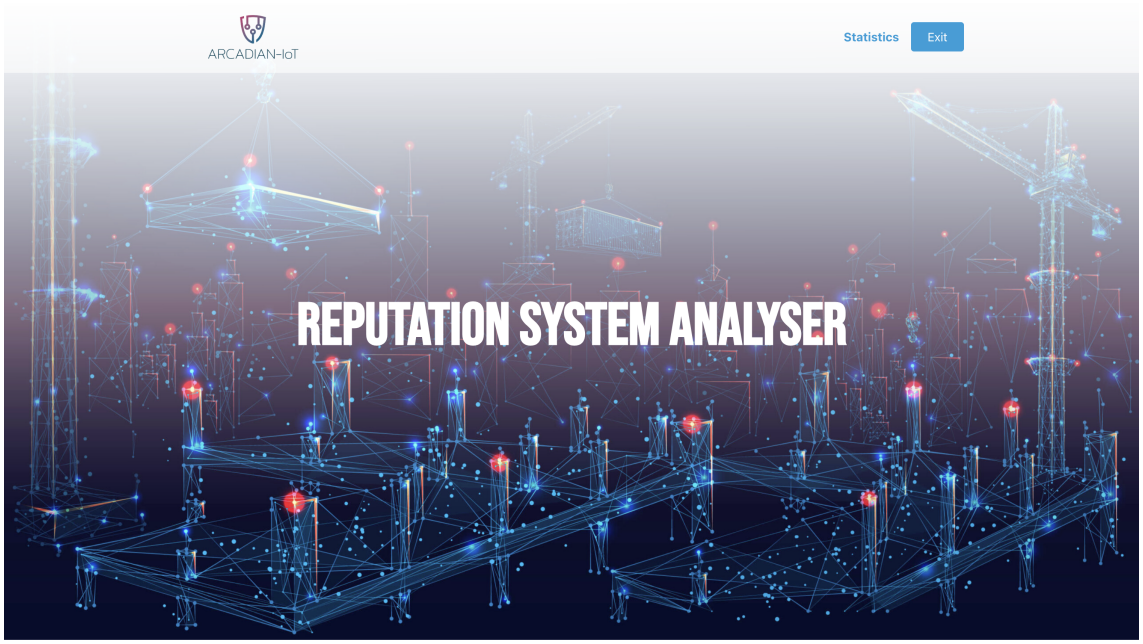


Figure C.2: Homepage welcome screen

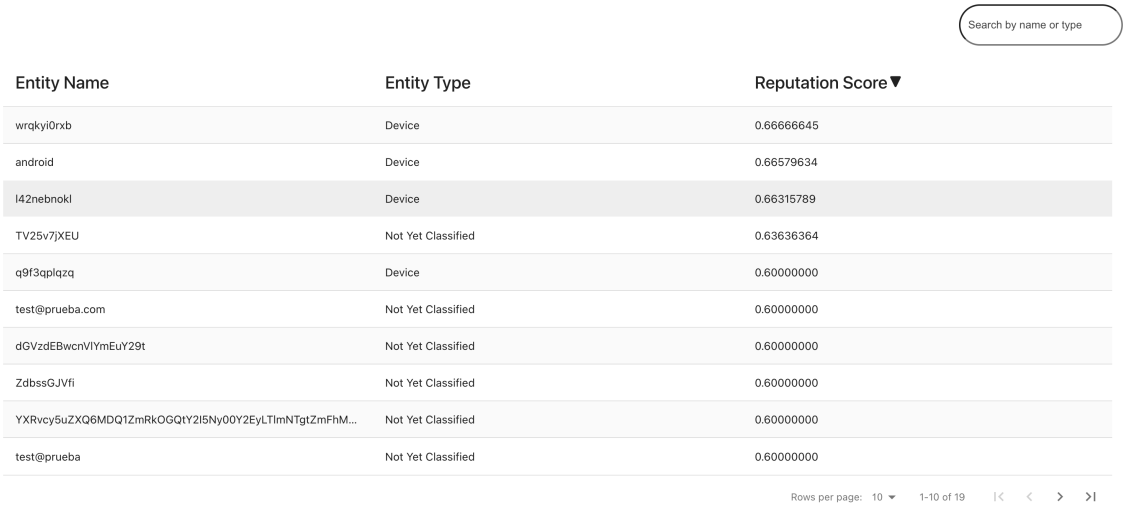


Figure C.3: Table with all the entities in the reputation system

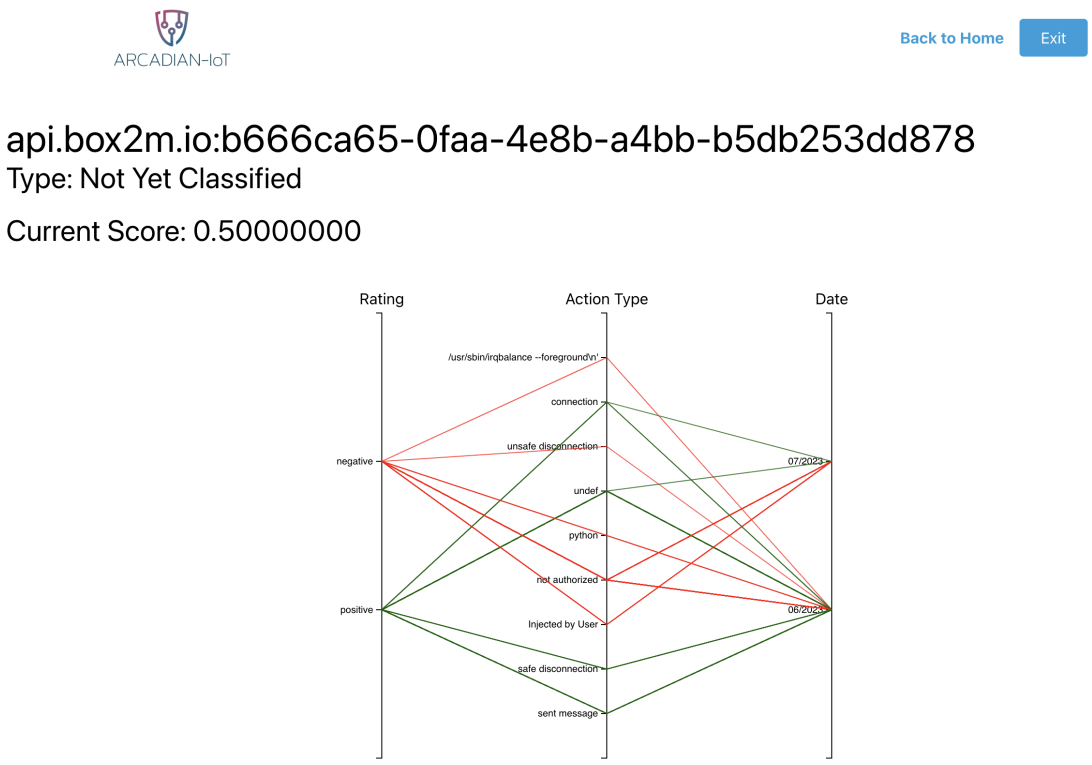


Figure C.4: Parallel coordinate chart with all of an entity's events

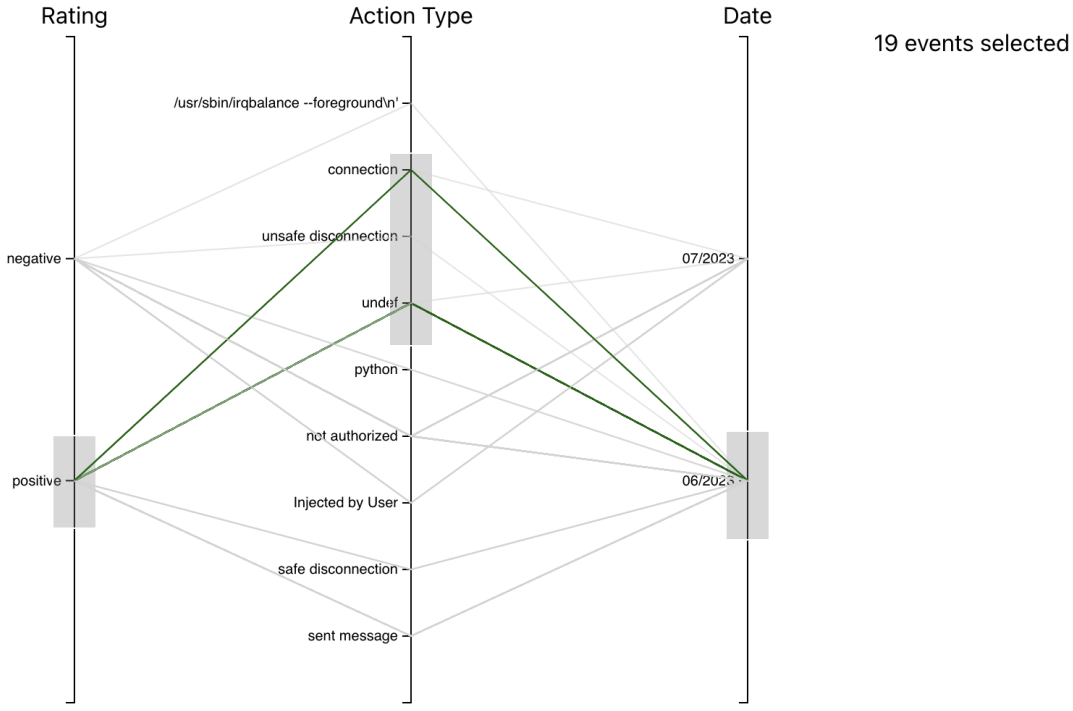


Figure C.5: Parallel coordinate chart with brushed events

All 84 Events Additional Information

Search date (yyyy/mm/dd HH:MM:SS)

Action Type	Model Used	Origin Exchange	Detailed Date
sent message	Alpha-Beta	middleware_exchange	2023/06/14 20:44:50
safe disconnection	Alpha-Beta	middleware_exchange	2023/06/14 20:36:35
Injected by User	Alpha-Beta	dbm_exchange	2023/07/12 15:49:55
not authorized	Alpha-Beta	middleware_exchange	2023/07/10 14:21:30
safe disconnection	Alpha-Beta	middleware_exchange	2023/06/14 20:37:40
not authorized	Alpha-Beta	middleware_exchange	2023/07/10 15:12:45
sent message	Alpha-Beta	middleware_exchange	2023/06/14 20:50:55
not authorized	Alpha-Beta	middleware_exchange	2023/06/14 13:34:50
python	Alpha-Beta	dbm_exchange	2023/06/26 20:01:20
not authorized	Alpha-Beta	middleware_exchange	2023/07/10 14:21:15

Rows per page: 10 1–10 of 84

Figure C.6: Additional information of an entity's events

Reputation Score History

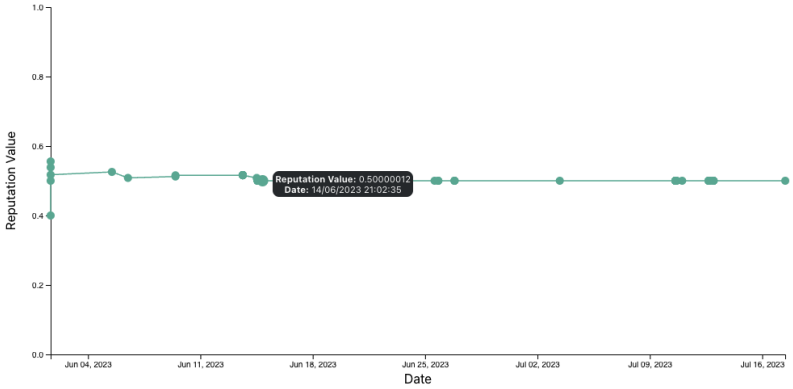


Figure C.7: Graphic with the reputation history of an entity

ARCADIAN-IoT

Back to Home Exit

Total Entities in System: 19

Total Processed Events: 108

Entity Type: Device
- Number of Device: 8
- Device Reputation Mean: 0.53456230
- Device Minimum Reputation: 0.36363636
- Device Maximum Reputation: 0.66666645

Entity Type: Not Yet Classified
- Number of Not Yet Classified: 9
- Not Yet Classified Reputation Mean: 0.54416541
- Not Yet Classified Minimum Reputation: 0.26086957
- Not Yet Classified Maximum Reputation: 0.63636364

Figure C.8: Page with the statistics of the reputation system