UiA University of Agder

Bachelor

Leveraging User Metadata for Bias Mitigation in AI Datasets

A GDPR-Compliant Approach

JON ANDREAS BULL LARSSEN
HÅVARD WILLOCH BERGSVIK
TOBIAS HOLMEN FRÆKALAND

SUPERVISOR
ARNE WIKLUND

University of Agder, 2024
Faculty of Engineering and Science
Department of Engineering and Sciences

Acknowledgements

We want to thank our supervisor, Arne Wiklund, for guiding and supporting us during this project. His knowledge and support played a important role in guiding our research and overcoming the challenges we had.

A thank you to Morten Goodwin for taking the time to an interview. His insights were helpful for our research.

We appreciate the university for giving us a good place to learn and do research. The things we learned during our studies were helpful for completing this project.

Abstract

As artificial intelligence (AI) continues to advance, the challenge of addressing biases within AI systems becomes more pressing. This thesis investigates the potential of developing a framework that leverages user metadata to reduce bias in AI datasets while maintaining compliance with GDPR regulations. An extensive literature review is conducted to understand the extent and sources of bias in AI, followed by a semi-automated system design approach and demonstration of key concepts with a simple prototype.

The literature review highlights that AI development is highly dependent on increased quantity, diversity, and quality of data. It also underscores that bias in AI is a significant issue originating from various sources, such as data annotation and dataset composition. The review discover a research gap regarding the role of user metadata and annotator diversity in causing these biases, and the potential for leveraging this user metadata to help mitigate bias.

The prototype demonstrate practical solutions for obtaining user consent, collecting data and facilitating data annotation in compliance with GDPR. It showcases the potential for users to continuously manage their data contributions and consents, ensuring it can be utilized ethically and legally. This proves that it is feasible to leverage user data while adhering to GDPR, provided proper measures are in place. Ensuring informed user consent, understanding implications, and allowing easy modification of consent without negative consequences are vital steps in this process. The prototype visualise a way developers and researchers can analyse and adjust the demographic representation of annotators within a set of training data in order to mitigate bias.

The findings suggest that developing a framework for incorporating user metadata can lay the groundwork for more equitable AI systems. Although this thesis does not evaluate alignment with the AI Act, the proposed platform serves as a foundational tool that supports its principles, such as ensuring high-quality, bias-free data. This thesis concludes with an approach that we believe to be a solid foundation for mitigating bias in AI, highlighting the importance of transparency, human oversight, and data governance.

Keywords— GDPR, AI Act, BIAS, AI, User Metadata, Bias Mitigation, AI datasets, Data Governance, High-Quality Data, Ethical AI Development

Contents

List of Figures

List of Tables

Acronyms

AI
Artificial Intelligence. ix, 1–8, 17, 19–21, 24–28, 39, 79, 81, 82, 87, 90, 91, 93, 94

AI Act
Artificial Intelligence Act. 10, 79–81, 94

API
Application Programming Interface. 17, 20–22, 39, 52, 55, 57, 59, 87, 88

CAGR
Compound Annual Growth Rate. 2

CSS
Cascading Style Sheets. 22

DSR
Design Science Research. 16, 24

ECB
Entity Control Boundary. 14

EEA
European Economic Area. 9, 92

GDPR
General Data Protection Regulation. iii, x, 1, 5, 9–11, 24–26, 29–32, 34, 36, 63, 69–71, 79, 80, 82, 83, 85–87, 92–94

HTTP
Hypertext Transfer Protocol. 17, 22

IDE
Integrated Development Environment. 26

JSON
JavaScript Object Notation. 22

NLP
Natural Language Processing. ix, 19, 38, 39

SQL
Structured Query Language. 23

UI
User Interface. 22, 52, 60, 67, 91

UML
Unified Modeling Language. 14, 20

WCAG
Web Content Accessibility Guidelines. 29

Chapter 1

Introduction

Imagine a world where an AI system determines your chances of getting a loan, landing a job, or even the outcome of a legal case. Consider the story of Alex, a person in their mid-fifties with an impressive educational background and extensive professional experience. Despite these qualifications, Alex repeatedly faces rejection for job positions by an AI-driven recruitment platform. Frustration grows with each rejection, not understanding why decades of experience and solid education are not enough to secure a job.

As Alex looks into the problem further, the reason for the rejections becomes clear: the bias embedded in the AI system. The AI, designed with biases from skewed training data and influenced by the implicit biases of its annotators, overlooks Alex in favor of younger candidates or those with different backgrounds. This example highlights a major issue. Bias in AI does not just harm specific groups. It can impact anyone, disrupting lives and careers indiscriminately.

Consider for a moment: What field or industry do you imagine Alex working in? Which gender do you imagine Alex to be? How many genders did you consider? These questions reveal our own biases and assumptions. Alex's story is a call for us to address and fix the biases in our AI systems, a challenge that is very relevant in our time, to ensure they serve all members of society fairly.

Bias in AI systems is not only a technical flaw, but also a critical issue with significant consequences. These biases can result in unfair and discriminatory outcomes, influencing important decisions that affect individuals and communities globally. This highlights the importance of creating fair and dependable AI solutions in our field.

Current research tells us that skewed training data is a major source of bias in AI. Acquiring diverse and balanced data is a complex challenge. Datasets can be unrepresentative, difficult to update, or become outdated, raising questions about how to effectively gather new data that accurately reflects diverse populations. While it is established that increasing the diversity of datasets can help mitigate this bias, it remains unclear how the demographics of the annotators contribute to this problem.

Furthermore, simply having a range of data is not enough. Understanding how the demographics of annotators, such as their cultural, social, and economic backgrounds, influence their labeling decisions is essential. If annotators hold biases in their viewpoints, these biases could unknowingly influence AI systems and perpetuate existing inequalities.

The existing literature lacks comprehensive studies on how user metadata, such as information about the annotators backgrounds, can be used to mitigate bias in AI systems. To address this gap, this thesis will explore how user metadata can be leveraged to create more balanced and equitable AI datasets. Furthermore, we will ensure that this approach complies with GDPR regulations, providing a framework for ethical and legally sound AI development

In addition, this thesis will reference relevant articles from the AI Act, including art.10 on data governance, art.13 on transparency, art.14 on human oversight, art.17 on quality management systems, and art.51 on transparency obligations. These articles will provide a regulatory framework to support the ethical development of AI systems.

1.1 Background

1.1.1 Emerging technology

Artificial intelligence and machine learning are substantially emerging technologies within the global market. With a size of 207.9 billion USD in 2023, Statista expects it to reach 1.8475 trillion USD by 2030 [1]. Accompanied by AIs total addressable market anticipated to surpass a 10x value growth from 2022 reaching between 12.14 and 15.72 trillion USD in 2030 [2]. "Business reasearch insights" [3] and "The brainy insights" [4] have both published forecasts for the development of the AI training data market. While their valuations differ significantly, both reports forecast the market to hold an average CAGR at about 24 % towards 2032 [3, 4].

1.1.2 Outline of steps in AI development

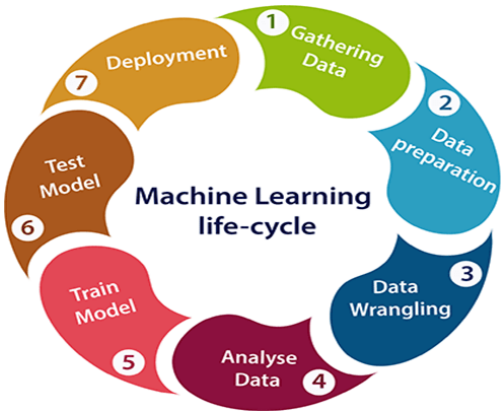

Figure 1.1: Machine Learning Lifecycle [5]

Gathering data

As AI models are also growing in size and complexity, the demand for training data to support this development is skyrocketing [6]. To develop robust AI models existing datasets can be used or new data can be gathered. Existing datasets can be sourced from various platforms. Several platforms have become essential resources for data gathering, machine learning, and other related activities. Table 1.1 shows some popular services for these purposes.

Table 1.1: Data platforms and services

Service	Datasets	Key Feature
Google Dataset Search	45M from 13k websites [7]	Dataset discovery
Kaggle	332k	Competitions
Papers with code	128k	State-of-the-art results
Huggingface	100k+	Model hub
OpenML	5k	Machine learning tasks
Microsoft	155	Research tools
Other sources [8]	Various	Curated lists

If existing datasets are insufficient or do not meet the required standards, methods like web scraping, sensor data collection, social media mining, and public records access can be employed to gather new data. The process involves several steps:

- Identifying the specific needs and requirements of the AI model.
- Designing a data collection strategy that ensures diversity and representativeness.
- Gather data with one or several techniques
- Ensure that the gathered data is properly annotated to maintain consistency and quality.

Data preparation, wrangling and analysis

Once data has been collected, the next step is to prepare and analyze it. This process involves cleaning, structuring, and enriching the raw data to make it suitable for analysis and machine learning model training.

In order to develop large training data sets, crowdsourcing is an essential strategy [9] and there are several crowdsourcing platforms being used for this. Some of them are shown in fig 1.2. But even with the presence of these solutions, finding the most suitable datasets to specific cases can be hard. The availability of data might be not sufficient, which can lead developers, researchers and practitioners to adapt their questions according to what data they have available rather than what data is most suitable [10, 11, 12].

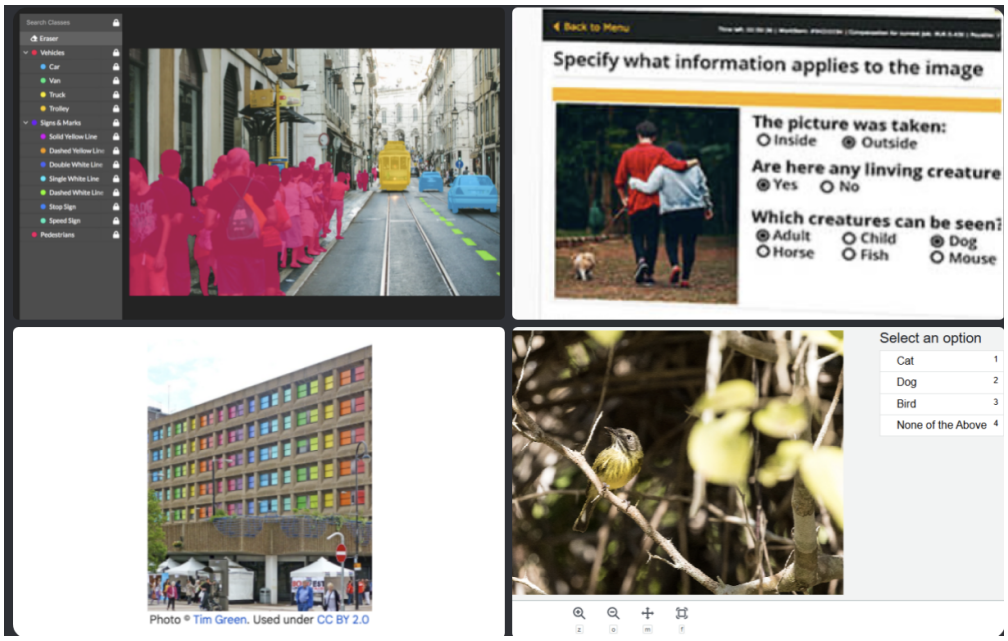

Figure 1.2: Dashboards of platforms from Google Crowdsource, Clickworker, Appen and Amazon mTurk

BIAS

A problem regarding the quality of training data, is human biases being reflected in the data. An AI model trained on data containing biases will adopt the bias and act accordingly. There is no lack of examples for cases where bias has been adopted by AI technology, and the harm it may cause to people in the real world [13, 10]. These biases can stem from training data bias [14, 13]. According to a statement by M. Goodwin, contemporary researchers lack a robust methodology for identifying the root of biases in pre-existing biased AI models. This insight was obtained during an expert interview on January 31, 2024 with M. Goodwin [15]. This insight is backed by [16]. Actions to mitigate this bias should start early in the development phase. As early as ensuring a diverse pool of human annotators, to gain diverse viewpoints, ensuring a diverse representation of population groups within datasets containing human-related content and continuously controlling technologies in production including updating systems as new data becomes available are good practice to limit bias in AI [17, 18].

Regulations

In 2020, the European commission highlighted the need for EU to act towards improving the access of high quality public data for reuse [11]. Additionally, the European Union's Artificial Intelligence Act (AI Act) [19] provides a regulatory framework to ensure the ethical development and deployment of AI systems, addressing issues such as data governance and bias mitigation.

1.2 Project objective

This project aims to explore the possibilities to leverage user metadata to reduce bias in AI datasets while ensuring compliance with GDPR. The obejctive of this bachelor thesis includes conducting an extensive literature review, developing a prototype, and discussing the results and findings.

The first goal is to do an extensive literature review of existing literature to evaluate how common bias is in AI and where it comes from. This review will lay the foundation for understanding bias in AI and how user metadata can be leveraged to address these biases.

The second goal is to develop a prototype that shows the practical application of integrating user metadata into AI datasets to reduce biases, while complying with GDPR. The prototype will include different features that follows the GDPR requirements. This will demonstrate how privacy and data protection can be balanced with efforts to reduce AI biases while complying with GDPR.

The third goal is to discuss the findings and results from both the literature review and the prototype development. This involves discussing how user data can be leveraged to reduce bias in AI, based on the results from literature review, and mentioning future implementation of the prototype.

Our objectives explicitly excludes:

- Full-scale implementation of the system.
- Full GDPR compliance in the prototype.

1.2.1 Research question

> *How can user metadata be leveraged to reduce bias in AI datasets, while ensuring compliance with GDPR?*

1.3 Project organization

The project was divided into four main areas: research, system design, development and documentation (report). We assigned roles based on individual strenghts and weaknesses, but we all helped in the areas it was needed. Our roles were assigned as follow.

Håvard Willoch Bergsvik: Research, Documentation
Tobias Holmen Frækaland: Research, Documentation
Jon Andreas Bull Larssen: Research, System design, Development, Documentation

1.4 Report Outline

This section provides an overview of the report structure and a brief description of the content for each part of the report.

1.4.1 Introduction

An introduction to the research topic, outlining the background, project organization, report outline and objectives of the thesis.

1.4.2 Theoretical Background

Discusses the theoretical framework and concepts related to AI, annotation, and bias in datasets.

1.4.3 Technical Background

Explores the technical environment and tools used in the project.

1.4.4 Methodology

Describes the research design and methods used, including the exploratory and design science research approaches.

1.4.5 Results

Presents the findings from the results.

1.4.6 Discussion

Discussing the results of the project, including further work.

1.4.7 Conclusion

Concluding the projects findings and reflecting on the research questions.

Chapter 2

Theory

2.1 AI

Artificial intelligence is the ability of a digital computer to perform tasks commonly associated with intelligent beings [18]. The development of AI algorithms are supported by the concepts machine learning and deep learning. Inspired by the human brains capability to learn from available data and make increasingly more accurate classifications over time. Machine- and deep learning algorithms use programmatic structures, consisting of interconnected layers of nodes. These nodes extract features from the data and make predictions of what it represents [20].

There are two types of AI, "weak" or "narrow" AI and strong AI [20]. Weak AI is trained to execute specific tasks. To this day, weak AI is the type that is applied in most of the AI implications surronding the world today, such as apples Siri, self driving vehicles and streaming suggestions on Netflix [20, 21, 18]. Strong AI is a theoretical type of AI, where developers aim to develop an AI that would have intelligence equal to humans, with self awareness and consciousness making it able to solve problems, learn and plan for the future.

2.2 Annotation

Data annotation is the process of attributing, tagging or labeling data to help machine learning algorithms understand and classify the information they process [22]. This data that is annotated can be any type of data that is relevant for an AI, such as labeling images, categorizing text or identifying relevant features in data sets. As machine learning and AI algorithms are developed to "learn" from data in order to improve their performance [20, 22], data annotation is a crucial part of the development of AI models.

The most basic technique of image annotation is classification. With image classification, the annotation seeks to determine the presence of a type of object (class) in an image. If an instance of the class is pictured, the image is "tagged" or "labeled" with that class [23, 24, 25]. There are also several other more detailed image annotations techniques [25] like object detection and image segmentation, which focus on marking the different contents of an image. Image annotation jobs done to a dataset set the standard the model will try to copy, this counts for any errors in the annotation too [25], hence annotation is a critical aspect with significant impact on models performance.

2.3 Crowdsourcing

When an individual or an organization requests specific resources from a body of people to obtain needed knowledge, goods or services its called crowdsourcing [26]. This concept can be an effective method to leverage resources on a large scale with low costs. An untraditional aspect is that the employer does not choose who will work on the tasks he solve with crowdsourcing. In order for the crowd and employer to communicate and exchange work for compensation a crowdsourcing platform is needed where the employer list tasks and crowd workers can submit completed tasks and get compensation through a reward system [27]. AI models are growing in size and the demand for training data is increasing [6]. Thus crowdsourcing has become a common solution to annotate datasets [28]. The method of crowdsourcing has become an essential aspect for the creation of large datasets [9]

2.4 BIAS

BIAS is the action of supporting or opposing a particular person or thing in an unfair way, because of allowing personal opinions to influence your judgement [29]. Bias can be adopted by AI models. This leads to AI systems that produce biased results that reflect and perpetuate human biases [13]. In their study, R. J. Thomas and T. J. Thomson analyzed over 100 AI-generated images, identifying trends of bias in the selection, they found different bias trends. For example images returned from terms like "journalist" or "Reporter" only featured light-skinned people with quite conservative appearance [30]. The hiring algorithm used at Amazon was suspended when they realized it favored applicants based on specific phrasings commonly used in men´s resumes [13]. AI bias can also occur as algorithmic and cognitive bias. Algorithmic bias can be a consequence of developers unfairly weighting factors in decision-making algorithms. Cognitive bias stems from inevitable influence by our experiences and preferences. Cognitive bias could be the root to training data- and algorithmic bias, as it for example leads to Americans favoring datasets gathered from Americans rather than sampling from a range of demographics in the world [13].

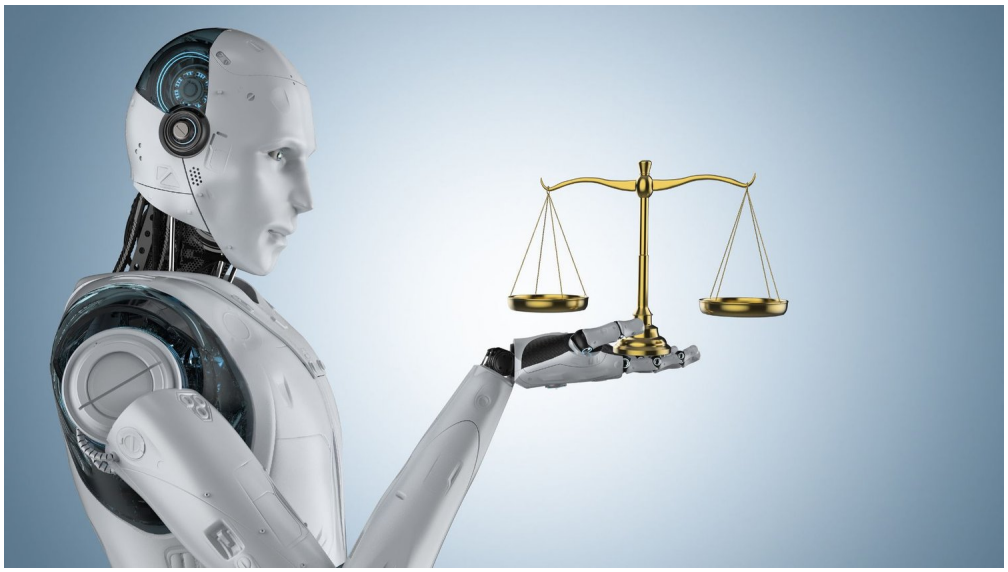

Figure 2.1: Bias in AI retrieved from [31]

2.5 General Data Protection Regulation

GDPR is the effective regulation of data privacy laws across Europe [32]. This regulation puts quite heavy boundaries on what personal data businesses are allowed to process, including with software.
Processing data is defined as "Any operation which is performed on personal data" [32]. To be allowed to process personal data one must fulfill one of six conditions[34]:

- User consent to processing of data
- Processing is necessary for performance of a contract the user is or will be a part of
- Necessary for compliance with a legal obligation
- Necessary in order to protect the vital interests of any natural person
- Necessary for performing a task in the public interest or in the exercise of official authority
- Necessary for legitimate interest pursued by the controller or by a third party, when they don't restrain fundamental rights and freedoms

Figure 2.2: GDPR retrieved from [33]

2.5.1 Geographical scope and impact

The main reasoning for the implication of GDPR is to ensure individuals right to a private life, and ensure effective interactions across businesses, organisations and states in an increasingly digitized society [35]. The regulations purpose is to protect data belonging to EU citizens and residents including residents and citizens in the European Economic Area "EEA". GDPR applies to organisations that handle this data, regardless of where the organisation is based [36, 37].

2.5.2 Rights

Amongst the main rights of identifiable natural persons, labeled as "data subjects" covered by GDPR is the right of access. This ensures the data subjects right to obtain confoirmation to whether or not personal data concerning them is being processed and their access to this personal data. The data subjects are also ensured the right to be forgotten, thus the right to have their personal data removed from any organisation. The data subjects have the right to data portability ensured with GDPR. This means that organisations shall provide the personal data they have concerning anyone that requests it, to them in a structured, readable and machine readable format. Theese rights have some exceptions, such as when processing is necessary for compliance with a legal obligation [38].

2.5.3 Controller

'Controller' means the natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of personal data [32].

2.5.4 Processor

'Processor' means a natural or legal person, public authority, agency or other body which processes personal data on behalf of the controller [32]. The controller and processor shall designate a data protection officer in cases where processing is done by a public authority, involves regular and large-scale monitoring of individuals, or includes processing large amounts of sensitive data or data related to criminal convictions.

2.5.5 Data processing agreement (DPA)

GDPR compliance requires data controllers to sign a data processing agreement with any parties that act as data processors on their behalf [39]. Stated in [40], the contract shall stipulate that the processor,

- processes the personal data only on documented instructions from the controller.
- ensures that persons authorised to process the personal data have committed themselves to confidentiality or are under an appropriate statutory obligation of confidentiality
- takes all measures required pursuant to GDPR article 32. Article 32 regards the security of processing.
- respects the conditions referred to in paragraphs 2 and 4 of GDPR article 28 for engaging another processor.
- Assist the controller with reaching fulfilment of the controllers obligations to exercise the data subjects rights.
- Assist the controller in ensuring compliance with the obligations pursuant to GDPR articles 32 to 36.
- At the controllers request, deletes or returns all personal data to the controller after the end of processing services.
- Provide the controller information necessary to demonstrate compliance with GDPR related obligations.

2.6 Artificial Intelligence Act

The AI Act Regulation [41] is a legal framework established by the European Union to regulate the development, deployment, and use of AI systems within the EU. At the time of writing, the final text is not officially published, but it is expected to be published in the Official Journal of the European Union in early May 2024.
The most relevant topics to this paper are:

- **Data and Data Governance (Art.10):** This article mandates that high-risk AI systems be developed on the basis of quality training, validation, and testing data sets. Paragraphs 10.2.b, 10.2.c, and 10.2.f emphasize the importance of data collection processes, preparation operations, and bias detection measures, respectively.
- **Transparency and Provision of Information to Users (Art.13):** This article states that high-risk AI systems must be designed and developed in such a way as to ensure that their operation is sufficiently transparent. This transparency enables deployers to interpret the system's output and use it appropriately.
- **Human Oversight (Art.14):** High-risk AI systems must be designed to allow effective human oversight. Paragraph 14.1 emphasizes the need for appropriate human-machine interface tools, while paragraph 14.2 outlines measures to prevent or minimize risks to health, safety, or fundamental rights.

2.7 Quality Attributes

Quality attributes, is a type of non functional requirement that describes the performance characteristics or service of a product. The quality attributes of a system makes it easier to use, leading to the user accomplishing tasks more efficiently. Quality attributes can also be stated as "statements that describe how well the system does something" [42].

2.7.1 Performance

Performance is a quality attribute of a system, indicating how responsive it is to different user requests and actions. This attribute includes many different trade-offs, including response time, throughput, data capacity, dynamic capacity, the predictability of real-time systems, latency, and the systems behavior under degraded or overloaded conditions [42].

2.7.2 Scalability

Scalability ensures that the application can handle more users, data, servers, places, transactions, network traffic, searches, and other services without losing speed or accuracy [42].

2.7.3 Reliability

Reliability address several attributes of a system. I.e. the percentage of operations that are completed correctly, the average length of time the system runs without failing and maximum acceptable probability of a failure during a given time scope [42].

2.7.4 Availability

Availability is how often the system is up and running as expected. In more specific terms, availability is calculated by taking the average time between failures and dividing it by the total of that time and the average time it takes to fix the system after a failure [42].

2.7.5 Usability

Usability is about making software easy to use. Usability looks at how much effort it takes to set up, use, and understand the outputs of a system [42].

2.7.6 Security

Security involves protecting the software from malicious interactions. Security is especially important for software that uses the internet. There are several considerations when evaluating security attributes, such as user authorization, user identification and authentication, data privacy, firewall, encryption of secure data, protection against viruses, corruption, theft and deliberate data destruction [42]. Security also regards protection of the physical areas such as the servers a system runs on. Protection against fire or strict rules on who are allowed to enter the room where the servers are located are examples of physical security [43].

2.7.7 Compliance

The system needs to comply with relevant laws and regulations, like GDPR.

2.7.8 Maintainability

Maintainability is about how easy it is to update or fix the system when needed [42].

2.7.9 Interoperability

Interoperability is about how well the system can share data and work with other software and hardware. To check interoperability, you need to understand what other applications users will use with your product and what data they plan to share [42].

2.7.10 Portability

Portability concerns the amount of effort required to migrate software from one operating environment to another. This attribute has gained relevance as applications must run on several environments, i.e. ios and android. The term can also be used to describe a system or products capability to be internationalized [42].

2.7.11 Data integrity

Data integrity regards the data´s accuracy and proper formatting. This includes making sure data fields are right, limiting fields to the correct type or size, ensuring data values are valid, and checking that entries in one field are appropriate based on values in another field [42].

2.7.12 Privacy

Data privacy is about ensuring that information is only accessible to those who are authorized to see it [42].

2.8 Requirements

Requirements are descriptions of what a system needs to do and the rules it needs to follow. They can be simple, like saying what the system should do, or detailed, like giving exact instructions on how it should work. These requirements gives details to potential stakeholders so they understand and can validate what the software can and will do [44].

2.8.1 Business requirements

When developing requirement in a project, business requirements are the first ones to be created. A business requirement represents the goal of either the organization constructing a product or the customer obtaining it. Basically, explain why the organization is putting the system in place and what benefits they hope to gain. It is all about the business goals of either the organization or the customer asking for the system [42].

2.8.2 User requirements

User requirements is a target or activity that a group of users should be able to do using a system, or a characteristic that the product should have. It describes different tasks that the user should be able to do within a system. There is several ways to represent these requirements, like making use cases or user stories. Ideally user requirements should come from actual users that will use the system, often called end-users [42]

2.8.3 System requirements

System requirements is a top-level requirement for a system or a product that has several subsystems. The system or product can be software and hardware or all software. The architecture of a system is very important, this includes allocating these requirements to the different subsystems and components. This is why these requirements is part of driving the architectural design. Modifying a high level requirement like the system requirements can affect multiple software and hardware requirements in numerous ways, because system requirements are "decomposed" into software and hardware requirements [42].

2.8.4 Functional requirements

Functional requirements is a description of how a system will act under certain circumstances. The functional requirements tells what developers must implement to achieve the user requirements. A certain functional requirement must be implemented to let the user be able to perform that certain task described by the user requirement.

In most cases there is almost always impossible to implement all functional requirements before moving on to the implementation and design. In these cases more functional requirements will be developed after receiving customer feedback.

2.8.5 Non functional requirements

A non-functional requirement specifies a quality or feature that a system must have or a limitation it must follow. Quality attributes are examples of non functional requirements. These describes a systems characteristics such as performance, availability, safety and portability. Other non functional requirements can show the external connections between the system and its environment. This can be connection to other hardware components, software connections and communication interfaces. By looking at each element of a use case to find relevant non functional requirements in a system [42].
Each use case description includes a sequence of actor actions and system responses, which you can model by using a dialog map to depict a possible user interface architecture. When users evaluate the prototype, their feedback might lead to changes in the use case descriptions or to changes in the dialog map.

2.8.6 MoSCoW

The MoSCoW method is a prioritization technique to rank the importance of requirements. The method provides four categories where each requirement is assigned one of these.

- "Must have": The requirement is essential for the solution, and must be satisfied for the solution to be considered a success.
- "Should have": Important, but not a vital requirement. It should be included if possible as its implementation is likely to significantly enhance the solution.
- "Could have": The requirement is considered desirable, but not necessary. Implementation should only happen if time and resources allow it
- "Won't have": Lowest priority. This requirement will either never be implemented, or not be implemented at this time, without excluding its future implementation.

Essentially MoSCoW tells developers which functionalities are vital and which ones they can look away from, with an additional guidance on which ones should be implemented after all the vital ones are covered. MoSCoW do not offer any guidance for how to rate the decision making, so this is left to the designers and developers[42] [45].

Figure 2.3: Moscow prioritizing retrieved from [46]

2.9 Use case descriptions

Every use case description contains a series of actions performed by actors and the corresponding responses from the system. These interactions can be illustrated using a dialog map to visualize a potential user interface layout. Changes in the use case descriptions can occur when users evaluate a prototype or a system depending on their feedback[42].

2.10 Diagrams

2.10.1 Use case diagram

Use case diagrams is an analysis model that shows a high-level view of user requirements. The system boundary is represented by a box frame. Actors are connected to the use cases they interact with via arrows. These arrows indicates the connection between the actors and use cases and does not represent a flow in any way. If an actor initiates a use case and receives the main value from it, they are considered the primary actor. This is indicated by the arrow that goes from the actor to the use case. If an actor helps with a use case but doesn't start it, they are considered a secondary actor and are represented by the arrow from the use case to the actor. Secondary actors, which may include other software systems, assist in the background during use case execution. I.e the relation between the use cases in the diagram can be extend and include [42].

2.10.2 Sequence diagram

Interaction diagrams shows how groups of objects work together to perform certain actions. In UML different types of interaction diagrams exist, with the sequence diagram being the most common one. A sequence diagram shows the steps of a particular scenario. It illustrates example objects and the communication between them during the process. The sequence diagram cleary shows the difference in how the participants interacts. This shows which participant doing which processing. One of the downsides using sequence diagrams is that it is not good at showing looping and conditional behaviours. This is where activity diagrams come in. Sequence diagrams are most commonly used when someone wants to look at the behaviour of many different objects within a use case[47].

2.10.3 Component diagram

Component diagrams splits a system into different components and show how these components connect or interact with each other. Component diagrams can also show the smaller parts inside these components[47].

2.11 Entity Control Boundary

The entity control boundary (ECB) is a pattern in software design that helps in structuring and organizing software applications. It involves categorizing software components into three types: entities, controls, and boundaries. Entities are the parts of the software that represent data and the rules related to that data. Controls handle the business logic or operations of the application. Boundaries deal with interactions between the software and users or other systems.

2.12 Model View Controller

MVC, or Model View Controller, is a design pattern in software development that organizes a program into three interconnected parts. This helps manage the complexity by separating the core logic (model), the user interface (view), and the user input (controller). This separation allows for changing one part without impacting the others[48].

2.13 Service Oriented Architecture

The service oriented architecture (SOA) pattern involves a set of distributed components, where each either provides or uses services. In SOA, the components providing services and those consuming them can be built using different programming languages[48].

2.14 Middleware pattern

Middleware is software that helps with common programming tasks by providing ready-to-use, standard solutions. These tasks include saving data permanently, preparing data for sending and receiving, managing message queues, sorting requests, and handling multiple tasks at once[49].

2.15 Repository patterns

The Repository pattern is part of Domain Driven Design and helps keep database details separate from the systems main model. It uses interfaces in the main model to define tasks, and these tasks are carried out by specific adapters set up elsewhere in the application[50].

2.16 Configuration Management

Configuration Management involves keeping systems, like computer hardware and software, in a specific, desired condition. It is a way to make sure that these systems work reliably and meet expectations consistently over time[51].

2.17 Research Methodologies

2.17.1 Exploratory

An exploratory research method is used to identify, understand, analyse, compare, evaluate, and interpret a problem and its possible solutions to give suggestions [52].
It involves the following steps.

- Activity 1: Identify a problem, issue, or system for exploratory research
- Activity 2: Develop objectives and methods for exploratory research
- Activity 3: Collect and analyze information
- Activity 4: Suggest solutions or improvements as new interpretations

2.17.2 Design Science Research

Design science research focuses on finding solutions. It results in various elements like new constructs, models, methods and new design theories, all adapted to solve specific problems [53]. DSR complements the exploratory aspect by focusing on the creation and evaluation of artifacts designed to solve identified problems [54]. It involves the following steps.

- Activity 1: Problem identification and motivation
- Activity 2: Define the objectives for a solution
- Activity 3: Design and prototype
- Activity 4: Demonstration
- Activity 5: Evaluation
- Activity 6: Communication

Chapter 3

Technical background

3.1 Python

Python is a programming language known for being simple and flexible. It is easy to use and works well for both beginners and experienced developers. Python allows for quick development of applications and smooth integration of different parts. Its clear and easy-to-understand syntax helps reduce the amount of maintenance needed. Furthermore, Python supports the use of modules and packages, making it easier to organize and reuse code [55].

3.2 Google Colab

Google Colaboratory, known as "Colab," is a cloud-based platform on Jupyter notebooks. It can be accessed through web browsers and serves as a place for working on machine learning and AI programming tasks. People can write and execute Python code, use preinstalled libraries, import external datasets, integrate with github, analyze and visualize data, share and edit simultaneously with team members, and produce detailed notebooks with text, graphs, pictures, HTML, and LaTeX explanations [56].

3.2.1 Google APIs

googleapiclient.discovery

The `googleapiclient.discovery` module is a component of the google-api-python-client library. It is utilized for accessing various Google APIs and enables dynamic discovery and interaction with them [57].

google.auth

The `google.auth` library in Python is used to authenticate with Google's APIs. It offers different authentication methods for interacting with Google services and can be used with different HTTP libraries. It also supports Google Application Default Credentials [58].

3.2.2 Google Sheets

gspread

Gspread is a Python tool that helps you work with Google Sheets. It works with Google Sheets API v4 and lets you open spreadsheets using the title, key, or URL. Gspread can be used to handle tasks like reading, writing, and formatting cell ranges and making batch updates quickly [59].

gspread_dataframe

The `gpsread_dataframe` library gives data exchange between a worksheet in a Google spreadsheet and a Pandas DataFrame. It makes the process of importing data from a Google Sheet into a DataFrame easier. The library also allows for export of data from a DataFrame back into a Google Sheet, making it a good tool for users who need to collaborate or present data using accessible spreadsheet formats [60].

gspread_formatting

The `gspread_formatting` library offers cell formatting capabilities for Google spreadsheets through the gspread package, and includes additional functionalities like the ability to set grids on rows and columns and other styling within a worksheet. It supports both straightforward and conditional formatting tasks. Additionally, the library enables formatting of Google spreadsheets using a Pandas DataFrame, improving the visual presentation and usability of spreadsheet data [61].

3.2.3 Data Handling

pandas

Pandas is a fast, reliable, and easy-to-use open-source tool created for analyzing and manipulating data, built on the Python programming language platform [62].

NumPy

NumPy is a key library for scientific computing in Python. It provides a multidimensional array object and other related tools such as masked arrays and matrices. The library includes numerous functions for quick operations on arrays, covering areas like math, logic, reshaping, sorting, and data selection [63].

3.2.4 Network Analysis

networkx

NetworkX is a Python library designed for building, modifying, and analyzing the structure, behavior, and functions of complex networks [64].

Features for handling complex networks include:

- Data structures for graphs, directed graphs, and multigraphs
- Different types of standard graph algorithms
- Tools for measuring and analyzing network structure
- Functions to generate typical graphs, random graphs, and artificial networks
- The ability to use any object as nodes (like text, images, XML records)

3.2.5 Visualizations

matplotlib

Matplotlib [65] is a Python library designed for creating static, animated, and interactive visualizations. It can make publication-quality plots, it supports interactive figures with features like zooming and panning, and offers extensive customization options for visual style and layout. Matplotlib also lets you export visuals to different file formats, works well with JupyterLab and graphical user interfaces.

plotly

Plotly is a browser-based, open-source visualization library that supports interactive graphics. It provides Python tools for charting, built on the plotly.js. The library offers different chart types, including 3D graphs, scientific charts, statistical diagrams, SVG maps, and financial charts and much more [66].

3.3 Natural Language Processing (NLP)

Natural language processing (NLP) is a branch of AI that merges computational linguistics, which involves rule-based analysis of human language, with statistical and machine learning models. The merging allows computers and digital devices to recognize, understand, and produce text and speech [67].

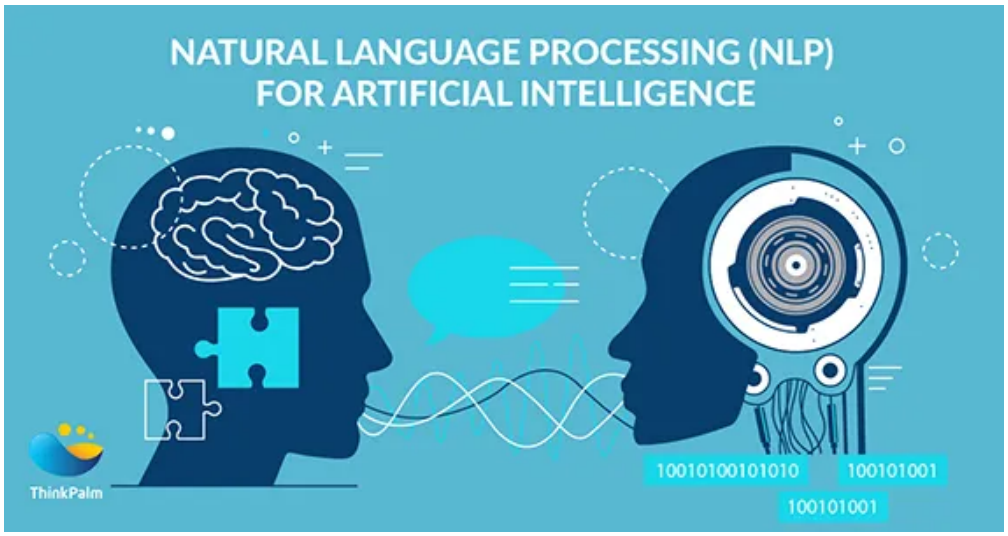

Figure 3.1: Natural Language Processing retrieved from [68]

3.3.1 spaCy

spaCy is a Python library that provides tools for quickly processing large amounts of text using Natural Language Processing (NLP). It is useful when users wants to create models and applications that analyze documents, create chatbots, and perform other text-related tasks [69].

`en_core_web_trf`

en_core_web_trf is a spaCy language model that uses Transformer architecture to give Natural Language Processing (NLP) capabilities. It gives users the availability to use entity recognition, part-of-speech tagging, and dependency parsing. A downside is that it requires more computational resources compared to simpler models [70].

`en_core_web_sm`

en_core_web_sm [71] is an efficient English language model provided by spaCy. It is designed for Natural Language Processing (NLP) tasks such as tokenization, part-of-speech tagging, and dependency parsing. It is faster and requires less computational resources than `en_core_web_trf`.

3.3.2 transformers

Transformers is a toolkit created by Hugging Face for deep learning. It provides access to advanced pre-trained models that can improve performance. These models are designed to handle different tasks like language processing, image recognition, audio analysis, and data integration [72].

`codebert`

Codebert is a dual-mode pre-trained model created for both programming language and natural language. The pretrained model is created to gather easy adaptable representations to assist various NL-PL applications, like natural language code search and code documentation generation. It uses the Transformer-based neural structure and is trained with a hybrid objective function that includes replaced token detection task to identify suitable alternatives from generators [73].

`graphcodebert`

Graphcodebert is a model for programming languages. It uses graphs and Transformer architecture to understand code better. Unlike other models, it also looks at how data moves through the code, which helps it understand code more [74].

`t5-large`

T5-large, also known as Text-to-Text Transfer Transformer is a set of big and advanced language models developed by Google AI. The models are trained on a large dataset of text and code [75].

3.3.3 OpenAI API

OpenAI is an API for accessing the latest AI models. The API offers a "text in, text out" interface, adaptable to different English language tasks. Users can integrate it into products and create new applications. With a simple text prompt, the API generates responses, learning from provided examples or human feedback to improve performance [76].

Compare models

The openAI API offers a feature called "compare". This features allows users to compare answers from different types of AI models. This is created to evaluate the models performances on different tasks. The user will be able to see what AI model that suits their needs the best by the response from the AI models [77].

3.4 UML

UML, or Unified Modeling Language, is used to create design models. It has different models that are useful. These can be class diagrams, use case diagrams and activity diagrams[42].

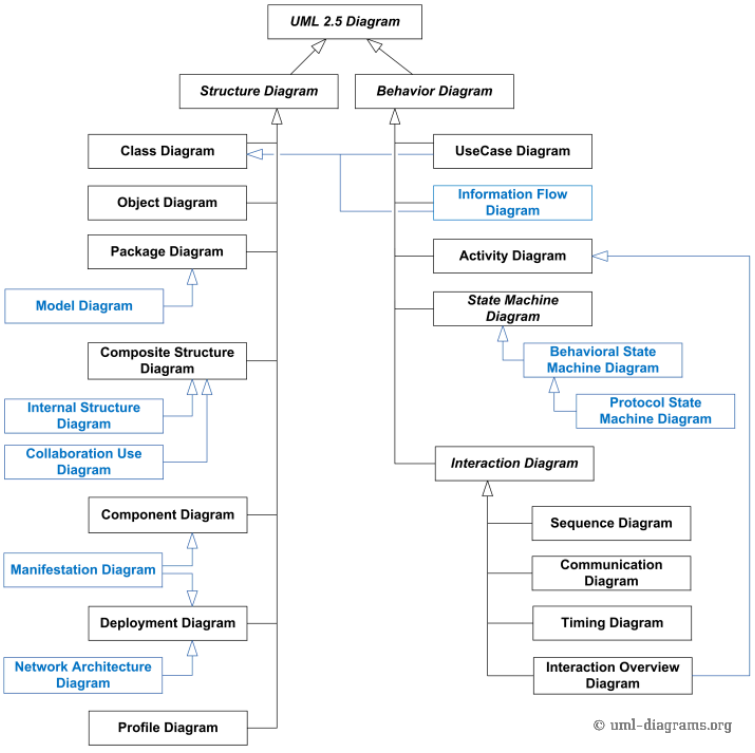

Figure 3.2: Classification of UML Diagrams [78]

3.4.1 PlantUML

PlantUML is an online tool that allows users to quickly and easily create diagrams like Sequence diagram, Usecase diagram, Class diagram, Object diagram, Activity diagram, Component diagram, Deployment diagram, State diagram, Timing diagram and non UML diagrams using a simple language [79].

3.5 postman

Postman simplifies the process of building and using APIs by streamlining each step of the API lifecycle and promoting collaboration among developers. This helps developers create better APIs in a shorter amount of time [80].

3.6 Visual Studio Code

Visual Studio Code is a source code editor developed by Microsoft. It supports different programming languages and provides features such as syntax highlighting, code completion, debugging capabilities, and version control integration [81].

3.6.1 github copilot

GitHub Copilot is a tool that uses AI to help users write code faster and better. It provides coding suggestions as you type, such as completing code lines or suggesting entire code blocks. Users can decide whether to accept, modify, or ignore these suggestions. The tool also includes a chat feature for users to ask for help with problem-solving, explanations for code snippets, or assistance with debugging identified bugs [82].

3.7 Node.js

Node.js is an open-source JavaScript runtime environment that works across different platforms. It allows developers to create servers, web applications, command-line tools, and scripts [83]

3.7.1 npm

`npm` (node package manager) is the package manager for Node.js, handling the distribution and management of Node.js packages and modules. `npm` is automatically installed on your system when installing Node.js. Each package in Node.js contains all the necessary files for a module. These modules are JavaScript libraries that can be used in projects [84].

3.7.2 express

Express is a web application framework in node.js that provides a set of features for web and mobile applications. It has minimal core features meant to be augmented through the use of Express middleware modules [85].

3.8 React

React is a JavaScript library used for building user interfaces for web applications. React is created and maintained by Facebook and is very popular in the web development community because of its simplicity, performance, and flexibility. The key concepts of react is component based architecture, virtual DOM, JSX and declarative syntax [86].

3.8.1 react-bootstrap

`react-bootstrap` is a frontend framework that has been rebuilt as `react-bootstrap` specifically for React. `react-bootstrap` is designed to be compatible with most of Bootstrap, maintaining accessibility and simplifying integration into React applications. This is why `react-bootstrap` is easy to use for developers for design within a React environment [87].

3.8.2 react-router-dom

`react-router-dom` is an npm package in the react library that is client and server sided. This package gives the user the possibility to implement dynamic routing in a web application. Additionally, it allows users to display pages and navigate them. There can be many pages or components in the application, but instead of refreshing the page, the content gets fetched based on the URL. This is what routing is [88].

3.9 react-bootstrap-icons

`react-bootstrap-icons` is a library with icons. It is an open source library that can be used with or without bootstrap in all projects [89].

3.10 axios

`axios` is a HTTP client used for node.js and browser. The same codebase can be ran in both node.js and the browser. Some of its main features is to make HTTP requests from node.js, posting html forms as JSON, Transform respons and request data and setting bandwith limits for node.js [90].

3.11 javascript

JavaScript is a programming language used to create interactive and dynamic content on websites. Javascript runs on web browsers and allows developers to add functionalitys like animations, form validation, and interactive elements. It can also manipulate, validate and calculate data and change and update HTML and CSS. Javascript has uses variables like, numbers, strings, arrays, objects and functions [91].

3.12 css

CSS, also called cascading style sheeets language is a language that describes how different elements is for example shown on a web page. Users can create rules that the web browser can parse into the selected styling based on the rule. CSS does not have anything to do with functionality, rather how things are styled. There is different ways to implement CSS into the HTML pages. The main way to do it is by writing the rules in a separate file, a .css file. It is also possible to do it directly in a HTML file using a style attribute [92].

3.13 Google Cloud Console

Google cloud console is a web admin UI to manage google cloud resources. It allows users to perform tasks like enabling APIs, creating/deleting buckets, manage files and set access controls. The console is accessed via a browser, allowing users to manage data quickly without setup or installation. It supports storage management tasks with advanced options available through a CLI or client libraries. Users can manage permissions, object level and upload, download or delete data [93].

3.14 Google Firebase

Google firebase is an application development platform suited for building and growing applications [94].

Rules

Rules is a well equipped security tool for the data in Firestore database and files in cloud Firestore. As the concept name implies, rules create certain boundaries to operations on data is tried executed. The rules are flexible in what they can enforce and how broadly they can impact an application. Rules are defined in Firebase, thus adding an additional security layer to the client of an application [95].

3.14.1 Authentication

Firebase authentication contains backend services in order to authenticate users to applications. It provides support for different authentication methods such as phone number, password, google, Facebook and more. Firebase authentication can also handle edge cases like account recovery and linking. Industry standards such as OAuth 2.0 and OpenID Connect are leveraged in the authentication solutions [96].

3.14.2 Firestore

Firestore is a NoSQL database solution within the Firebase platform, well suited for flexibility and scaling. The database can synchronize data for client and server development. Firebase operates in real-time ensuring synchronization across client applications through listeners. Firestore structure data with documents and collections, which allow complex and hierarchical data structures. It support indexing of the content to ensure optimized query performance. The indexing makes the query performance dependent on the size of the result set, rather than the size of the dataset [97].

3.14.3 Storage

Cloud storage for Firebase is a storing solution for files generated for or in applications. The object storage service offers security to file uploads and downloads, including Firebase authentication and rules. Uploads and downloads can be performed regardless of network quality, if they stop they will restart again to save time and bandwidth. Cloud storage will grow for an application that grows in userbase, leveraging googles storage real estate for automatic scaling [98].

3.15 github

Github is a web-based code software-management platform. It provides version control, allowing experimentation within the code without any risk as one can revert back to an earlier version. Github provides solutions to store, change, merge and collaborate on files including code. Members of a team can execute these tasks onto the repository, with possibilities for restrictions based on roles. Github offers DevOps automation solutions like code compilation, run tests, deploy and Ci/CD [99, 100].

3.16 Microsoft Azure Content Moderator and Google Cloud Vision

Microsoft Azure Content Moderator [101] and Google Cloud Vision [102] are content moderation APIs that has detection capabilities that can can be integrated into systems to filter out explicit content by screening in our case images and text.

Chapter 4

Method

4.1 Chapter Introduction

This chapter outlines the methodology adopted for this research. The research adopts a hybrid methodology, integrating principles of exploratory research [52, 103] and Design Science Research (DSR) [104]. This methodology is chosen for its flexibility and effectiveness in addressing complex problems where theoretical and practical insights are needed.

4.1.1 Hybrid approach

The hybrid approach is justified by the nature of the research question, which requires both theoretical and conceptual understanding. The Exploratory research method lays the groundwork by identifying the key issues regarding bias in AI. While the Design Science Research method focuses on developing, implementing, and evaluating solutions. This approach ensures a comprehensive understanding of the problem to address the complexities of reducing bias in AI datasets within GDPR constraints, by an extensive litterature review and developing a prototype that visualizes key concepts.

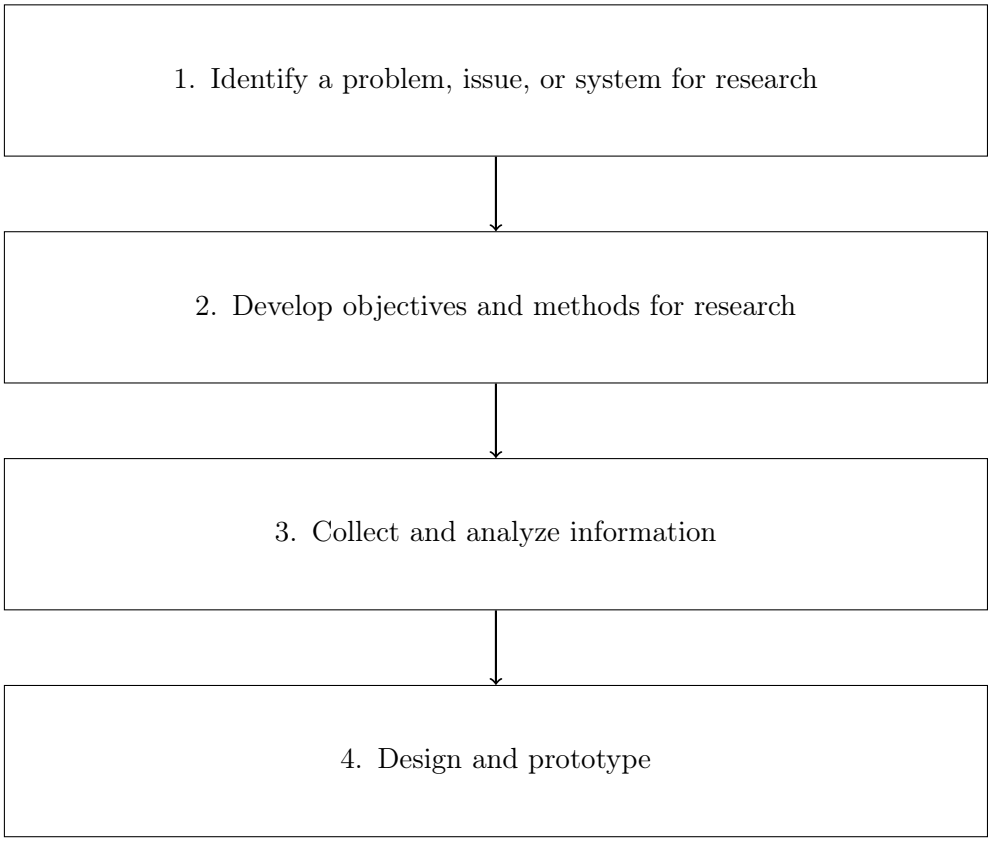

Figure 4.1: Hybrid Research Procedure

1. Identify a problem, issue, or system for research

The initial phase involves identifying the core problem through a preliminary research. The research will serve as the foundational basis for a comprehensive review, aimed at understanding how common bias is in AI and where it comes from. By examining a wide range of articles, insights are gained into the various approaches and techniques used by researchers to address bias in AI. This knowledge will serve as a very valuable starting point for the research, guiding efforts towards identifying strategies for leveraging user metadata to reduce bias in AI datasets.

2. Develop objectives and methods for research

Objectives and methods are defined to address the identified problems. The main goal is to leverage user metadata to reduce bias in AI datasets while ensuring compliance with GDPR regulations. First, existing research on where bias comes from and what it is, is reviewed. This lays the foundation for answering how user metadata can be leveraged to reduce bias in AI datasets. Then, a prototype is developed to demonstrate how GDPR-compliant solutions can be implemented using user metadata to reduce bias in datasets.

The hybrid method is selected because it combines theoretical understanding with practical solutions. It involves doing an extensive literature review and developing a prototype to visualize how our core problems can be solved.

3. Collect and analyze information

Information are going to be collected through an extensive literature review. This approach ensures an understanding of the theoretical and the practical challenges in addressing bias in AI datasets.

Google Scholar [105] will serve as the primary search tool. Additionally, Oria [106], Semantic Scholar [107], and Connected Papers [108] will be utilized to broaden the search.

The literature search will be guided by a selected set of keywords and phrases to find the most relevant studies. These will include 'dataset', 'metadata', 'bias', 'annotation', 'crowdsourcing' and 'reduce bias' etc. Each keyword is selected to develop an understanding of the AI field, how training data works and where BIAS comes from. To ensure a thorough literature review, the search strategy will also involve a combination of these keywords. For instance, searches like 'dataset AND metadata', 'bias AND annotation', and 'annotation AND reduce bias' will be performed.

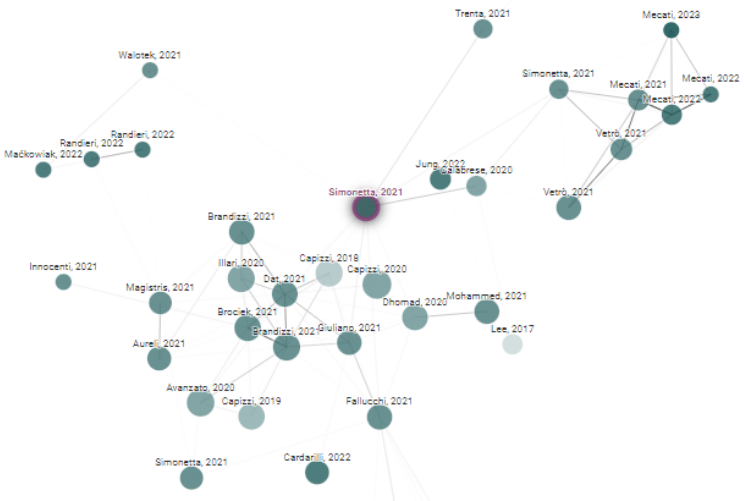

Figure 4.2: Graph visualizing connected articles [108]

Connected Papers [108] will be used as a tool in our research methodology. Once a relevant article is identified, Connected Papers will expand our understanding of the research on that article. This approach allows us to find and visualize other articles of related research. For instance, when an article is discovered, Connected Papers generates a visual graph of connected research articles as shown in figure 4.2. This graph not only highlights direct citations but also identifies papers that are similar, methodology, or cited works, even if they do not cite the article directly.

4. Design and prototype

The design and development of the prototype starts with a thorough analysis of the General Data Protection Regulation (GDPR) articles [32]. Based on our understanding of these articles, a set of Business requirements will be created. After the business requirements have been created, a set of user requirements will be created, directly translated from the business requirements. User requirements will focus on the functionality and usability expected by users who will interact with the system.

The next phase involves translating user requirements into system requirements. System requirements are more technical and detailed specifications that describe what the system must do and the conditions it must meet. From the system and user requirements, a set of functional requirements will be created. These requirements detail the specific behaviors, actions, or functions the system must be able to perform. In addition to functional requirements, a set of non functional requirements will be defined. These requirements will address the system's performance, such as security, reliability, scalability, and efficiency.

After all the requirements are created, they will be prioritized. The MoSCoW method will be applied in this phase. This will give an indication as to how important the different requirements are for the initial roll out of the system. The prioritization will be reflected in the prototype and highlighted in the project.

A semi-automated approach will be attempted for creating linked requirements and a set of use case descriptions. One for each set of linked requirements. These use case descriptions will illustrate typical user interactions with the system and the system's responses. To visually show these interactions, use case diagrams will be developed. The use case diagrams help visualize the scope of the system and understand the relationships between user actions and system responses.

Following the use case diagrams, a set of sequence diagrams will be created, one for each use case description. Sequence diagrams will be used to detail the sequence of events that occur within the system when a particular use case is executed.

Once all this is done, the requirements intended for the prototype will be used to create high-level diagram(s) (component or similar) to map out the major parts of the system and their interactions before moving on to the prototype development. The high-level diagrams will be followed by more detailed mid-level component and sequence diagrams. After creating the high-level and mid-level diagrams, the development of the prototype begins. This phase involves turning the designs into a prototype.

The purpose of the prototype is to demonstrate the concept, thus to ensure its ready relatively quickly, testing, devops, extensive error handling and similar practices will not be prioritized. As the focus will be to develop a simple prototype, the following technologies will be used:

- IDE: Visual Studio Code
- Version Control: github
- Programming Language: javascript
- Frontend Framework: React
- Backend Environment: Node.js
- Database and Hosting: Google Firebase
- AI Assistance: github copilot

4.1.2 Text Language Washing with ChatGPT

To ensure that our text is clear, concise, and easy to understand, we will use ChatGPT [109], to perform text language washing. First, we will input our text into ChatGPT [109] and request it to generate a revised version. We will then carefully review the output to ensure that it accurately reflects our intended message and adheres to our writing style.

Chapter 5

Results

5.1 Literature review

5.1.1 Significant demand for large amounts of quality training data

The literature review is mainly focused on today's status regarding training-data for AI, BIAS in AI, where BIAS in AI originates, thus how BIAS can be mitigated and crowd-sourcing as a solution to develop training data. AI systems require large volumes of high-quality labeled data to perform as efficiently and accurately as intended [110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 12, 6, 22, 20, 123, 124, 125, 126]. The literature review makes it clear that there currently exists a substantial demand for large and diverse datasets to train various AI models [30, 10, 110, 111, 112, 113, 114, 115, 127, 116, 117, 128, 119, 129, 25, 130, 131, 120, 118, 132, 121, 133, 134, 135, 136, 122, 12, 6, 22, 20, 123, 124, 125, 126, 137, 4, 3]. However it is of critical importance that the quality, availability and diversity of training data is sufficient. AI models reflect the quality of their training data, thus low-quality, incomplete or unbalanced data leads to poor model performance [138, 10, 14, 139, 110, 113, 140, 114, 127, 116, 128, 119, 129, 25, 130, 131, 120, 132, 133, 134, 135, 6, 122, 12, 141, 124, 142]. According to the literature, training data rarely satisfies these requirements. Data scarcity and costs of data curation provide challenges for AI developers. High-quality datasets are costly and hard to acquire, as they frequently lack diversity and have insufficient labeling [14, 139, 140, 114, 128, 129, 25, 130, 131, 120]. Due to this lack of satisfying datasets, increased investment in collecting high-quality training data is needed [6, 12].

5.1.2 Utilizing Crowd-Sourcing for Efficient Data Annotation in AI Development

Efficient and effective data annotation processes are necessary but challenging. High-quality datasets is costly, time-consuming and often requiring a lot human annotation and validation to create. Annotation can slow down the development of more advanced AI models [114, 120, 118, 25, 132, 122]. Balancing the cost while maintaining high-quality annotations is a challenge, especially for smaller organizations or academic institutions [120]. To meet this challenge crowd-sourcing can be a effective method for developing large amounts of quality-data. Crowd-sourcing improves the speed and efficiency of data labeling tasks, and is becoming an important part of AI development. Enabling the creation of large, diverse and high-quality data datasets necessary [17, 113, 114, 115, 117, 119, 143, 25, 120, 118, 122].
Crowd-sourcing has proven to work well, leveraging the "wisdom of the crowd" to produce outcomes for a variety of tasks, including data annotation and forecasting, that have exceeded the accuracy of their annotations over individual experts when interpreting medical image data while reducing costs and increasing efficiency [117]. Important to highlight though, the success of crowd-sourcing depends significantly on the design of annotation guidelines and the management of crowd worker incentives. Providing detailed instructions and quality control mechanisms are crucial to ensure the accuracy and reliability of the data [138, 110, 117, 144]. There is however no lack of challenges to overcome with crowd-sourcing. The numerous annotators create a need for ensuring consistent high-quality, which could require proper guidance and expert reviews. Crowd-sourcing workers tend to be demographically skewed, influencing the representation of populations in datasets [117, 144, 120, 132, 119, 115].

5.1.3 Addressing Bias in AI Challenges and Strategies for Mitigation

The literature review revealed the widespread issue of BIAS in AI, highlighting that biases in training data are often only identified after AI models are deployed [145, 30, 146, 17, 137, 10, 147, 14, 139, 13, 115, 127, 116, 119, 129, 120, 118, 134, 135, 136, 12]. These biases can lead to AI models that perpetuate and even amplify societal stereotypes and inequalities [30, 146, 13]. There are many examples of AI causing damage in peoples lives based on their appearance and other attributes, such as criminal justice, healthcare, hiring, facial recognition, education, loan approvals, and housing applications [10, 139, 136, 127, 147, 14] Despite the challenges, there is no straightforward solution to mitigating BIAS in AI. The literature highlights the importance of involving various stakeholders, carefully evaluating data sources and AI models to identify and address biases [10, 13, 135]. BIAS in AI can stem from datasets that is incomplete or unrepresentative of specific groups such as minorities. This leads to AI models that consistently favour the elements most represented in its training data, such as people with a conservative appearance or poor performance of object recognition on household items more common in low-income countries [145, 146, 115, 127, 116, 119, 120, 118, 128, 12, 141, 137, 10, 147, 139, 13, 30].

While BIAS can stem from the contents of training data, the literature review did also reveal another way datasets can forward biases. In the annotation process practitioners and annotators could introduce subjectivity, indicating that diversity in the pool of human annotators can further mitigate BIAS applied in datasets and its negative effects [115, 127, 12, 17, 10, 119]. The literature review revealed that mitigating biases requires robust methods to assess and correct for BIAS in datasets. This includes diverse data collection, careful annotation, and continuous monitoring for BIAS [17, 120, 134]. There is no perfect algorithm to detect all biases, so manual review and correction are challenging but essential [145]. Analysis of language models suggest the language models which perform better on certain fairness benchmarks tend to have worse gender BIAS [135].

5.2 Requirements

The project requirements are mainly derived from the GDPR regulations [32]. These requirement is formulated with the help of chatGPT [109]. Each GDPR article was reviewed and translated into business requirements. Any requirements that were not relevant to our project were excluded. The requirements that were excluded are mainly from article 50 - 99 [32]. The reason for that is because they are mainly focusing on supervisory authorities, cooperation's among EU member states and the European commission.

A total of 266 requirements were formulated, categorized into Business, User, System, Functional, and Non-Functional requirements. These requirements are all rooted in the business needs and are ranked accordingly. Some significant requirements are highlighted in this section, with the complete list available in the Appendix A. Additionally, the complete list of 266 requirements provides a guide for the development and implementation of the project to meet GDPR standards.

5.2.1 Prioritization

All the requirements formulated was later manually prioritized using the "MoSCoW" method. The decision making for each prioritization was based on what software the initial roll out of a full-worthy system must , should, could and wont have. Due to MoSCoWs flexibility when it comes to the importance of each category, which could lead to confusion, we also added a short reasoning for most of the priorities. With this approach requirements like BR38, stating "The system shall inform the data subject before lifting any restriction of processing on their personal data" were prioritized as should and could have on the basis that they are manually manageable during the initial roll-out. Vital functionalities like proper user data management and annotation task related requirements were prioritized as must have. Few but some requirements like NFR25 stating "The application shall comply with the Web Content Accessibility Guidelines (WCAG) 2.1 Level AA, ensuring that users with disabilities can access and use the system effectively." were prioritized as wont have, reasoned by its lack of relevance and proportional work relative to benefits in the early stage of the systems production lifetime.

5.2.2 Business requirements

The business requirements were derived directly from the GDPR articles, specifically from article 1 to 50. For example, art. 5.1 a [32].

"Personal data shall be: processed lawfully, fairly and in a transparent manner in relation to the data subject ('lawfulness, fairness and transparency') [148]"

were translated as

"The system must process personal data only after obtaining clear, affirmative consent from the data subjects, ensuring compliance with legal standards and maintaining transparency about data usage.".

The business requirements are mainly focused on what the system shall or must do to comply with the GDPR rules. Below is some of the most important business requirements.

Table 5.1: Business requirements

ID	Requirement	Article	Priority	Priority Reasoning
BR5	The system must collect and process only the personal data that is necessary and relevant to the specified purposes, and no more.	5.1.c	Must have	GDPR
BR7	The system must process personal data only after obtaining clear, affirmative consent from the data subjects, ensuring compliance with legal standards and maintaining transparency about data usage.	5.1.a	Must have	GDPR
BR1	The system must verify and document the consent of data subjects for the processing of their personal data for specific purposes, in compliance with legal obligations.	6.1.a, 7.1	Must have	GDPR
BR10	The system must present requests for consent in a clear, distinguishable manner from other matters, using intelligible and easily accessible form, employing clear and plain language.	7.2	Must have	GDPR
BR26	The system must inform the data subject about any intended further processing of personal data for a purpose other than that for which they were collected, prior to that further processing.	13.3	Must have	GDPR
BR38	The system shall inform the data subject before lifting any restriction of processing on their personal data.	18.3	Should have	Required for GDPR compliance, manually manageable in the initial roll-out of the project
BR157	The system must adhere to standards for providing information and communications in a manner that is easily understandable by all users, including children.	12.1	Must have	GDPR

5.2.3 User requirements

User requirements are designed to ensure that the system's interface and functionalities are aligned with the expectations and needs of users. These user requirements is translated from the business requirements into specific, actionable functionalities that directly affect the user interaction with the system. They focus on making sure that user actions meet GDPR requirements by prioritizing usability, accessibility, and efficiency. Here is some of our most important user requirements;

Table 5.2: User requirements

ID	Requirement	Source	Priority	Priority Reasoning
UR1	The user must be able to control the types of personal data provided to the system, ensuring that only necessary information relevant to the users interactions with the platform is collected and processed.	BR5	Must have	UR16, UR27
UR2	The user must be presented with clear consent requests, providing detailed information about data usage before any processing of personal data occurs.	BR7	Must have	Partly covered in UR27
UR27	The user must be able to easily give, manage, and withdraw consent within the system, with a user experience prioritizing clarity and ease of use.	BR152	Must have	Key feature for compliance with GDPR
UR13	The user must be informed about any intended further processing of their personal data for purposes other than those for which it was originally collected, prior to that further processing	BR26	Must have	Not proportionately relevant to automate in the start-up phase. Manually fixed before the platform has gained significant traction
UR3	The user must be prompted to provide explicit consent for the processing of special categories of personal data, such as health information or political opinions, for specific purposes. The system should maintain clear documentation of user consent in compliance with relevant laws and regulations	BR15	Must have	Legal claim
UR7	The user must be informed about the process to withdraw their consent at any time before giving consent	BR9	Must have	Easy to implement, partially covered in UR27

5.2.4 System requirements

The system requirements are directly translated from the user requirements because it ensures that the functionality aligns with what users need. Here is some of the most important system requirements.

Table 5.3: System requirements

ID	Requirement	Source	Priority	Priority Reasoning
SR1	The system must include user-configurable settings that allow users to specify the types of personal data collected and processed by the platform. These settings should provide detailed control over the information shared with the system, ensuring that only necessary data relevant to the user's interactions with the platform are collected and processed	UR1	Must have	Not technically difficult to implement. Partually covered in UR27 og UR16
SR2	The system must present clear and detailed consent requests to users, providing information about how their personal data will be used before any processing occurs. The consent interface should be designed to ensure user understanding and transparency regarding data usage.	UR2	Must have	Covered in UR27
SR7	The system must present consent requests to users in a clear, easily distinguishable manner from other content, using an accessible format and written in plain language.	UR8	Must have	Covered in UR27
SR11	The system must inform users about any intended further processing of their personal data for purposes other than those for which it was originally collected, prior to that further processing.	UR13	Must have	Not proportionately relevant to automate in the start-up phase. Manually fixed before the platform has gained significant traction
SR3	The system must prompt users for explicit consent to process sensitive personal data and maintain clear documentation of user consent in compliance with relevant laws and regulations.	UR3	Must have	Covered in UR3
SR10	The system must provide users with the identity and contact details of the controller, the data protection officer (if applicable), the purposes of the processing, the legal basis for processing, and the recipients or categories of recipients of their personal data at the time the data is obtained.	UR12	Must have	Easy to implement, key for GDPR compliance

5.2.5 Functional requirements

The functional requirements are directly translated from both user requirements and software requirements because it ensures that every level of requirements helps define the systems functionality. Here is some of the most important functional requirements;

Table 5.4: Functional requirements

ID	Requirement	Source	Priority	Priority Reasoning
FR5	The system must include a user-accessible feature that enables users to request the erasure of their personal data under specific conditions, such as when the data is no longer necessary for its original purposes, consent is withdrawn, or the data has been unlawfully processed. The feature should facilitate user-initiated data erasure while ensuring compliance with relevant data protection regulations	UR5, SR5	Must have	Covered in UR27
FR20	The system must provide users with the capability to request and receive their personal data in a structured, commonly used, and machine-readable format. Furthermore, it shall include functionality to facilitate the seamless transmission of this data to another controller without encountering obstacles, especially in cases where processing is predicated on consent or a contract and conducted through automated means.	UR22, SR19	Must have	Covered in UR22
FR35	The system shall enable users to customize datasets by applying filters based on image tags, allowing users to refine and tailor the dataset according to their specific requirements.	UR37	Should have	Covered in UR37
FR36	The system shall allow users to customize datasets by applying filters based on meta-information about the annotators, enabling users to refine and tailor the dataset according to specific criteria related to the annotators' information.	UR38	Should have	Covered in UR38
FR37	The system shall provide users with access to statistics on various metrics of meta-information about the annotators within the datasets they create, allowing users to analyze and understand the distribution and characteristics of annotator-related data.	UR39	Could have	Covered in UR39

5.2.6 Non functional requirements

The non functional requirements were created with the purpose of covering a list of quality attributes as shown in table 5.6. The quality attributes and requirements was consulted with "system requirement specification generator" provided by chatGPT [109] after providing the model with a detailed and inclusive description of the project. The process resulted in 67 non functional requirements for the intended system. Table 5.4 displays some of the most important ones.

Table 5.5: Non Functional requirements

ID	Short	Requirement	Priority	Priority Reasoning
NFR5	Search Functionality	Search queries within the platform, including searches for images, annotation tasks, or datasets, should return results within 2 seconds under typical load conditions.	Must have	If this isn't fulfilled in the initial rollout of the project its probably not suited for scaling.
NFR13	Data Consistency	Ensure all data, including image uploads, annotations, and user-generated datasets, is consistently stored and retrieved without errors or corruption. Any data submitted to the system should be accurately saved and available for subsequent retrieval and use.	Must have	Quality = Value
NFR32	Search Functionality	The system shall offer robust search functionality, allowing users to quickly find images, annotation tasks, or datasets, with filters to refine their search effectively.	Must have	In this way users can choose more reliable sources
NFR41	Data Anonymization	When processing user data for statistical or analytical purposes, ensure that personal identifiers are anonymized to protect user privacy and comply with relevant data protection regulations.	Must have	Key to ensure privacy and lawfull processing of data
NFR44	Data Minimization	Consistent with GDPR principles, the system should collect only the data that is necessary for the specified purposes and ensure that such data is not retained longer than necessary.	Must have	Statistical purposes

Table 5.6: Mapping of Quality Attributes to Non-Functional Requirements

Quality Attribute	**Related NFRs**
Performance	NFR1 - NFR5
Scalability	NFR6 - NFR11
Reliability	NFR12 - NFR16
Availability	NFR17 - NFR22
Usability	NFR23 - NFR32
Security	NFR33 - NFR42
Compliance	NFR43 - NFR49
Maintainability	NFR50 - NFR55
Interoperability	NFR56 - NFR58
Portability	NFR59 - NFR60
Data integrity	NFR61 - NFR63
Privacy	NFR64 - NFR68

5.3 Data Processing and Visualization

This section outlines the initial stages of data processing and the visualization techniques employed to interpret the requirement datasets. Through the integration with Google Sheets/ gspread [59], data from various GDPR articles, the developer team and quality attributes, were translated into actionable requirements and subsequently visualized to provide preliminary insights. (see Appendix C)

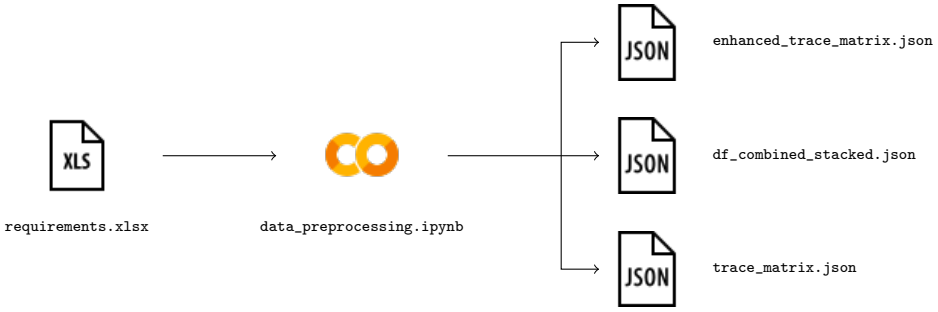

Figure 5.1: Workflow for data processing

5.3.1 Data Import, Cleaning and Restructuring

Data was imported from `requirements.xlsx` (Appendix A), which contained multiple tabs categorizing different types of requirements, to `data_preprocessing.ipynb` (Appendix C).
Using Python and pandas in a Colab environment, the data was cleaned and structured to ensure accuracy and relevancy. This process involved removing duplicates, filling missing values, and merging data from multiple tabs based on specific keys.

	id	req	full_trace	priority	frame
0	FR1	The system shall provide users with a settings...	5.1.c -> BR5 -> UR1	Must have	FR
1	FR1	The system shall provide users with a settings...	5.1.c -> BR5 -> UR1	Must have	FR
2	FR2	The system shall display consent requests to u...	5.1.a -> BR7 -> UR2	Must have	FR
3	FR2	The system shall display consent requests to u...	5.1.a -> BR7 -> UR2	Must have	FR
4	FR3	The system shall prompt users to explicitly pr...	9.2.a -> BR15 -> UR3	Must have	FR
...	...	...	...	...	...
236	BR53	The system shall integrate data protection pri...	25.1	Must have	BR
237	BR53	The system shall integrate data protection pri...	25.1	Must have	BR
238	BR54	Demonstration of compliance with data protecti...	25.3	Could have	BR
239	BR136	Controllers or processors must document the as...	49.6	Must have	BR
240	BR172	The system shall include controls to ensure pr...	10	Should have	BR

241 rows × 5 columns

Figure 5.2: Snapshot of `enhanced_trace_matrix`

5.3.2 Visualization and Grouping

To better understand the relationships and distribution of the requirements from `enhanced_trace_matrix.json` (Appendix F), they were visualized in `visualization.ipynb` (Appendix G).

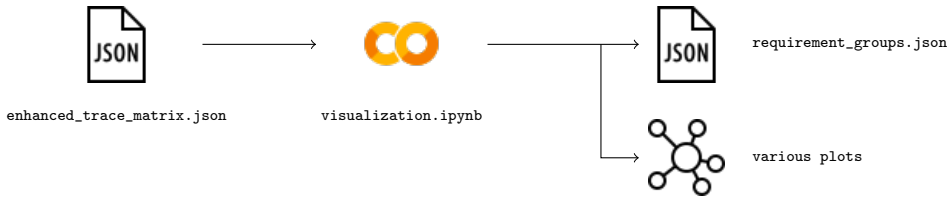

Figure 5.3: Workflow visualization

Graphs, Plots and Network Diagrams:
Utilizing networkx and plotly, for interactive plots that allow for an analysis of the requirements distribution across different categories.

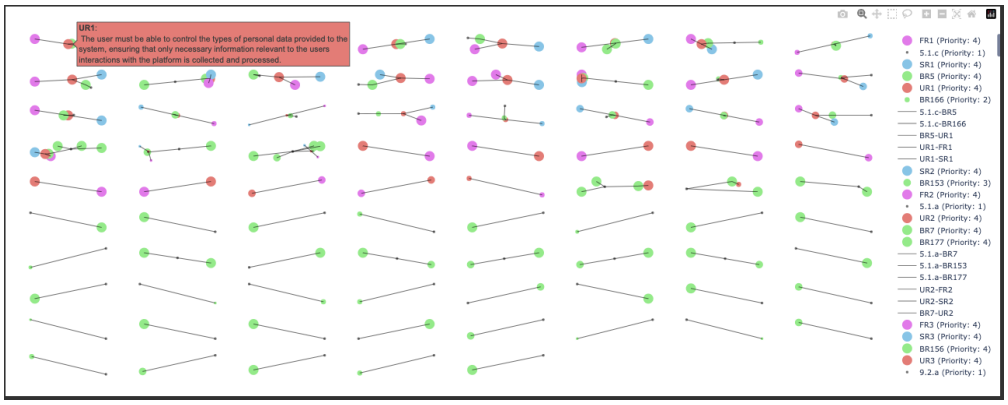

Figure 5.4: Snapshot of interactive graph plot

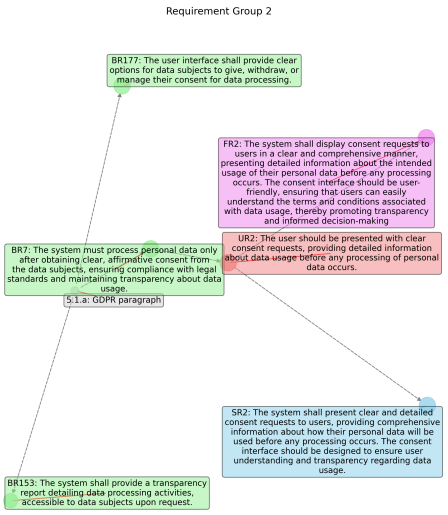

Figure 5.5: Plot example, Requirement Group 2

The linked requirements from the graph structures were stored as requirement groups in `requirement_groups.json` (Appendix H)

5.4 Natural Language Processing (NLP)

For further enhancement of the requirements, and in an attempt to automate the writing of the use case descriptions, several NLP techniques were applied. This involved multiple stages, each aimed at refining the data to extract more meaningful information from `requirement_groups.json` (Appendix H). The steps are shown in `nlp.ipynb` (Appendix I) and the outputs are available in `requirement_groups_with_summaries.json` and
`enriched_groups_with_summaries.json`.

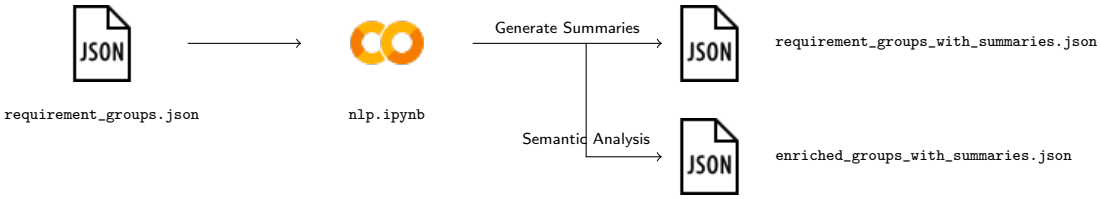

Figure 5.6: Workflow for NLP

5.4.1 Text Processing and Embedding

Initial NLP operations included using the `t5-large` transformer model for generating text summaries, which helped in clustering similar requirements. The notebook `nlp.ipynb` (Appendix I) processes the data from `requirement_groups.json` (Appendix H), applies the model to generate concise summaries, and outputs these to
`requirement_groups_with_summaries.json` (Appendix J).

5.4.2 Semantic Enhancement

Further semantic analysis was conducted using the `en_core_web_trf` model from spaCy to enrich the requirements data. This step extracted entities, keywords, dependency parsing, tags, noun phrases, sentences, lemmas and relationships in order to enhance the understanding and categorization of the data. The results of this analysis were stored in
`enriched_groups_with_summaries.json` (Appendix K), providing a detailed overview of the semantic landscape of the requirements.

5.5 Integration with OpenAI

While the initial stages of applying Natural Language Processing (NLP) techniques showed promising results, the time constraints inherent in the scope of this thesis made it necessary with a more immediate solution than further fine-tuning of the NLP approach. Consequently, the OpenAI API was utilized for automating the generation of use case descriptions.

A simple comparison of some requests/responses was conducted between the `gpt-4-turbo` and `gpt-3.5-turbo` models in the OpenAI Playground. The comparison was aimed at determining which model would yield the most relevant responses for the specific needs. The results indicated that `gpt-3.5-turbo` provided the most fitting responses for the context of the project. Thus, this model was selected to assist in generating the text for the use case descriptions, which is documented in `enriched_groups_with_summaries.json` (Appendix K).

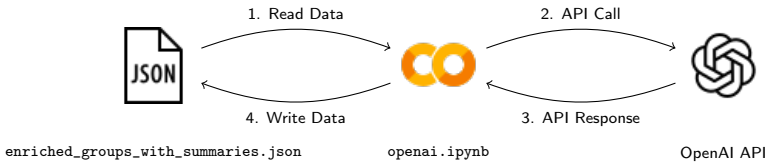

Figure 5.7: Workflow OpenAI API

The model was given the following context:

```
You are an experienced and professional software architect.
Your task is to convert software requirements into structured use case descriptions, formatted as key-value pairs for easy parsing.
Utilize the following keys: ID, Title, Brief description, Primary Actor, Supporting Actor(s), Trigger, Pre-conditions, Basic Flow, Assumptions, Alternate Flow, and Post-condition.
Each key should have a concise value.
Present the Basic and Alternate flows as arrays of actions and responses.
Ensure the format adheres to the provided example.

Example format:
{
    "ID": "UR1",
    "Sources": "5.1.c -> BR5 -> UR1"
    "Title": "Customize User Settings",
    "Brief description": "Allows users to customize settings within their user profile.",
    "Primary Actor": "User",
    "Supporting Actor(s)": "Database",
    "Trigger": "User logs into the application.",
    "Pre conditions": "User must have valid credentials.",
    "Basic Flow": [
        {"Line": 1, "Actor action": "User selects 'Settings'", "System response": "System displays settings options"},
        {"Line": 2, "Actor action": "User changes language setting", "System response": "System updates language"}
    ],
    "Assumptions": "The system supports multiple languages.",
    "Alternate flow": [
        {"Line": 1, "Actor action": "User selects 'Settings'", "System response": "System fails to load settings"},
        {"Line": 2, "Actor action": "User retries", "System response": "System displays error message"}
    ],
    "Post condition": "User settings updated as per user inputs."
}
```

Figure 5.8: Snapshot of the context for `gpt-3.5-turbo`

The rest of the code is documented in `openai.ipynb` (Appendix L).

5.6 Use Case Descriptions

This section details the process of creating the layout and entering data for the use case descriptions. The libraries gspread, gspread_dataframe, and gspread_formatting were utilized for interactions with Google Sheets. The specific procedures and code are documented in `use_case_descriptions.ipynb` (Appendix M). The data from `enriched_groups_with_summaries.json` was systematically written to `Use Case Descriptions - All.xlsx` (Appendix B).

Figure 5.9: Workflow for generating Use Case Descriptions

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
1			2024-04-25

Use Case ID:	UC1
Brief Description:	Allows users to specify types of personal data collected and processed by the platform.
Primary Actor:	User
Secondary actor(s):	System
Trigger:	User accesses data collection settings.
Pre conditions:	User is authenticated.
Sources:	5.1.c -> BR5 -> UR1 -> FR1 5.1.c -> BR5 -> UR1 -> SR1 5.1.c -> BR166

Basic Flow:		
Assumptions:	The system identifies and distinguishes between necessary and unnecessary data.	
Line:	**Actor Action:**	**System Response:**
1	User navigates to 'Data Collection Settings'	System displays data collection options
2	User selects data types to collect	System updates data collection preferences immediately
Alternate flow:		
1	User navigates to 'Data Collection Settings'	System fails to load settings
2	User retries	System displays error message
Post condition:	Data collection processes adjusted according to user's configuration.	

Figure 5.10: Use case description, UC1

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
1			2024-04-25

Use Case ID:	UC2
Brief Description:	System obtains clear and affirmative consent from users before processing personal data.
Primary Actor:	User
Secondary actor(s):	System
Trigger:	User accesses system or performs an action that requires personal data processing.
Pre conditions:	User must be logged in or identified within the system.
Sources:	5.1.a -> BR7 -> UR2 -> FR2 5.1.a -> BR7 -> UR2 -> SR2 5.1.a -> BR153 5.1.a -> BR177

Basic Flow:		
Assumptions:	Users have the capability to provide or deny consent.	
Line:	**Actor Action:**	**System Response:**
1	User navigates to data processing feature	System displays consent request with detailed data usage information
2	User reviews terms and conditions	System provides easy-to-understand explanations
3	User gives affirmative consent	System records user's consent and proceeds with data processing
Alternate flow:		
1	User navigates to data processing feature	System does not display consent request
2	User declines consent	System restricts data processing and provides options for user to review settings or preferences
Post condition:	System processes personal data based on the obtained user consent.	

Figure 5.11: Use case description, UC2

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
1			2024-04-25

Use Case ID:	UC25
Brief Description:	Allows users to manage consent preferences through intuitive interfaces.
Primary Actor:	User
Secondary actor(s):	User Interface
Trigger:	User accesses the consent management section.
Pre conditions:	User must be logged in.
Sources:	6.1.a -> BR152 -> UR27 -> FR25 6.1.a -> BR152 -> UR27 -> SR25 6.1.a -> BR1 6.1.a -> BR162 6.1.a -> BR175

Basic Flow:		
Assumptions:	The system provides clear and easily accessible options for each type of consent.	
Line:	**Actor Action:**	**System Response:**
1	User navigates to 'Consent Settings'	System displays consent options
2	User selects 'Give Consent'	
3	User sets preferences and confirms	System saves user preferences
Alternate flow:		
1	User navigates to 'Consent Settings'	System fails to load consent options
2	User retries or refreshes page	System successfully loads consent options
Post condition:	User consent preferences updated as per user interaction.	

Figure 5.12: Use case description, UC25

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
2			2024-05-25

Use Case ID:	UC7
Brief Description:	Ensure clear and distinguishable consent requests in plain language.
Primary Actor:	User
Secondary actor(s):	System
Trigger:	User interacts with the system.
Pre conditions:	User is in a situation where consent is required.
Sources:	7.2 -> BR10 -> UR8 -> FR7 7.2 -> BR10 -> UR8 -> SR7 7.2 -> BR168

Basic Flow:		
Assumptions:	Users have the ability to understand the presented information.	
Line:	**Actor Action:**	**System Response:**
1		System presents consent request in plain language
2	User reads and understands the request	
3	User provides consent	System acknowledges consent
Alternate flow:		
Post condition:	User's consent is clearly obtained and acknowledged by the system.	

Figure 5.13: Use case description, UC7

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
2			2024-05-25

Use Case ID:	UC11
Brief Description:	System notifies the data subject about any intended further processing of personal data for a different purpose.
Primary Actor:	System
Secondary actor(s):	Data Subject
Trigger:	System intends to process personal data for a different purpose than initially collected.
Pre conditions:	System has collected personal data.
Sources:	13.3 -> BR26 -> UR13 -> FR11 13.3 -> BR26 -> UR13 -> SR11

Basic Flow:		
Assumptions:	Data subject has provided valid contact information for communication.	
Line:	**Actor Action:**	**System Response:**
1	System identifies need for further processing for a new purpose	
2	System prepares notification to inform data subject	
3	System sends notification to data subject detailing the new purpose	
Alternate flow:		
1		System fails to send notification
2	System logs failure for further investigation	
3	Retry: send notification	
Post condition:	Data subject is informed about the intended further processing of personal data for a new purpose.	

Figure 5.14: Use case description, UC11

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
2			2024-05-25

Use Case ID:	UC18
Brief Description:	Inform the user before lifting any restriction on the processing of their personal data.
Primary Actor:	System
Secondary actor(s):	User
Trigger:	A restriction on the processing of user's personal data is about to be lifted.
Pre conditions:	The system has imposed a restriction on the processing of the user's personal data.
Sources:	18.3 -> BR38 -> UR20 -> FR18 18.3 -> BR38 -> UR20 -> SR18

Basic Flow:		
Assumptions:	The system is capable of notifying the user through appropriate communication channels.	
Line:	**Actor Action:**	**Response:**
1	System prepares to lift the restriction	
2	System notifies the user about the upcoming change	User acknowledges the notification
3	System proceeds to lift the restriction	
Alternate flow:		
Post condition:	User is informed and grants consent before the restriction on personal data processing is lifted.	

Figure 5.15: Use case description, UC18

Use Case Description			
Version:	**Description:**	**Author:**	**Date:**
2			2024-05-25

Use Case ID:	UC39
Brief Description:	System will display data processing information and communications in a concise, transparent, and accessible manner.
Primary Actor:	Users, Children
Secondary actor(s):	System
Trigger:	User interacts with data processing features.
Pre conditions:	User accesses data processing functionalities.
Sources:	12.1 -> BR19 -> UR9 12.1 -> BR157

Basic Flow:		
Assumptions:	Users have the intent and ability to understand the information provided.	
Line:	**Actor Action:**	**System Response:**
1	User requests data processing information	System displays information in a clear and concise manner
2	User seeks communication related to data processing	System provides easily accessible communications
Alternate flow:		
1	User encounters complex information	System offers simplified explanations
2	Child user accesses information	System presents information suitable for children
Post condition:	Users receive data processing information and communications following standards for clarity and accessibility.	

Figure 5.16: Use case description, UC39

5.7 High-level diagrams

This section details the process of generating the PlantUML code for diagrams and the related diagrams. The code for the notebook is documented in `diagrams.ipynb` (Appendix N). The use cases from `Use Case Descriptions - All.xlsx` (Appendix B) was used for generating the diagrams in the `Diagrams folder` (Appendix O).

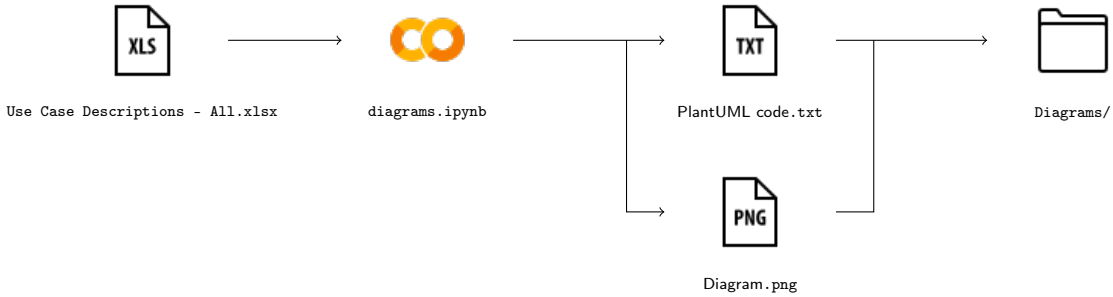

Figure 5.17: Workflow for generating diagrams

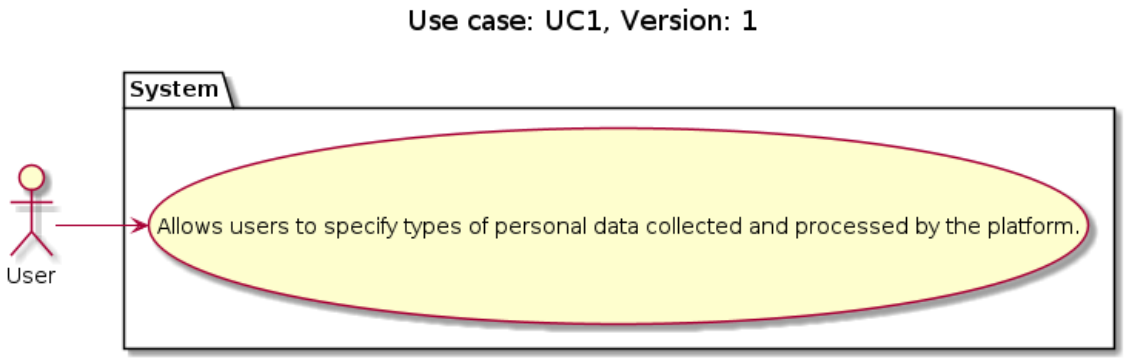

Figure 5.18: Use case diagram for case 1

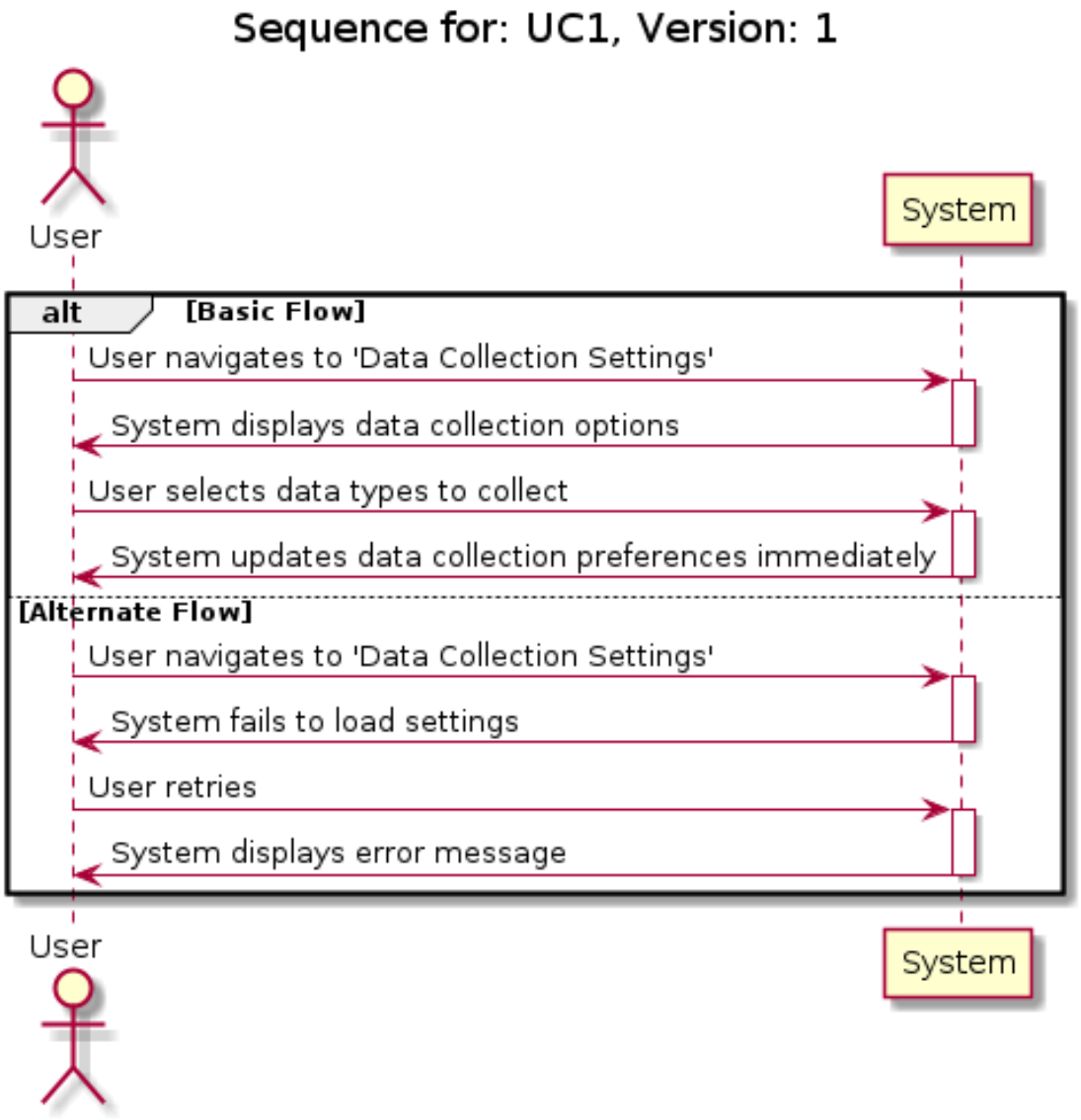

Figure 5.19: Sequence diagram for case 1

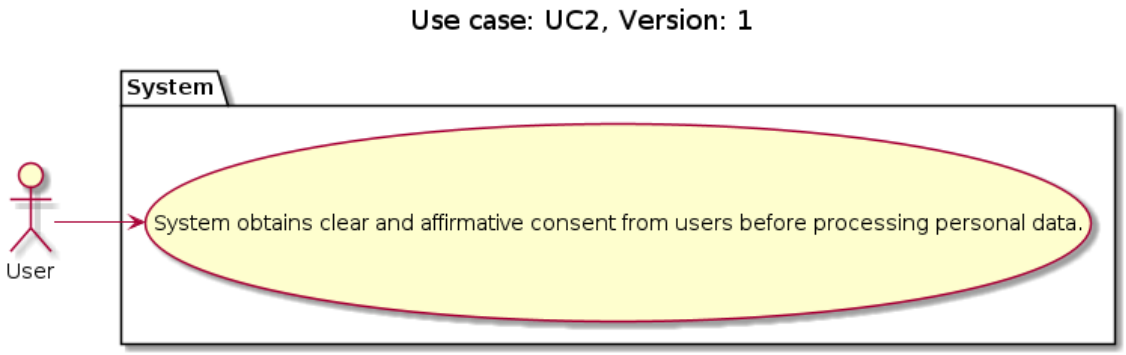

Figure 5.20: Use case diagram for case 2

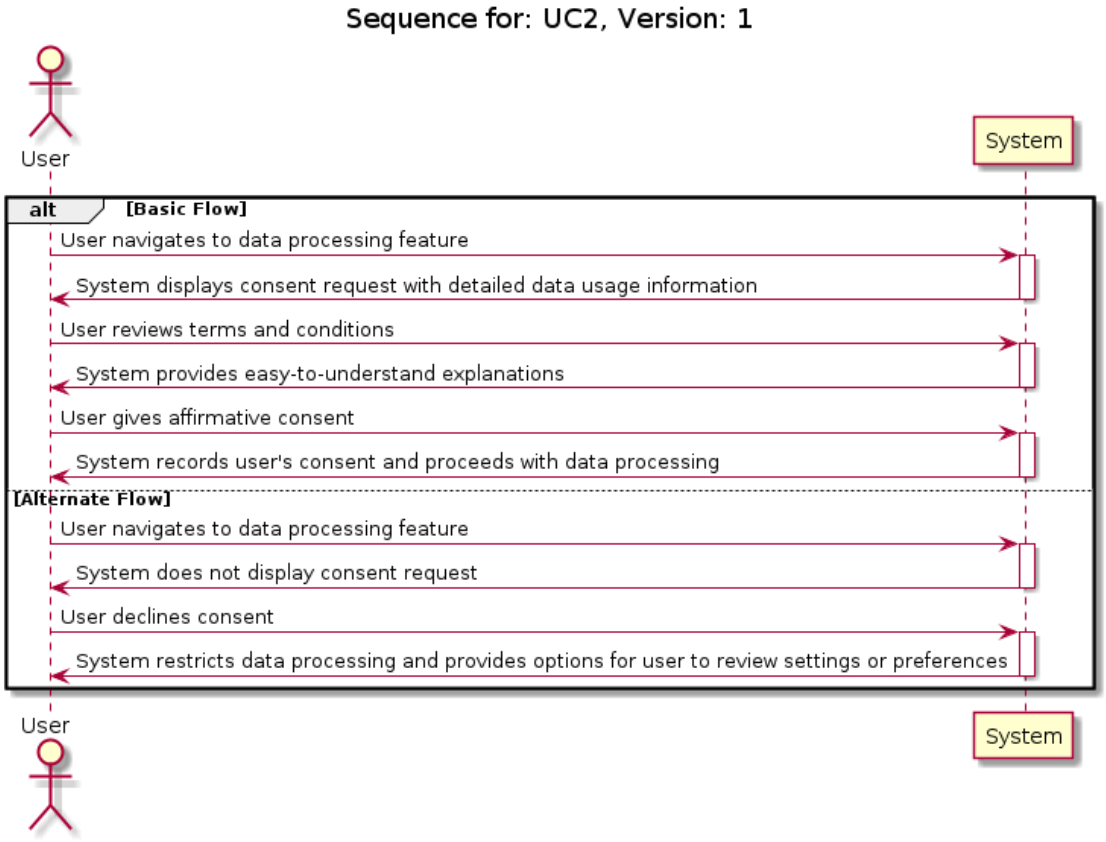

Figure 5.21: Sequence diagram for case 2

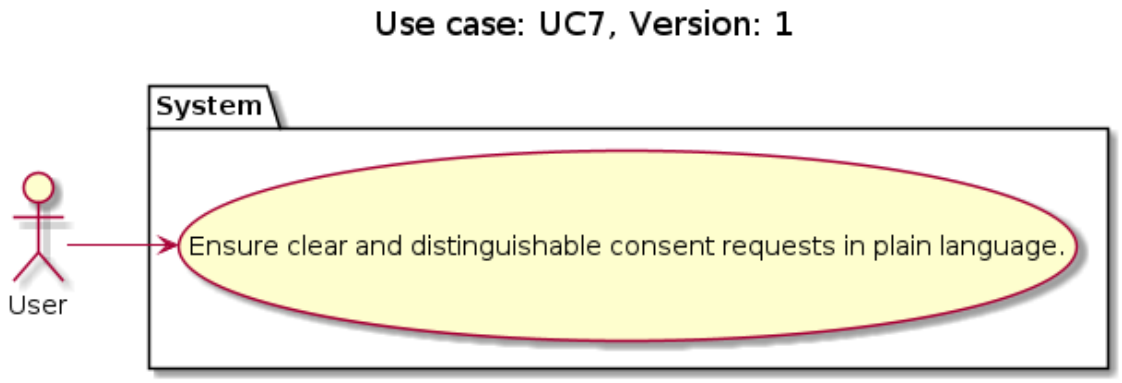

Figure 5.22: Use case diagram for case 7

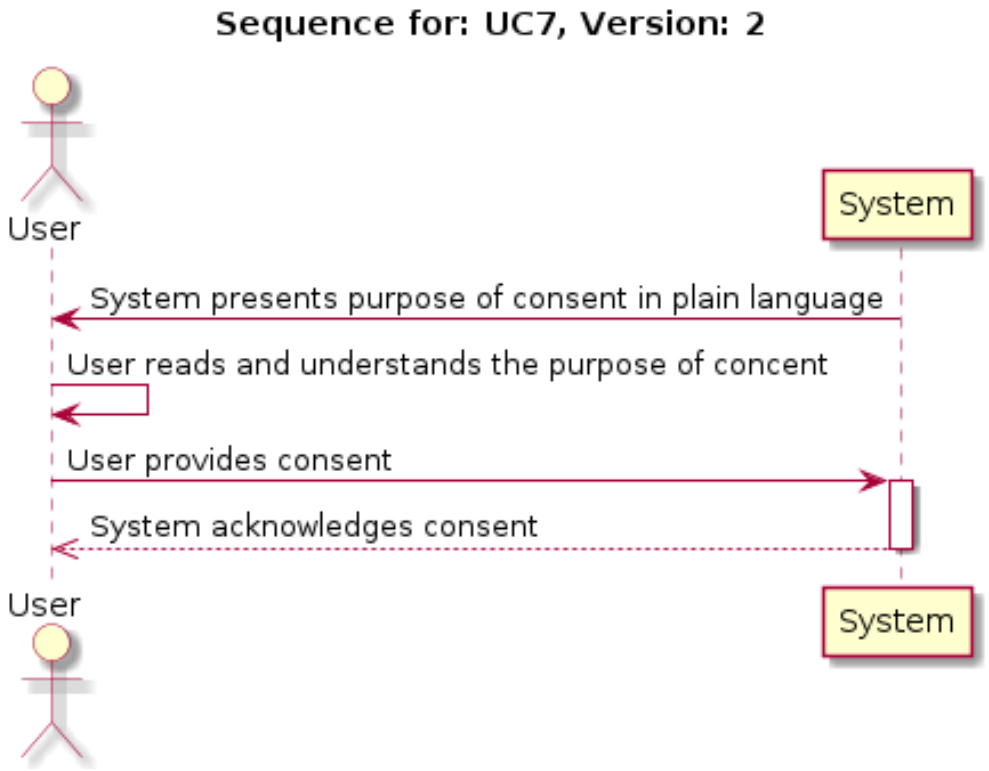

Figure 5.23: Sequence diagram for case 7

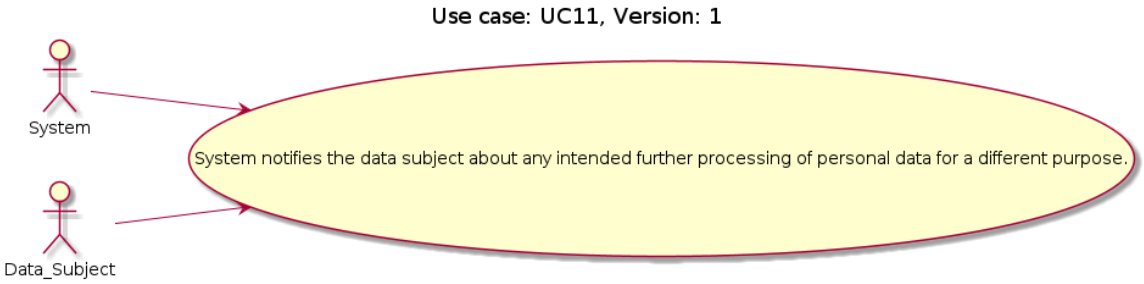

Figure 5.24: Use case diagram for case 11

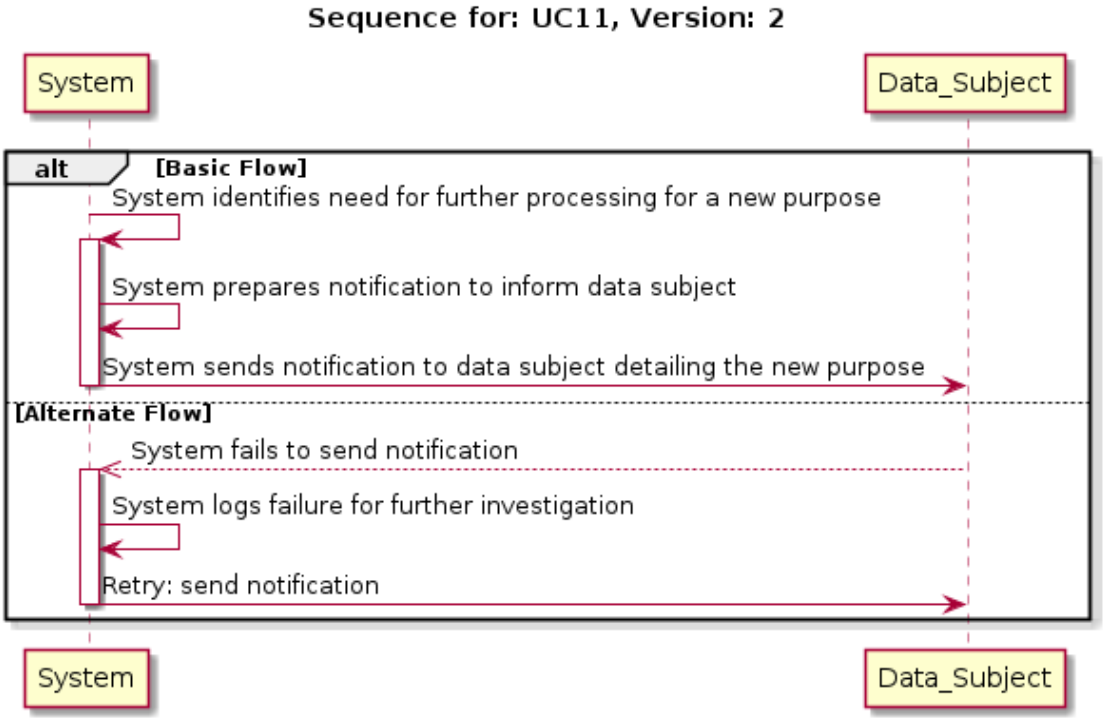

Figure 5.25: Sequence diagram for case 11

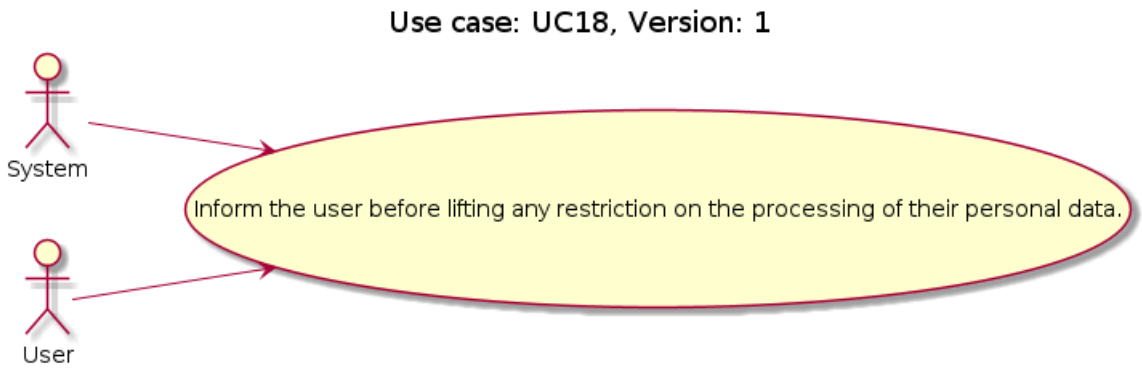

Figure 5.26: Use case diagram for case 18

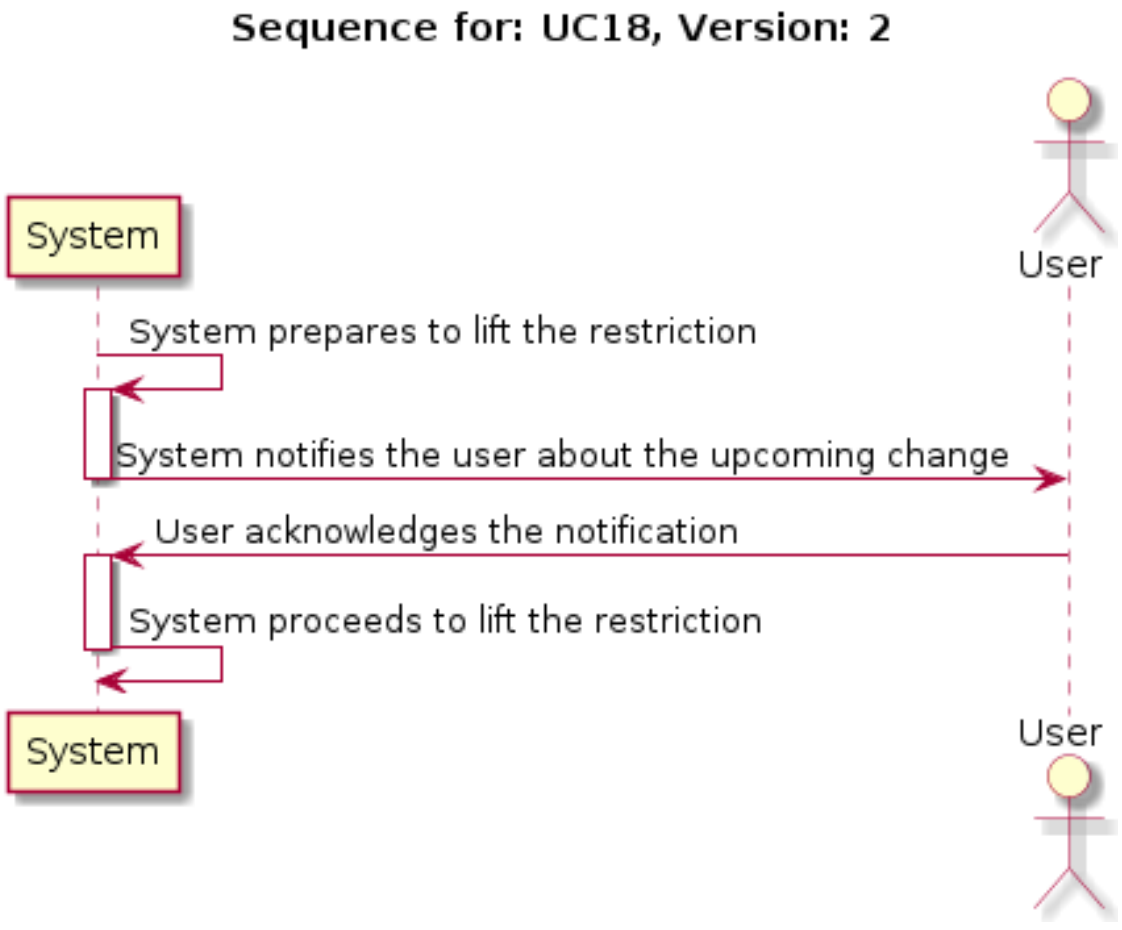

Figure 5.27: Sequence diagram for case 18

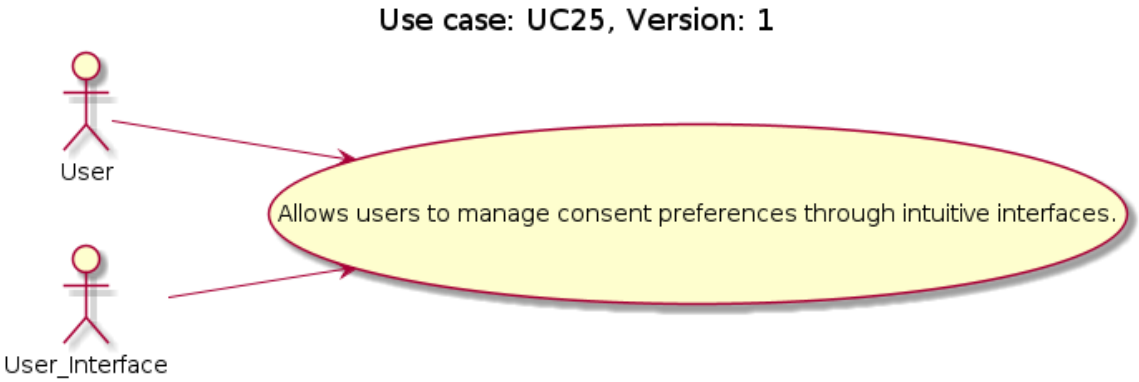

Figure 5.28: Use case diagram for case 25

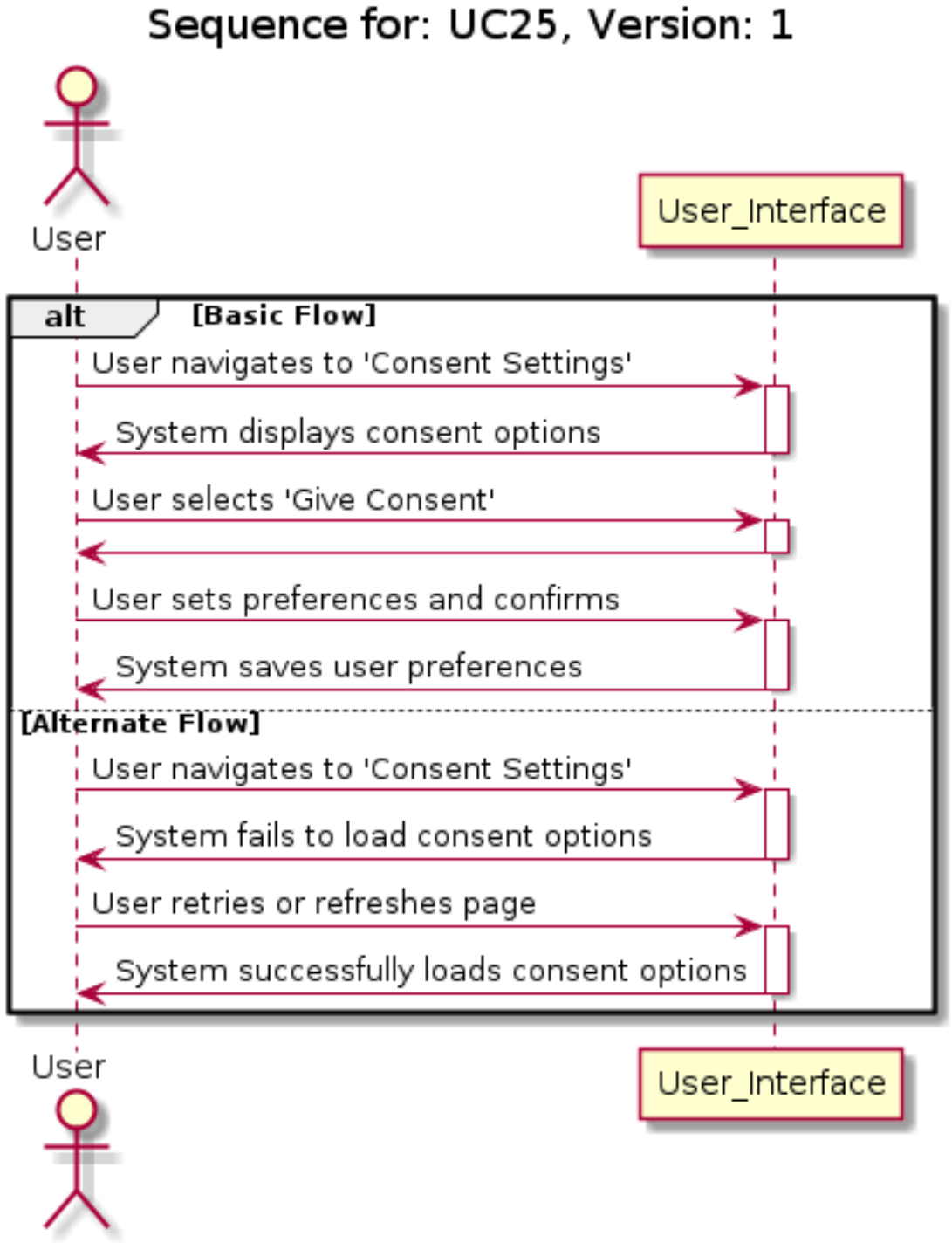

Figure 5.29: Sequence diagram for case 25

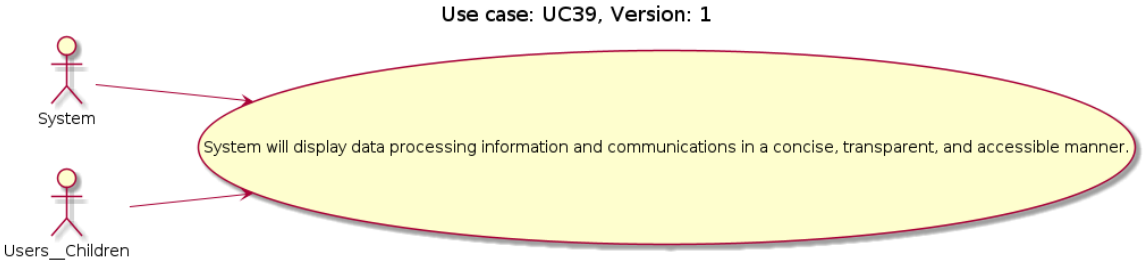

Figure 5.30: Use case diagram for case 39

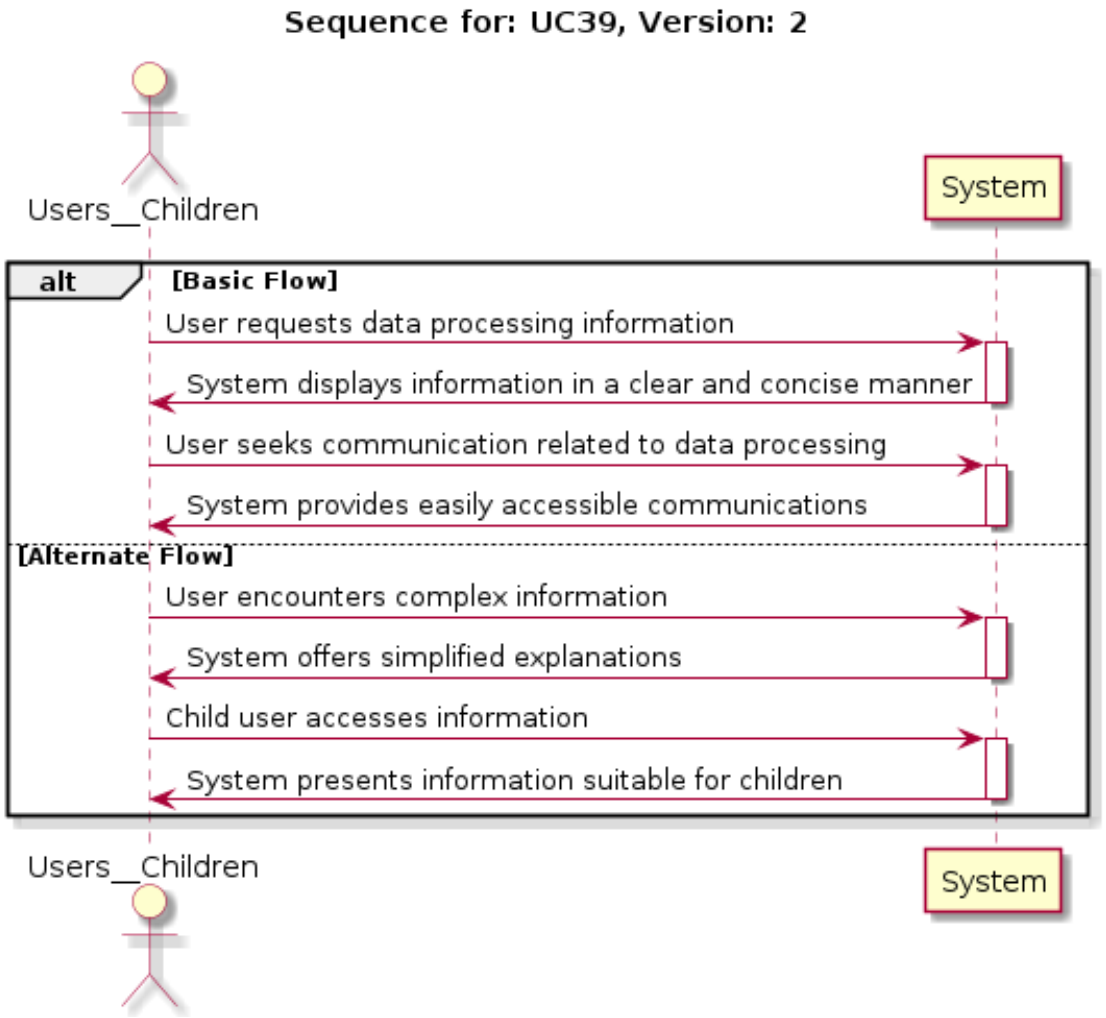

Figure 5.31: Sequence diagram for case 39

5.8 High-level UML diagrams for prototype

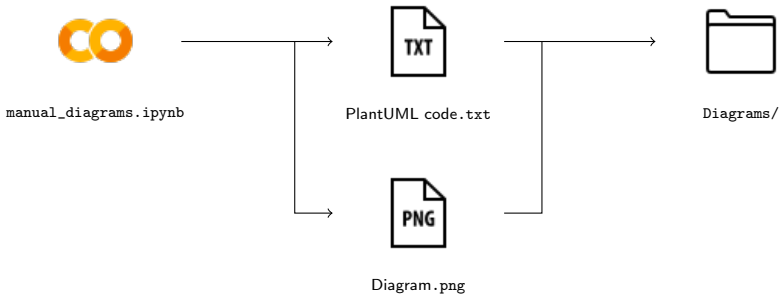

Figure 5.32: Workflow for generating manual diagrams

5.8.1 High-level UML component diagram for prototype

The high-level component diagram show the structure of the prototype and how parts of the backend and frontend work together.

In the frontend, the React Application is the main container for the user interface. The Pages are created to use React components to construct the user interface. React components are interacting with services to fetch or send data. Custom hooks are designed to simplify the interface to UI components by abstracting and managing data fetching or state logic. Services handles external API communication and data storage. The context provides a React context for managing authentication state across the prototype.

The backend is a Node.js Server with Express. The Middleware includes security checks that make sure only allowed users can access certain parts of the app. Controllers handles specific tasks like managing user data or uploads, and they talk directly to the database to get or store information. The Config holds all the settings needed for the server to connect to databases and manage other necessary configurations. The prototype also use Firebase.

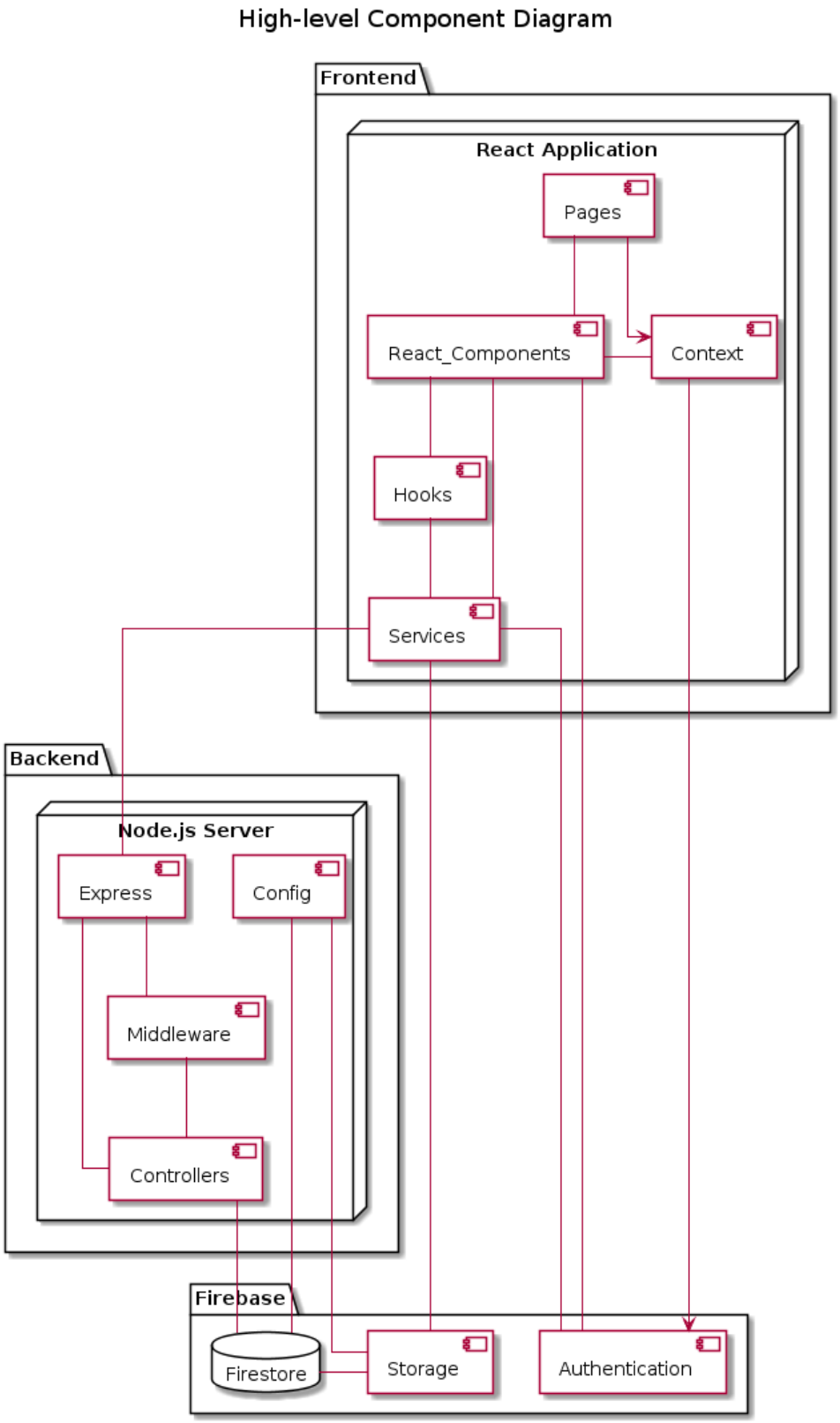

Figure 5.33: UML - High-level component diagram of prototype

5.8.2 High-level UML use case diagram for prototype

The high-level use case diagram shows how users will interact with the prototype. The diagram divides the users into two roles: Data Providers and Data Consumers, each with specific tasks they can perform.

DataProviders can upload images to the prototype and add tags to the image if preferred. They can also perform annotation tasks, where they can answer "yes" or "no" to a question asked about an image in already uploaded to the database. An important part of their interaction with the prototype is giving consent to use their data. Depending on the data they provide, they might need to give explicit consent, ensuring that the system handles their data responsibly.

DataConsumers use the system to search for images tagged in specific ways. They can define their searches by filtering the results, making it easier to find exactly what they need. Additionally, they can view statistical charts that show the distribution of tags and user metadata. For example, users can see which types of user metadata have been involved in annotation tasks on images, identify the most common age group performing annotations on images with specific tags, and more.

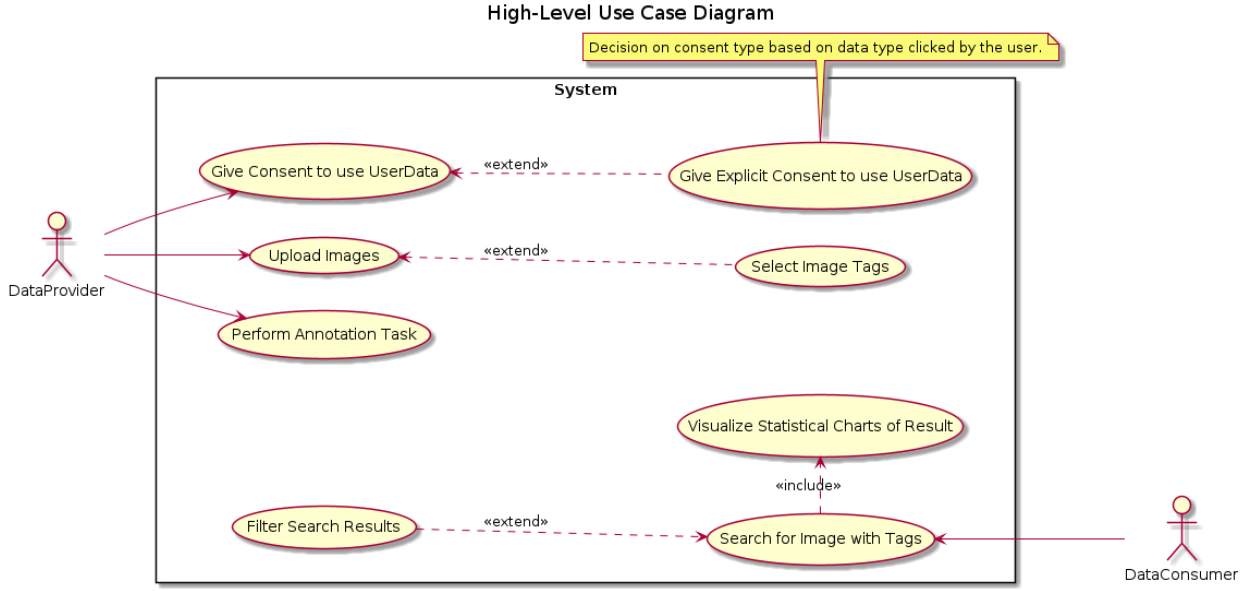

Figure 5.34: UML - High-level use case diagram of prototype

5.9 Mid-level UML diagrams

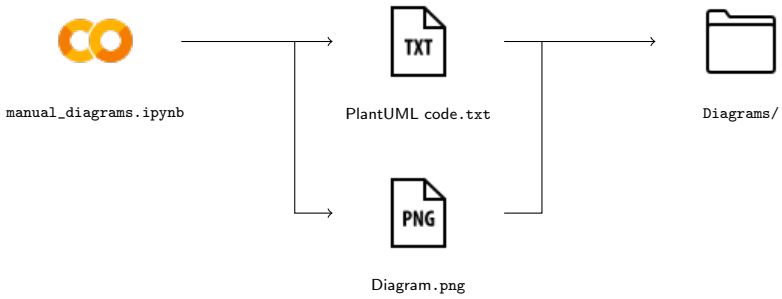

Figure 5.35: Workflow for generating manual diagrams

5.9.1 Mid-level UML component diagram for prototype

The mid-level component diagram shows a more detailed diagram than the high-level component diagram of how the prototype is structured, showing how backend and frontend components work together. Please refer to Appendix O for full diagram.

Frontend

On the frontend, React Components such as BarChartComponent, ChartAccordionItem, NavigationBar, and Footer are going to be created to display the user interface. They are going to be used within different React Router pages like HomePage, Register, Login. The prototype will use custom hooks like useApiData to handle data management, ensuring fetching and management of data. Services like apiService and storageService are going to be created to handle external API communication and data storage operations. Additionally, the Context component with AuthProvider is going to be created to provide a React Context for managing authentication state across the prototype. The Config component containing firebaseConfig will contain configuration settings for Firebase integration.

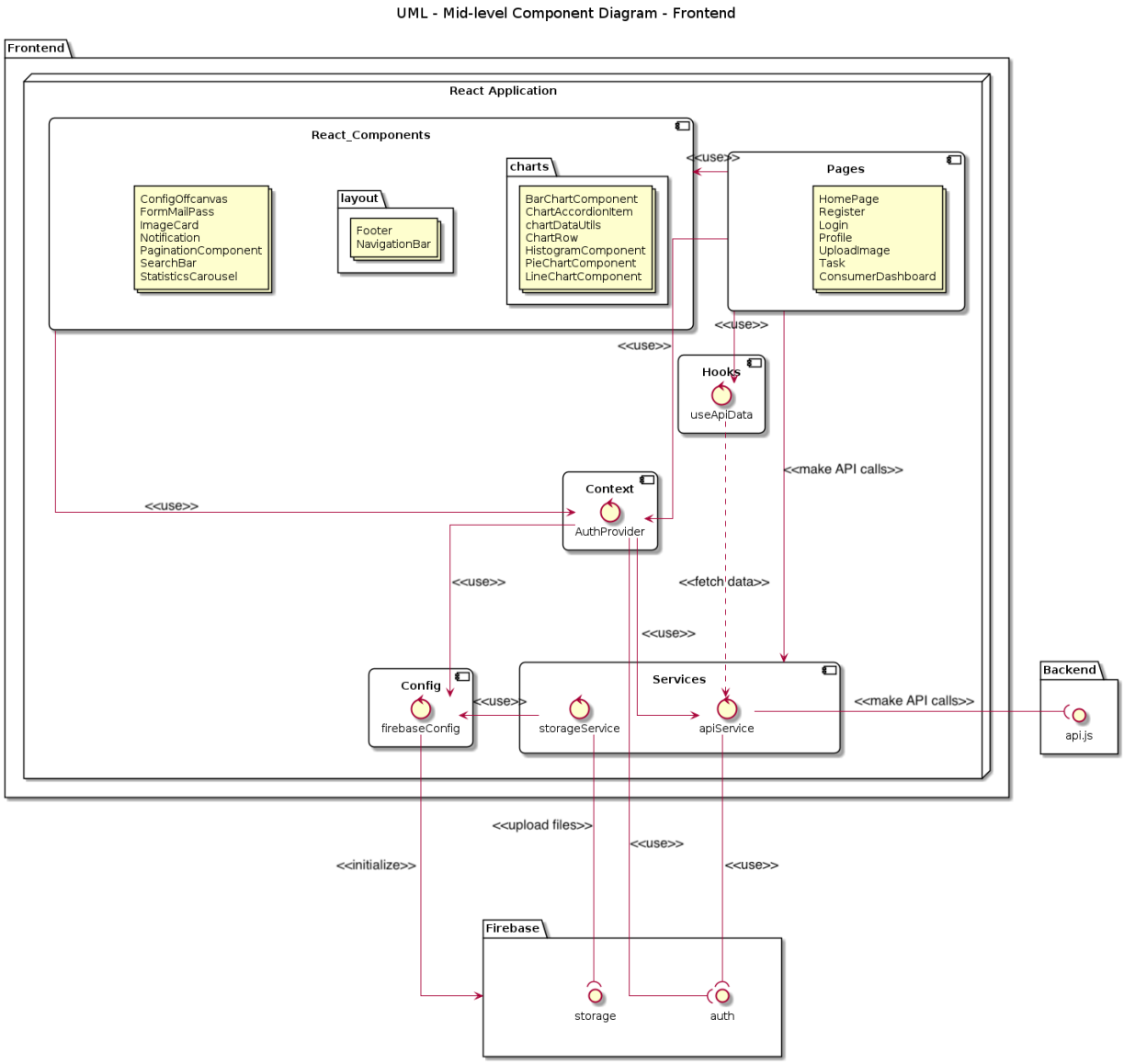

Figure 5.36: UML - Mid-level component diagram of frontend in the prototype

Backend

On the backend, the server-side of the prototype is Express as explained in the high-level diagram. Security is prioritized with the Middleware component verifyToken.js, which validate incoming requests before allowing them access to API endpoints. This component is implemented by api.js to maintain secure routes. The Controllers, including userController.js, taskController.js, and uploadController.js, manages user interactions and task operations. They work with database configurations and API routes to handle and retrieve data. The Config component dbConfig.js manage database connection settings, linking directly with Firestore to set up secure database interactions.

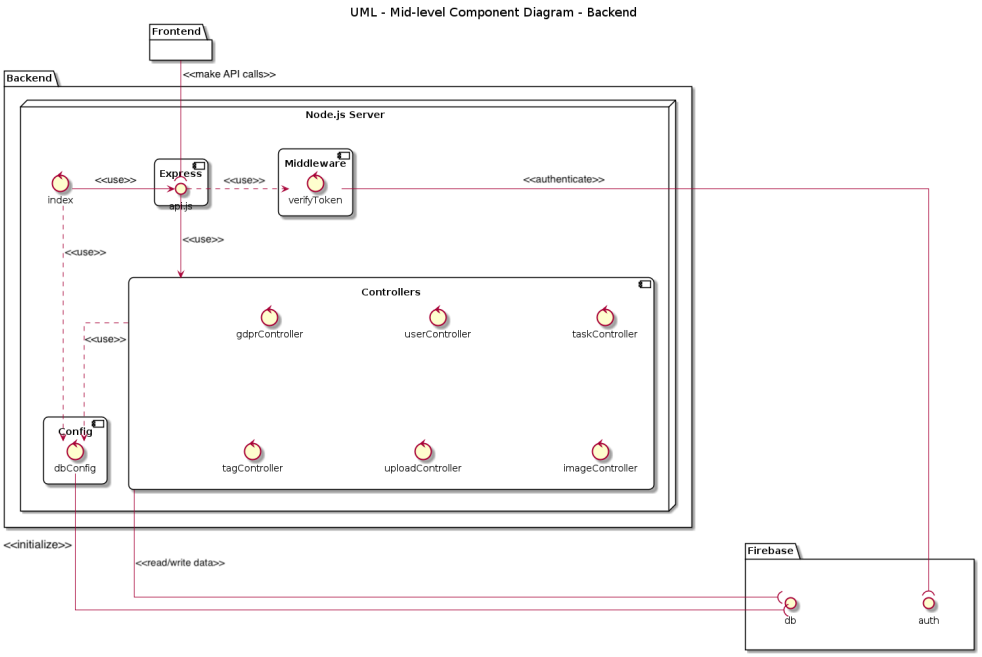

Figure 5.37: UML - Mid-level component diagram of backend in the prototype

Firebase

The Storage component of Firebase manages all file storage for things like user uploads, while the Authentication part takes care of logging users in.

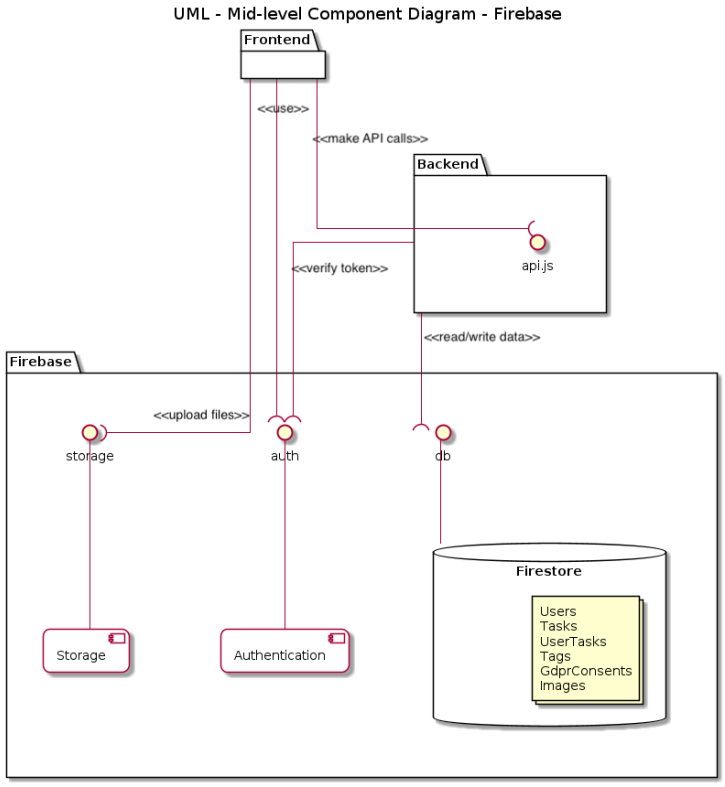

Figure 5.38: UML - Mid-level component diagram of firebase in the prototype

5.9.2 Mid-level UML sequence diagram for Firebase Auth

The sequence diagram shows the authentication token flow between the frontend and Firebase Auth service. The process is initiated at the start of API-request that require token. The **ApiService** requests a token from **Firebase Auth**, then the Firebase Auth issues a token back to the frontend. This token is important because it is attached to API requests to the backend services. The backend uses Firebase Auth to validate the token, ensuring that only requests with valid, attached user tokens are processed. The Firebase rules can allow or deny access to data based on the tokens validity and the permissions it has.

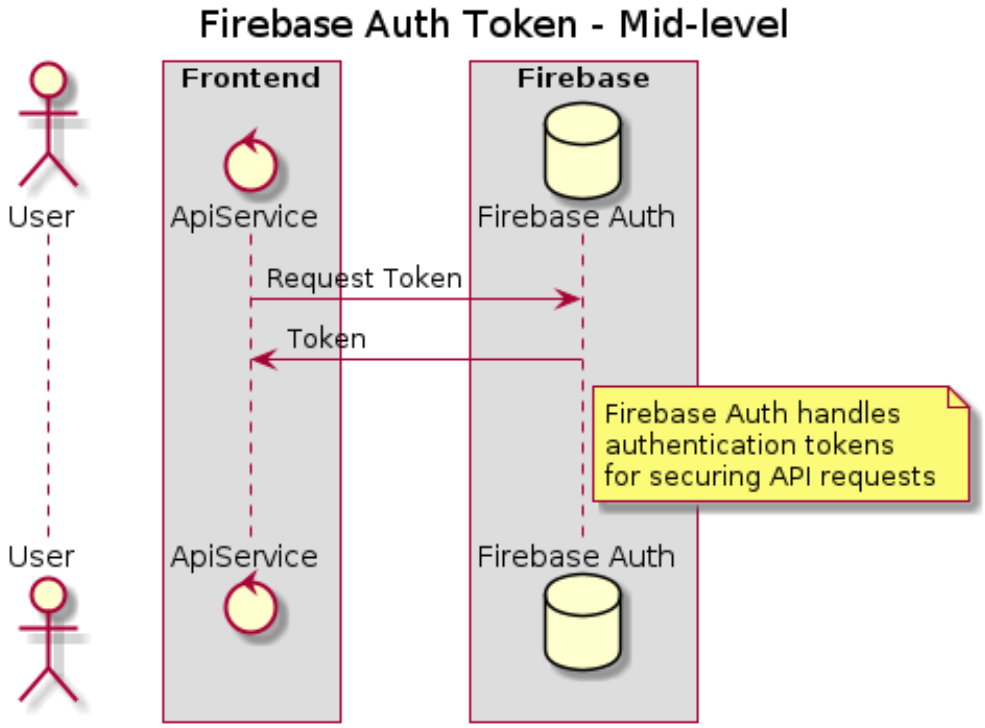

Figure 5.39: UML - Mid-level sequence diagram for Firebase Auth

5.9.3 Mid-level UML sequence diagram for AuthProvider

Initialization
This diagram shows the authentication flow managed by the **AuthProvider**. The sequence starts when the UI initializes the AutProvider. The AuthProvider then sets up a listener which monitor changes in user authentication status. When Firebase Auth detects a user state change, it notifies this to the Auth provider.

User Authentication Change
Additionally, the Firebase Auth notifies the AuthProvider if a user state is changed. If a user is authenticated, the AuthProvider send a GET request on the users details to Firebase Auth. Then Firebase Auth returns the user details back to the Auth provider. Once the AuthProvider has received the user details, it fetches the user roles by querying the user roles through the ApiService to **Firestore**. Then Firestore returns the role details back to the AuthProvider through the ApiService. Then the AuthProvider updates the context on the UI. If the user is not authenticated, the application context is updated to a state indicating no active user.

Firebase Auth Token Handling
Ref to fig. 5.39.

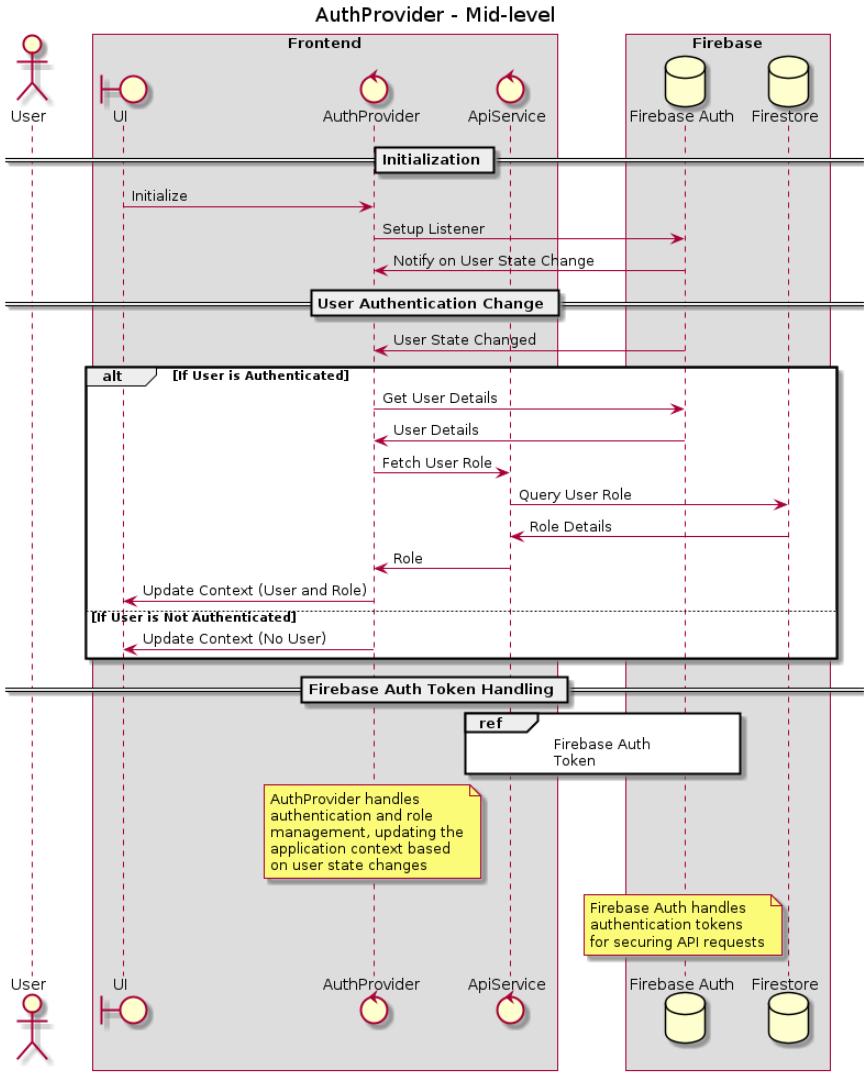

Figure 5.40: UML - Mid-level sequence diagram for AuthProvider

Mid-level UML sequence diagram for task

Fetch New Task Data
When a user goes to the task page, the system starts a Task Processing Cycle. First, the system asks for new tasks by sending a request from the front end to the ApiService. Refer to fig. 5.39 for Firebase Auth Token. The ApiService then sends a GET request to the **backend server**. The backend server gets this request and looks for the task and image data stored in Firebase, then sends these back to the front end through the ApiService.

Perform Task
Once the task and image show up on the users screen, the user can start working on the task. E.g, they might need to answer 'Yes' or 'No' to a question related to the image.

Submit Task Response
After the user finishes the task, their response is sent back to the ApiService. Refer to fig. 5.39 for Firebase Auth Token. The ApiService then sends a POST request to the backend server. The backend server takes this response, saves it in Firebase, and sends a success message back to the front end and the response was saved successfully. This is the end of one loop in the task processing cycle, and the system can either end there or prepare to start the next task

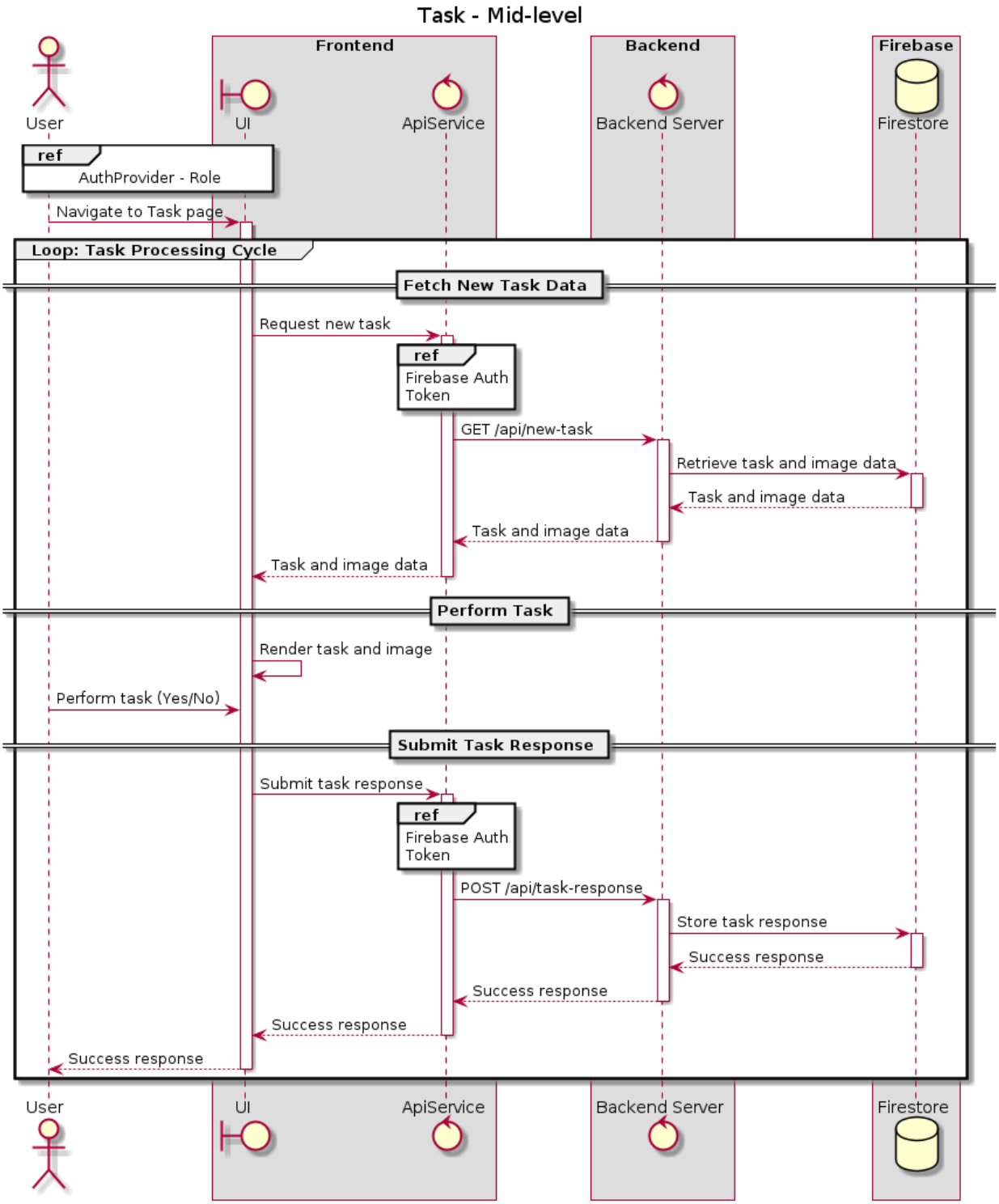

Figure 5.41: UML - Mid-level sequence diagram for task

Mid-level UML sequence diagram for consent

Load page
When a user visits their profile page, the system starts by validating the users session to make sure it is still active. Once confirmed it is confirmed by the AuthProvider, the page loads.

Load consent templates
Next, the system loads GDPR consent templates from the server. Refer to fig. 5.39 for Firebase Auth Token. This involves the frontend making a request to the backend server through a GET request from the ApiService, which fetches these templates from Firestore. The consent templates then gets sent back to frontend through the backend server.

The system also retrieves specific consents that the user has already agreed to or declined. Refer to fig. 5.39 for Firebase Auth Token. This is done by the frontend asking the backend for the users consent details through a GET request from the ApiService, and the backend getting this information from Firestore again.

Once both the templates and the uses existing consents are fetched, they are displayed on the users profile page. Here, the user can see what consents they have agreed to and can make changes if they want to. If a user previously have consented to processing the type of user data, the field will be open for editing. Else the user must consent. E.g, if the field allows, a user can edit their profile information or change their consent choices.

User actions
If a user decides to read more about a GDPR consent, they can click to display further details and choose to accept it directly. If the user data is a type that requires explicit consent the user must input the required text to in order to consent. After reviewing and accepting the changes, the user can submit these updates by clicking a button to save the changes. Refer to fig. 5.39 for Firebase Auth Token. This action sends a request from the frontend of the system to the backend server through a PUT request from the ApiService. The backend server then processes this request and updates the users information and consent preferences in the Firestore database. The user then get a response on the transaction

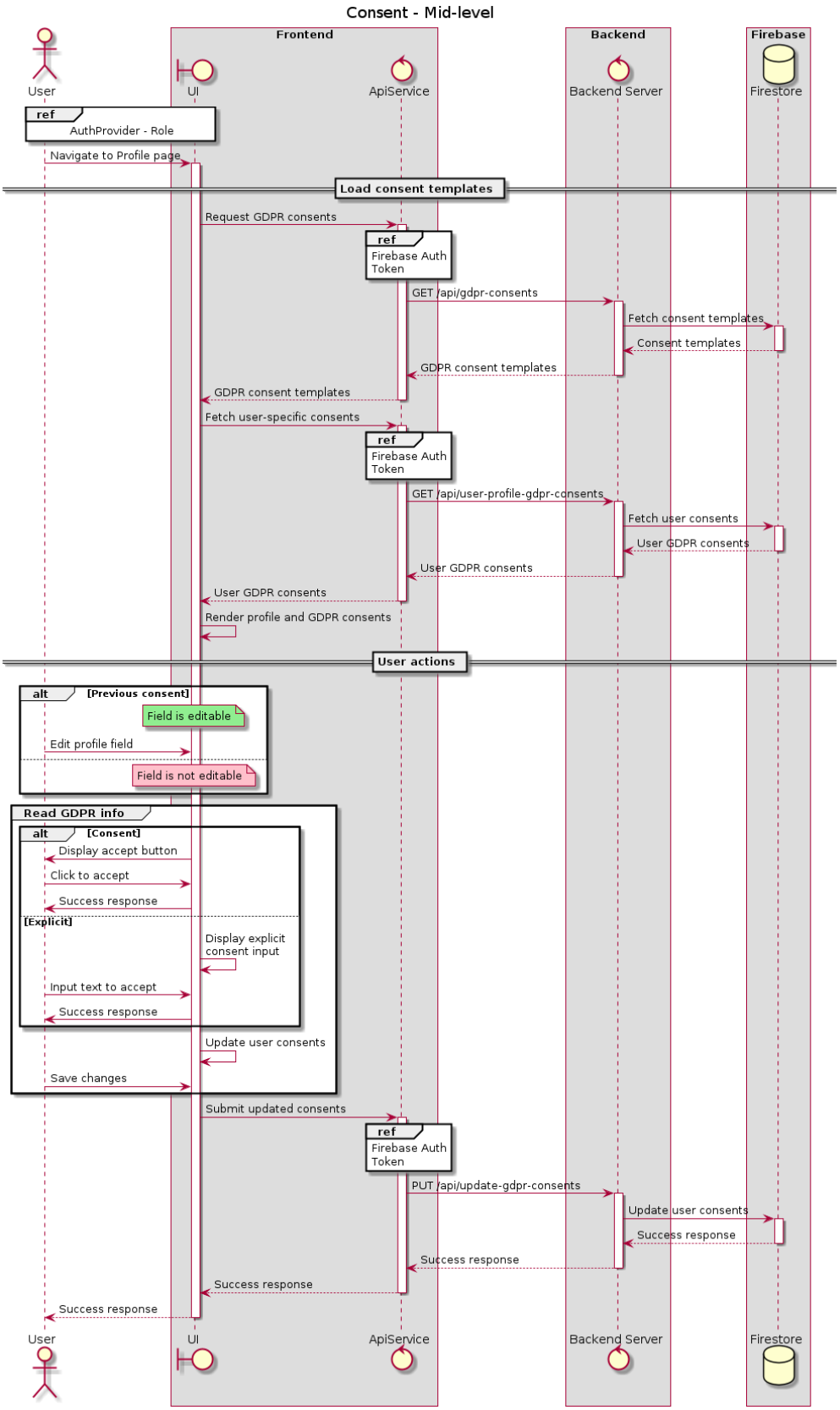

Figure 5.42: UML - Mid-level sequence diagram for consent

Mid-level UML sequence diagram for upload image

Load Upload Image Component
When a user wants to upload an image, they begin by visiting the image upload page. The system automatically loads the components that allow the user to upload images.

Fetch Tags
To help the user tag the image, the system needs to provide existing tags. The frontend of the application requests these tags by sending a GET request to the backend server. Refer to fig. 5.39 for Firebase Auth Token. The backend server handles this request by retrieving the tag data from Firestore, and then sends this data back to the frontend.

User Uploads Image
When the tags is displayed, the user selects an image file and assigns the relevant tags from the fetched list. Additionally, the system retrieves the current users data to link with the image upload through the AuthProvider.

Upload Image to Storage
Once the user has selected an image and the preferred tags they click the upload button. This action triggers the frontend to send the image file containing the image and the user id to the **storage service**. Refer to fig. 5.39 for Firebase Auth Token. The storage service then sends the image file to **storage** on the firebase database. Storage on the firebase database then returns the image URL back to the frontend through the storage service.

Submit Image Metadata
Additionally, frontend takes the image metadata, which includes the download image URL, user id, tags and the timestamp, and sends it to Firestore through a POST request from the ApiService to the backend server. Refer to fig. 5.39 for Firebase Auth Token. Once the image metadata is stored in Firestore, it sends a success response back to the frontend through the backend server and ApiService.

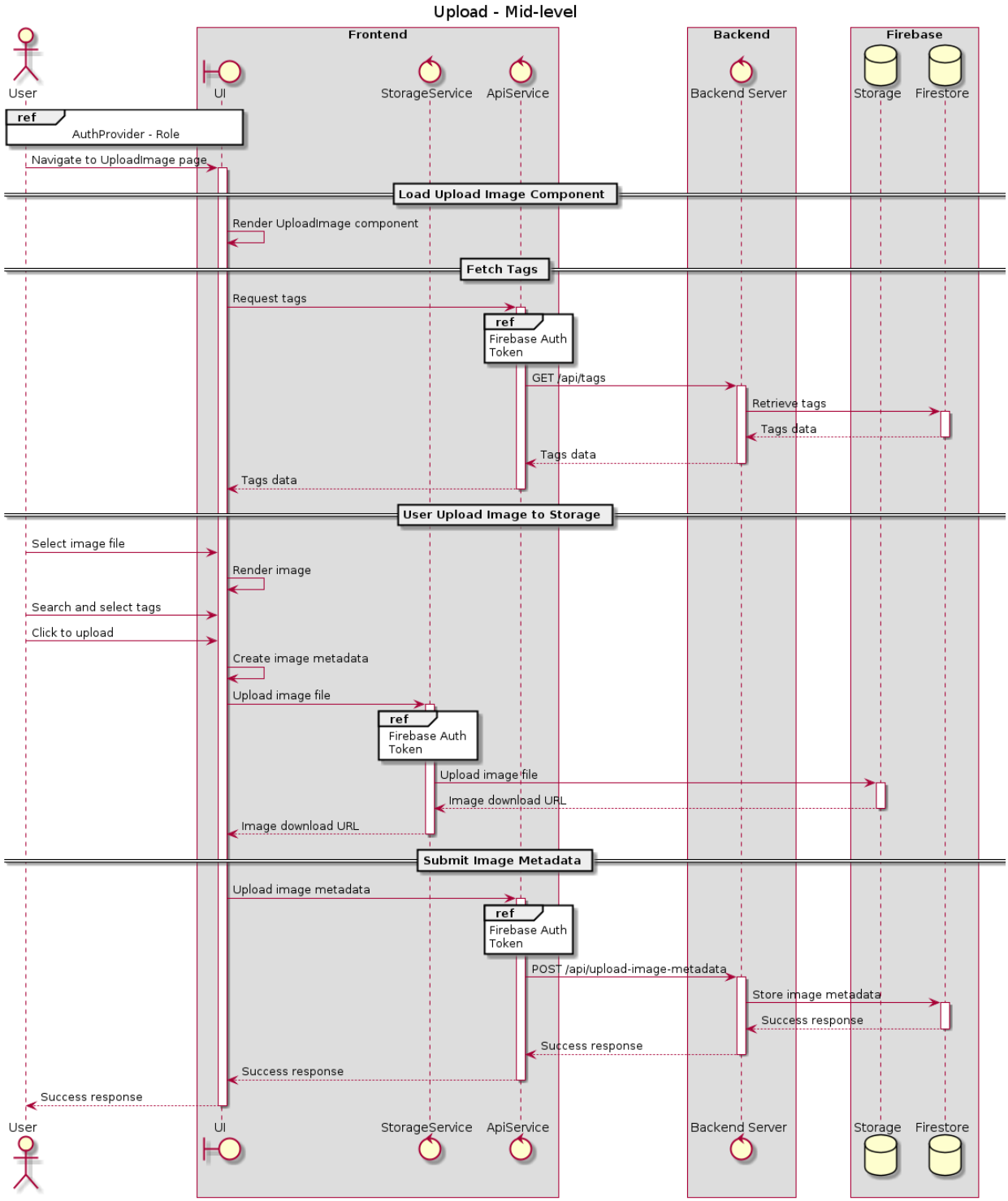

Figure 5.43: UML - Mid-level sequence diagram for upload

5.9.4 Mid-level UML diagram for search and filtering in the prototype

Load Page
Ref. fig. 5.40 for the reference in the diagram. The user starts by navigating to the consumer dashboard. Then the UI renders tabs for the image and search page.

Search
When the users has navigated to the consumer dashboard and the tabs are rendered, the user searches and select tags and clicks search button. Then the UI starts the search by requesting the images by the tag through **useApiData** to ApiService. Refer to fig. 5.39 for the reference for Firebase Auth. Then the ApiService sends a POST request to the backend server to retrieve the images by tags from Firestore. Then Firestore returns the images data. For each image retrieved the backend server fetches its performed tasks. For each task of each image the backend server aggregates and anonymize the user data.

The images and its correlating anonymized user data is returned to useApiData through ApiService. UseApiData then utilizies **chartDataUtils** to prepare the chart image data for chart friendly structures. Additionally, the chartDataUtils prepares the chart user data by grouping the data by tags and further categorizing it by user attributes. Once these chart data is prepared the useApiData returns the images to the UI. When the images is returned, the UI renders the images and filters the search result if there is any filter applied.

Filter
After the user sets the filters they want to use, the frontend handles this change by automatically updating what the user sees. It re-renders the search results to show only images that meet the new filter criteria. The frontend utilize useApiData and chartDataUtils with the new filter information in order to prepare filter chart data.

Charts
When the user clicks on the "Statistics" tab, the UI renders the button for carousel or row view. Then the UI maps the user data to useApiData and the useApiData returns the prepared chart data. Once the UI has received the prepared chart data, it renders the tab for Carousel view and the corresponding charts. Once the user switches to row view the UI once again maps the user data to useApiData and gets the prepared chart data in return. The UI then renders the tab for row view and the corresponding charts.

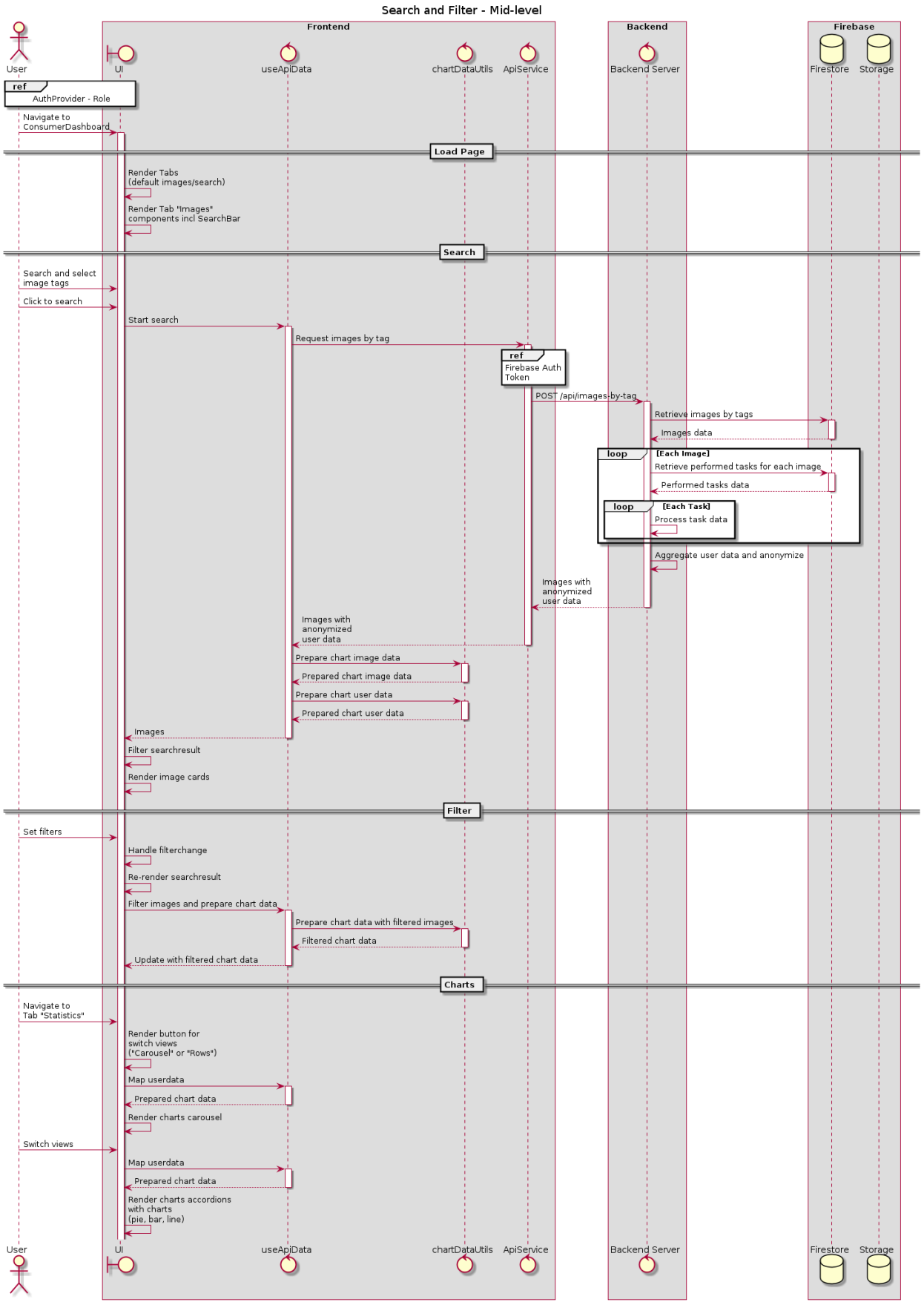

Figure 5.44: UML - Mid-level sequence diagram for upload

5.10 Database

"You may notice that documents look a lot like JSON. In fact, they basically are. There are some differences (for example, documents support extra data types and are limited in size to 1 MB), but in general, you can treat documents as lightweight JSON records" [149].

The Firestore database for the prototype is organized into several collections such as gdprConsents, users, tasks, tags, images, and userTasks. Each collection contain various attributes to store relevant data.

The gdprConsents collection contains templates for the consents. The templates contain information for providing required information to users in order to adhere to GDPR regulations, such as the consentName, options provided, purpose for processing and whether explicit consent is needed. It includes arrays of legal bases for processing personal data and the rights of the users.

User information is stored in the users collection. This contains details like email, role, and a mapping of GDPR consents. Each consent contain consentName, when the consent was accepted and the data of the consent (e.g. age, gender).

The tasks collection contains templates with identifiers and task description for different tasks that users can perform. This is designed as a foundation for future implementation of various tasks. Similarly, the tags collection are templates for keeping a dictionary of tags in order to prevent duplicates, misspelling, etc. The documents contain tagId and tagName.

Images are managed in the images collection, which keep downloadURL, array of tags and the userId of the data providers who provide or upload the image. Additionally, each image includes a sub collection for each task that is performed on or with the image. Each document in the performedTasks sub collection contain a tag, a reference to the userTask collection and a userDataSnapshot. The userDataSnapshot includes data from users who have given their consent and performed a specific task, capturing information from the time the task was carried out.

The userTasks collection links users, tasks, and images. It records which user performed what task on which image, including attributes like the userId, taskId, imageId, task question, the task answer and timestamps. In order to facilitate efficient data retrieval the userTasks collection serve as a relational table that links users to tasks they perform on specific images. Each entry in the collection represents a unique instance of a task performed by a user on a image.

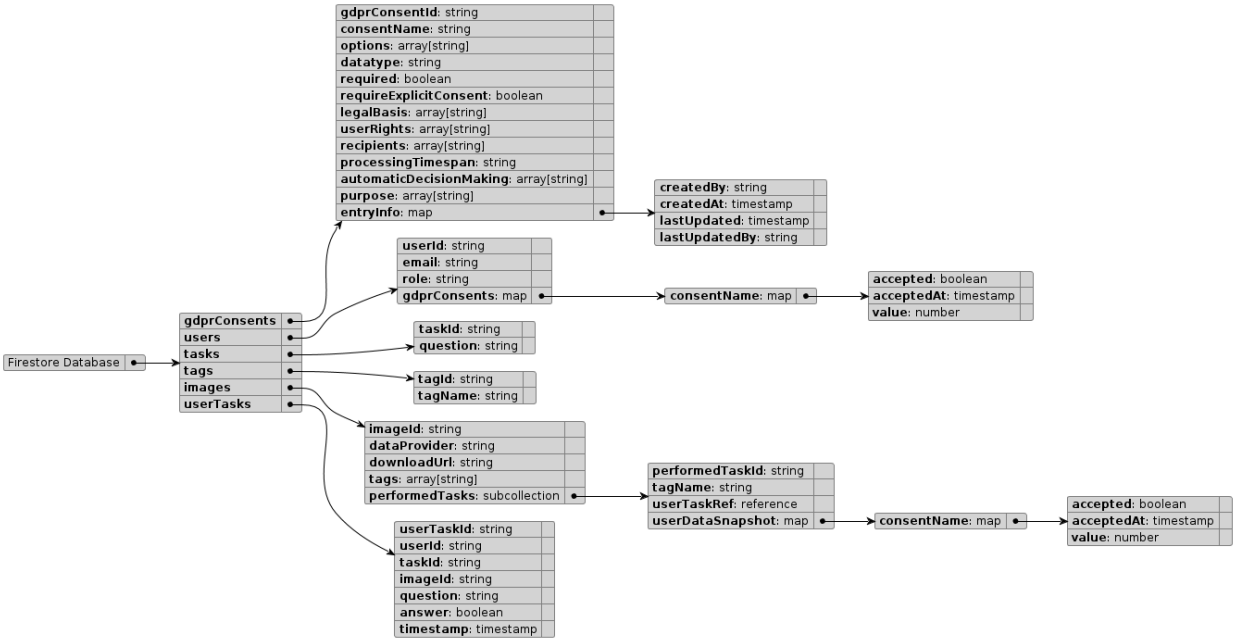

Figure 5.45: Database structure for prototype

5.11 Prototype

5.11.1 Consent

The snapshot in fig. 5.46 shows the GDPR consent page in the user profile section of the prototype. Here, users are prompted to accept or reject the use of their personal information, such as including their age in statistical analyses to ensure diversity in dataset annotations. This consent page is implemented in accordance with use cases UC1, UC2, UC6, UC7, UC25, UC38, UC39, UC59, and UC60 (Appendix B).

The GDPR consent page is created so users can easily manage their GDPR preferences, giving users information about how their data will be used. By requiring users to accept each data usage condition, the prototype ensures that all personal information is handled responsibly and ethically. This prevents any personal data being used without consent.

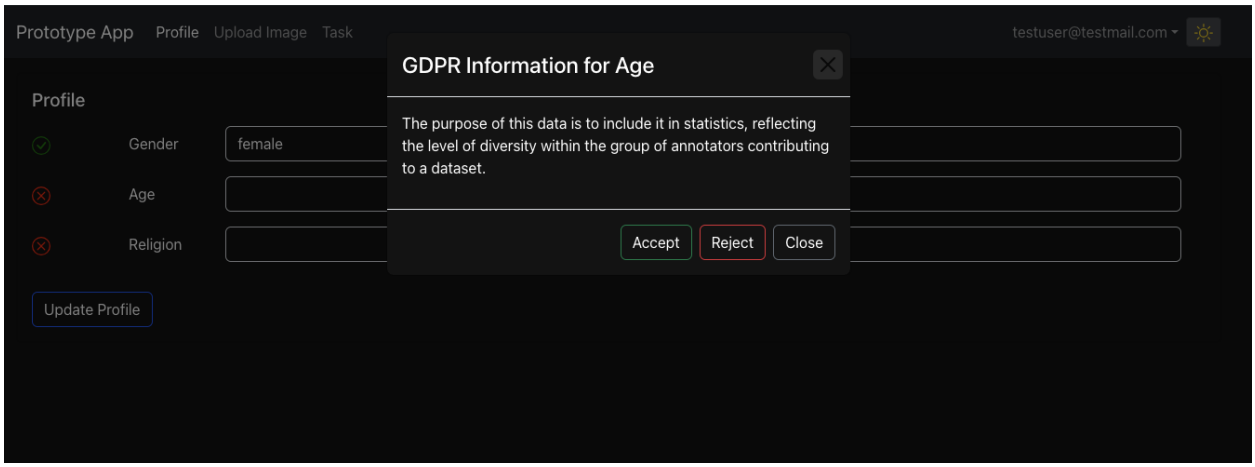

Figure 5.46: Snapshot of GDPR consent page

5.11.2 Explicit consent

The snapshot in fig. 5.47 shows the GDPR explicit consent page used in the prototype for sensitive data categories, such as religion shown in the figure. Users must type the suggested word before accepting to the GDPR consent for the use of their data. Requiring users to manually typing the suggested words, like "Religion", "Gender" or other words, adds another layer of involvement from the user to the consent process. This is required by GDPR when explicit data are going to be used. This module is implemented in accordance with use cases UC3, UC6, UC7, UC38, UC39, UC59, UC60. (Appendix B).

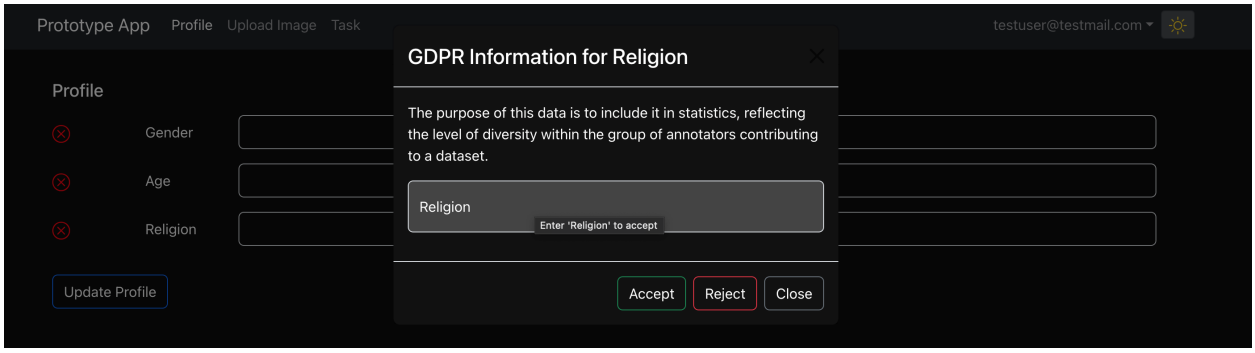

Figure 5.47: Snapshot of GDPR explicit consent page

5.11.3 Upload

The snapshot shown in fig. 5.48 shows the "Upload Image" page of the prototype app, where users can upload their images. When uploading an image, the users are able to select tags from a suggested list added to the database by us. These are shown in the dropdown list. There is no functionality where users can create their own tags and add them to the image in the prototype, this is later discussed in the discussion section. The "upload image" functionality is implemented in accordance with use case UC28 (Appendix B).

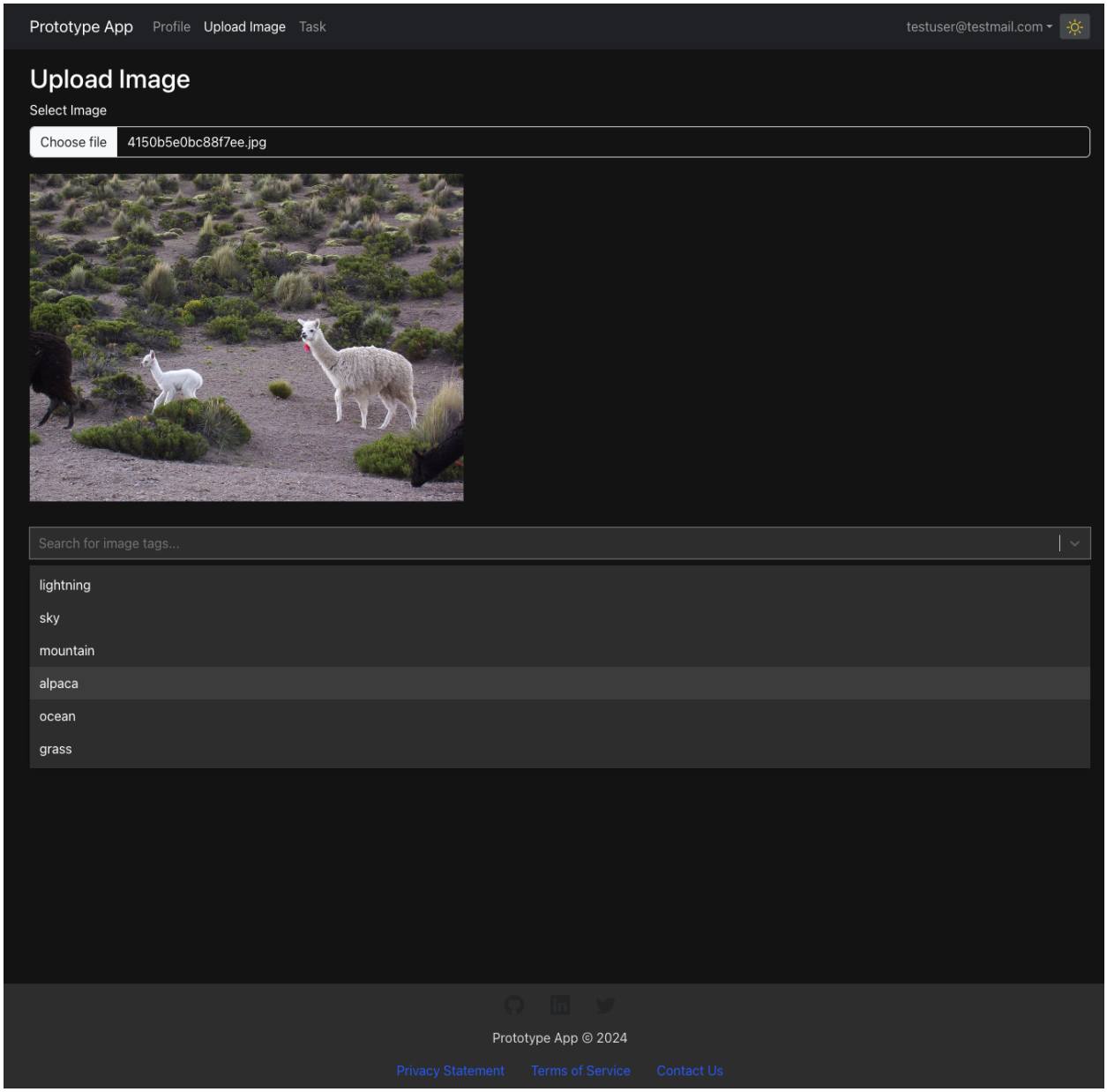

Figure 5.48: Snapshot of page where users upload images

5.11.4 Task

The snapshot in fig. 5.49 shows the task page of the prototype app, where users engage in annotating images based on different images and questions. In this example, the user is presented with an image of an alpaca and asked to determine whether the image contains grass. When a user answers the question, their answer contributes to the annotation of the image, linking their metadata, such as age, gender, or any other relevant user information to the task, answer and image. The task functionality is implemented in accordance with use cases UC29, UC31 (Appendix B). UC29, users are not able to tag the images displayed during a task.

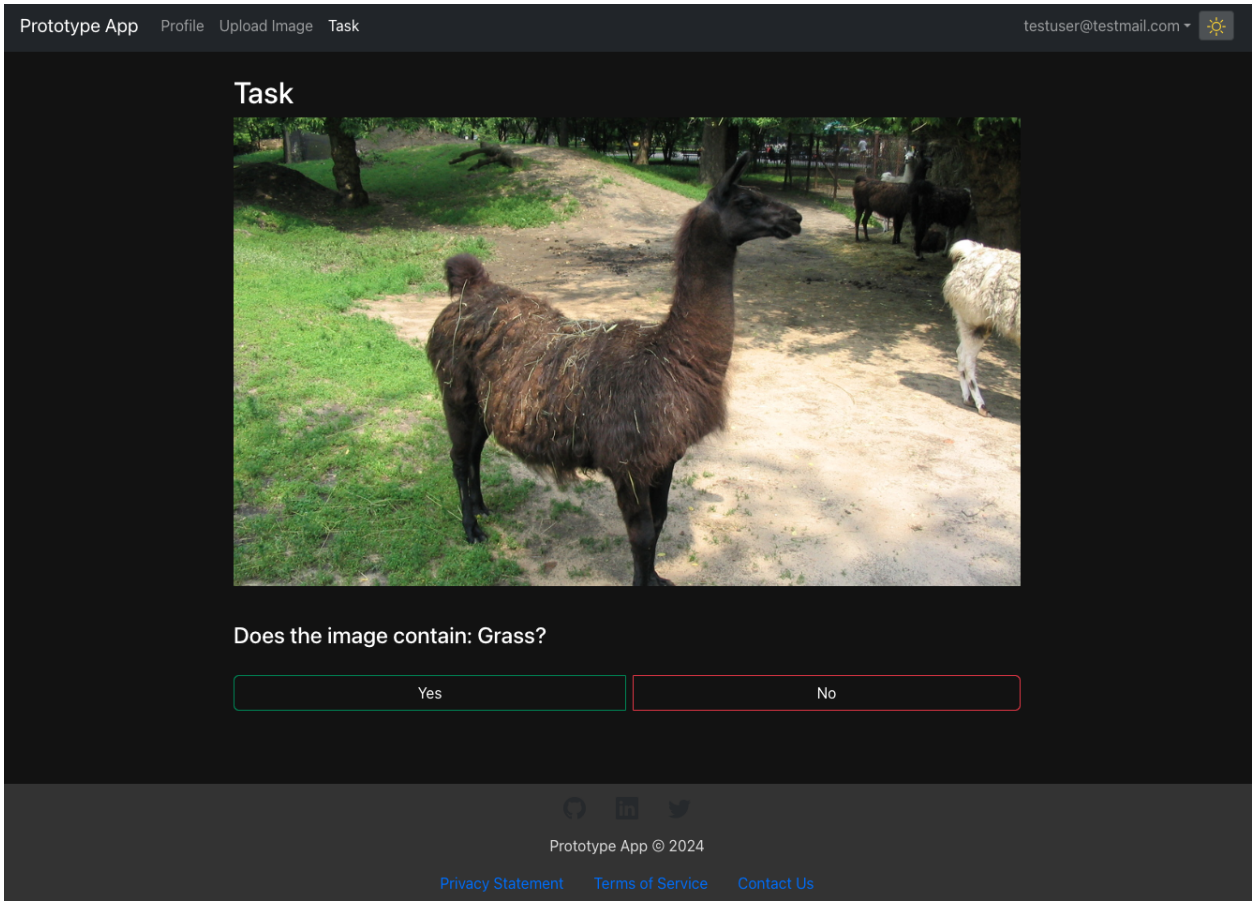

Figure 5.49: Snapshot of task dashboard

5.11.5 Dashboard

The snapshot in fig. 5.50 shows the image search functionality of the prototype dashboard. Users can search for images by entering specific tags in the search field. The system displays images that contain all the tags specified, allowing for searches through the use of multiple tags. For better user convenience, the "auto search" feature can be enabled, which automatically updates search results each time a tag is added or removed. Another feature is the "Close Menu on Select" option, which is designed to control the behavior of the dropdown menus. When enabled or disabled, it causes the list of tag suggestions to close or remain open automatically after a tag is selected.

Additionally, when users hover over an image, the tags associated with that image are displayed, and clicking on any tag will add it to the search field if it is not already included. The upvote, include, and exclude buttons are inactive. This dashboard is implemented in accordance with use case UC34 (Appendix B), except "customizing and creating dataset" is not implemented.

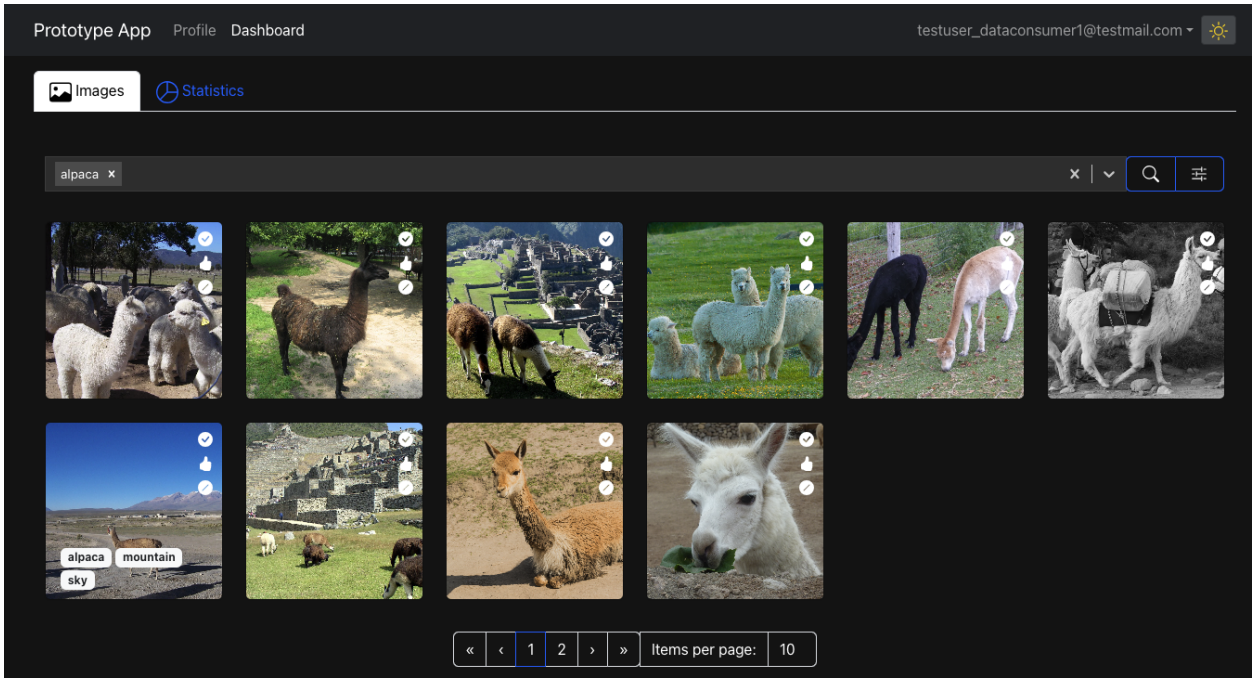

Figure 5.50: Snapshot of consumer dashboard

5.11.6 Filtering

The snapshot in fig. 5.51 shows the filtering functionalities of the prototype dashboard, enabling users to define image searches based on specific annotator metadata such as gender, age, and religion. The filtering interface allows users to select criteria that correspond to the characteristics of the annotators that has been a part in the image annotation process. E.g, a user can filter images annotated by individuals of a certain age range, gender, or religion. The filtering functionality is implemented in accordance with use cases UC35, UC36 (Appendix B). In UC36 the "customizing dataset" is not implemented, but the filtering functionality has made arrangements for it.

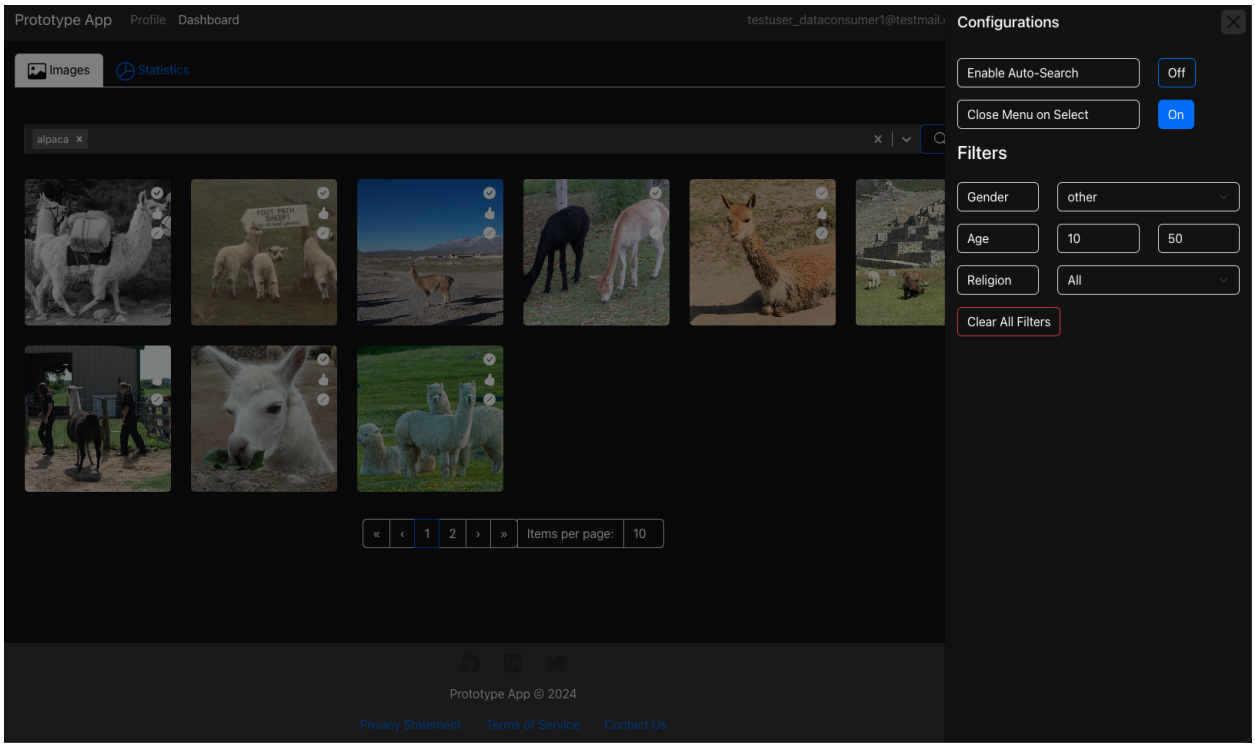

Figure 5.51: Snapshot of filtering dashboard

5.11.7 Statistics

The snapshot shown in fig. 5.52 shows the Statistics dashboard of the prototype app, where users can view statistics on image tags based on the images from the search results. The dashboard shows a pie chart and a corresponding bar graph, both illustrating the distribution of tags across the images.

The pie chart shows how each tag is related to the images, making it easy for users to see which tags, like 'alpaca,' 'sky,' 'grass,' and 'mountain,' are most common. The bar graph shows the exact number of times each tag has been used. The statistics dashboard is implemented in accordance with use case UC37 (Appendix B).

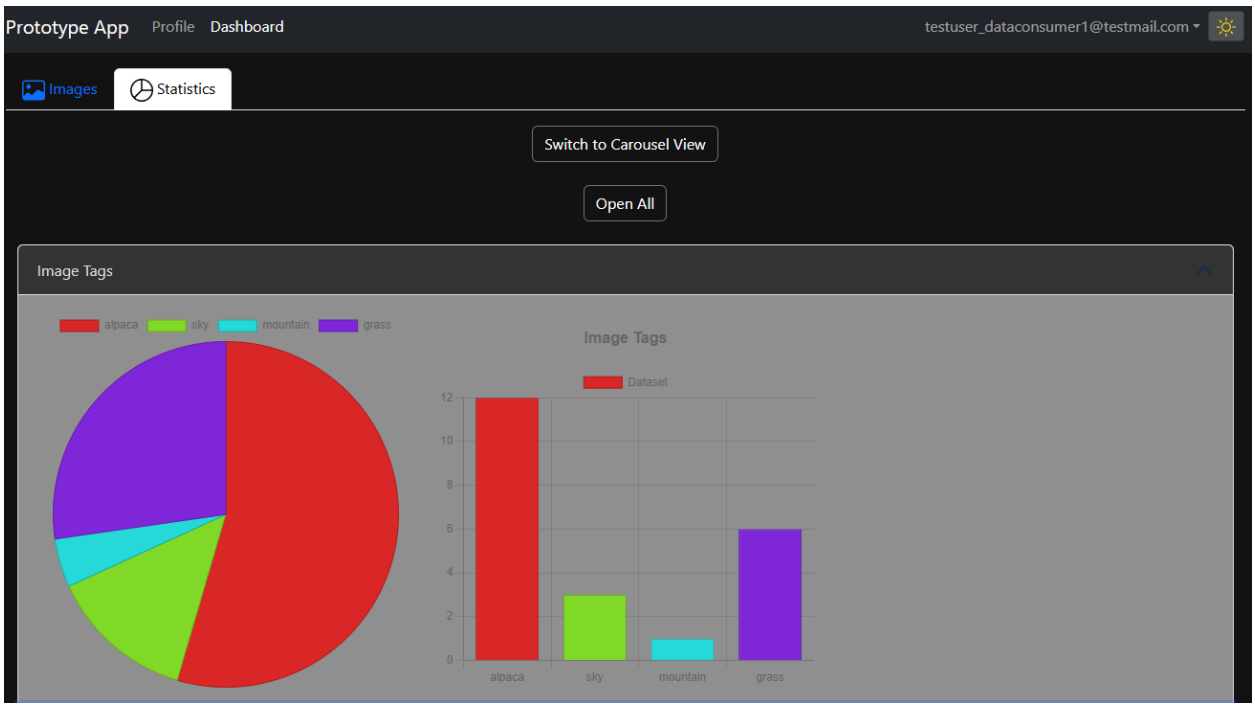

Figure 5.52: Snapshot of statistics dashboard of all tags

The snapshot shown in fig. 5.53 shows statistics related to the annotation tasks for specific tags, in the figure it shows for the tags 'Alpaca' and 'Sky'. Each tag has a pie chart showing the gender distribution and a bar chart showing the age distribution of the annotators linked to that tag. For example, the bar chart for the 'Alpaca' tag suggests that a significant number of annotations are from people in their 50s, with fewer annotators in their 20s and 40s, while the 'Sky' tag is mainly annotated by users in their 50s, with a lower number of annotators in their 40s. The statistics dashboard is implemented in accordance with use case UC37 (Appendix B).

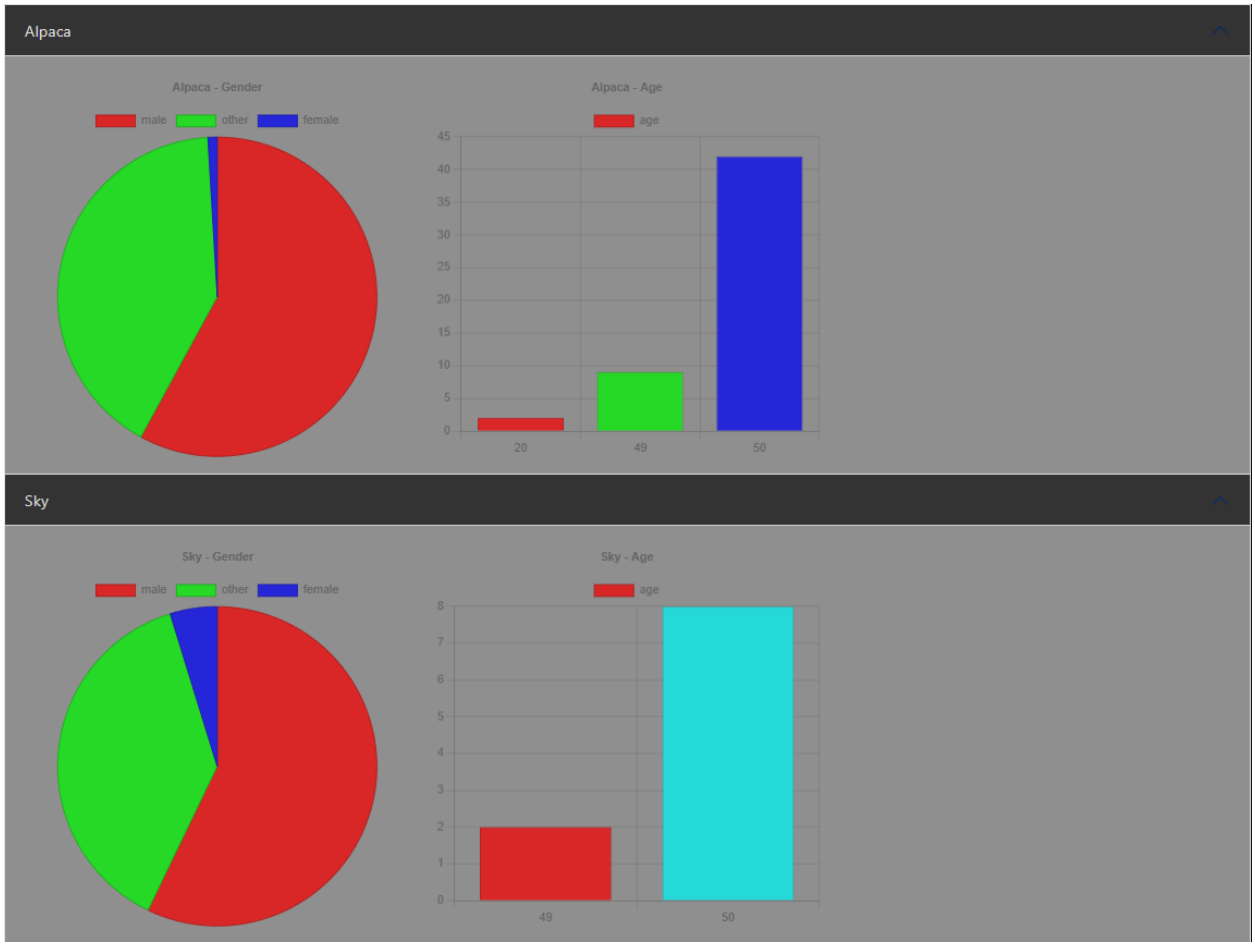

Figure 5.53: Snapshot of statistics dashboard of specific tags

The snapshot shown in fig. 5.54 is just a different view of the same pie and bar charts shown in fig. 5.52 and fig. 5.52.

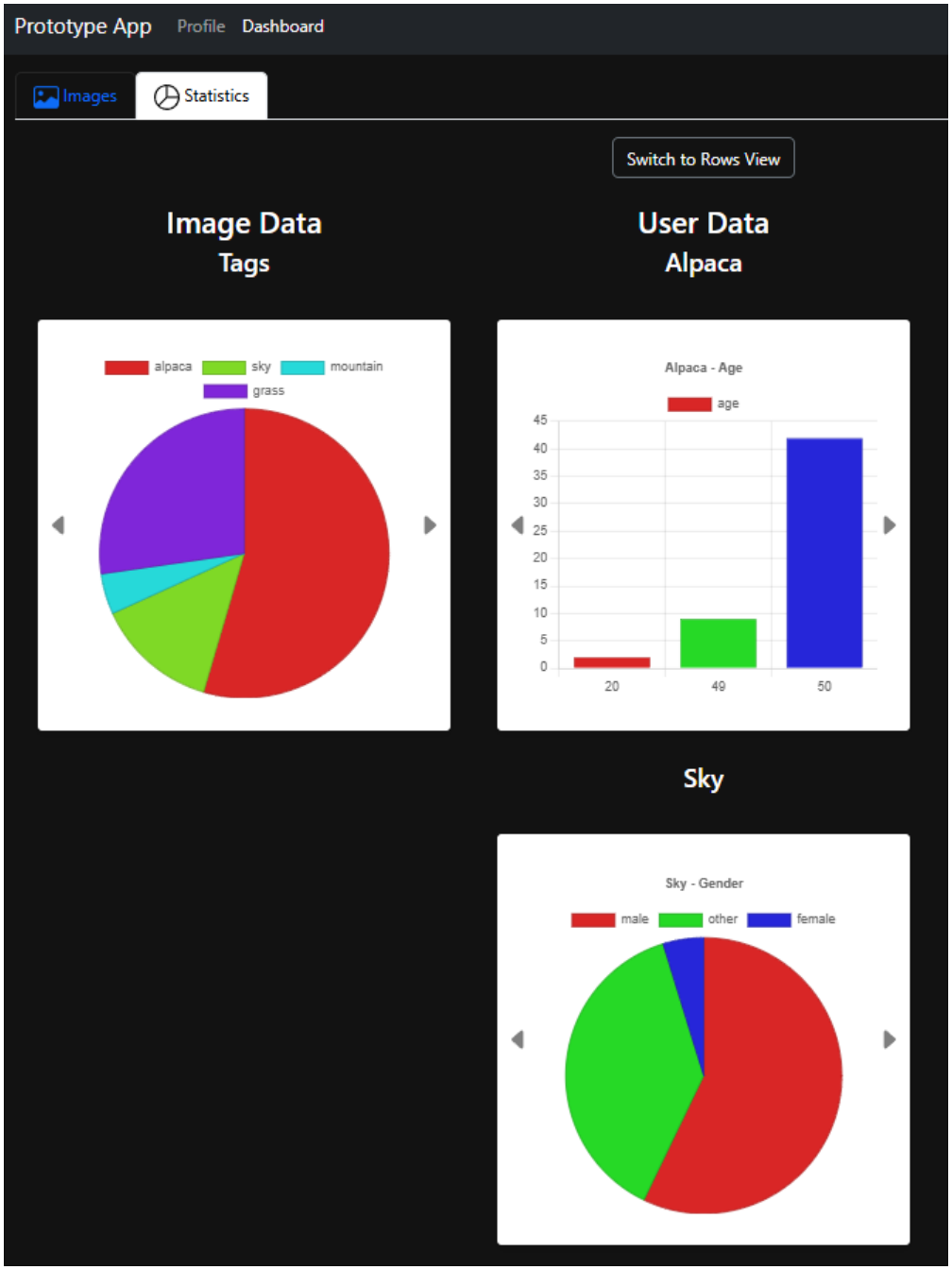

Figure 5.54: Snapshot of statistics dashboard with a different view

Chapter 6

Discussions

6.1 Interpretation of Results

6.1.1 Literature review

The literature in section 5.1.1 revealed the substantial demand for a large volume of training data, due to recent advancements in AI technology. The importance and lack of high-quality of this training data was also clearly reflected. This made it clear that current solutions for training-data Annotation are far from sufficient, supporting the theory that there is room for more and better data Annotation solutions.

While Crowdsourcing is already widely used and to some degree thriving in the training-data market, a lot of unsolved challenges still remain. The nature of Crowdsourcing raises concerns about the workers ability to produce sufficient Annotations, without subconsciously applying BIASes to the data.

The amount of examples of AI BIAS negatively affecting people´s lives tells us that AI BIAS must be investigated more and mitigated as much as possible. There was no shortage of literature stating that AI BIAS can be backtracked to training-data with disproportional representations of populations, objects or phrasings as mentioned in section 5.1.3. However literature on the annotator demographics impact on BIAS in training data was scarcer. While some literature, as displayed in section 5.1.3 indicates that a skewed demographic in annotators can increase the introduction of BIAS in training-data, none of the studies we examined discussed this finding very thoroughly. This could indicate that its an unexploited area, potentially hiding tools against BIAS in AI.

The literature review backs up the theory that our problem statement "How can user metadata be leveraged to reduce BIAS in AI datasets while ensuring compliance with GDPR?" adds onto a relatively new perspective on BIAS mitigation in AI. Developers today can to some degree monitor datasets and the diversity their content holds or lacks. However according to the literature we analysed, they have no tools to track the diversity within annotators, hence potential BIAS adopted from them.

The literature review builds up under the potential for the system we propose in this thesis, a Crowdsourcing platform with proper user-data management, annotator compensation and data processing consent that offer statistics and customization functionalities on the workers meta-data. The proposed system may have potential to gain traction in the data annotation market today.

The research scope regarding the quality demands for training data and BIAS in AI were so large that we had stop analysing at one point. Due to the time scope of this project, we had to start dedicating our time to the vast task of going through GDPR, and designing the system and prototype.

It is worth noting that the AI Act, specifically articles 10, 13, and 14, reinforces the importance of high-quality data, transparency, and human oversight in AI systems. These regulatory requirements align with our findings and emphasize the necessity for rigorous data governance to mitigate BIAS in AI systems.

6.1.2 Requirements

One of the main things we thought about when making these requirements was making sure they were compliant with GDPR. Even though the prototype doesn't completely comply with GDPR, we made sure to include the most important aspects in the prototype. All the requirements we created are meant to be the foundation for a system that fully complies with GDPR. This way, we were able to focus on showing the main functionalities that may enable techniques for a better understanding and mitigation of bias, while also recognizing the significance of privacy and protecting data. Additionally, the AI Act's focus on data quality and transparency supports our approach to developing robust requirements.

User involvement in creating user requirements

When developing user requirements, involving users is typically recommended. However, for our prototype, we chose not to involve users directly. This decision was influenced by time limitations and our focus on ensuring compliance and reducing bias under the GDPR regulations.

Prioritization

With our interpretation of the MoSCoW method we chose to add a reasoning for most of the requirements except for those who are straightforward and cannot be misunderstood. While MoSCoW is a good and easy method allowing for the delegation of features to future versions, it also had its drawbacks. Its lack of a solution for planning the order and scheduling for the implementation of each feature, could easily lead to confusion. The blurred lines between the categories, specifically "should" and "could" have was also a drawback as well as a convenience. On this basis we found it helpful to add an aspect to the method, with the priority reasoning. This way the prioritization is more robust and capable to act as a foundation for further development of the system. But our need to adjust the method is probably an indication that we should have gone with another prioritization method. If we were to go with a full-scale implementation of the system we would also have to add a schedule to guide developers on what Requirements should be implemented before others. However since we have noted linkage between the different Requirements we think we are left with a good base to create a plan for implementation of the system, and categorize different requirement groups to be implemented in individual phases of development.

Alternative approaches

While our approach was to use the GDPR guidelines as the framework for the requirement development, we considered several alternative approaches when developing the Requirements. One of those approaches was a top down approach. This approach starts by looking at the high level end of the system goals and gradually break them into more detailed components.

Another approach we considered was a user centered approach for creating Requirements. Instead of us pretending to be users to create Requirements in compliance with GDPR, we could have engaged with users through feedback sessions to get information on their needs and expectations and directly translate them into Requirements. However, we considered this approach as a very time consuming process and the Requirements would be more about user convenience rather than bias reduction while complying with GDPR.

Once we had created the Requirements, one of our alternatives was to simplify them, putting more focus on the technical aspects of bias reduction without considering GDPR compliance as much as we did. This would have allowed more resources to be dedicated to increase the prototypes capabilities in bias mitigation. However, considering the sensitivity of user metadata, we considered it more important to focus on data protection, even if not immediately achieving full compliance.

Alternative prioritization methods
There are many other methods we could have utilised when prioritizing our requirements. "Numerical assignment" is quite similar to MoSCoW but arguably more precise. Here all requirements are assigned a number from 1-5. where the numbers represent a priority group [150]:

- 1: Does not matter
- 2: Not important
- 3: Rather important
- 4: Very important
- 5: Mandatory

While more precise than our method, it would likely still have the same drawbacks, such as the lack of scheduling and proper division of prioritization between each requirement. We would probably still find it necessary to provide reasoning for each prioritization. With bubble sort all Requirements are individually ranked from most important to least important in a list [150]. While time-consuming, bubble sort would have left little confusion and the individual sorting list could easily been transformed into a schedule. In retrospect bubble sort might have been a better method for prioritizing our Requirements, but definitely more time consuming.

Further work

When eventually user feedback is integrated into the system, the Requirements can be reviewed and modified to achieve a better balance between compliance, functionality, and user experience. More functional Requirements will be created, since in most cases there is always impossible to implement all functional requirements without user feedback.

6.1.3 Automated Design Approach

The implementation of an automated design approach for generating use case and sequence diagrams in the project was aimed at maintaining the quality, traceability, and maintainability of the design documentation. This approach supports the theory that automation in software engineering can enhance efficiency and consistency across the development process. Compared to traditional manual methods, which might result in variability and potential human error, the automated approach ensures a standardized and repeatable process for diagram generation.

This finding is significant because it validates the utility of automation in managing complex software engineering tasks, particularly in the documentation and design phases. For the field of software engineering, it fills a gap in understanding how automated tools can be integrated effectively to improve the precision and reliability of design artifacts. This is especially interesting in large-scale projects where the volume and complexity of Requirements can overwhelm manual processes. The use of automated tools enhances the traceability and auditability of design processes, that also aligns with AI Act's requirements for transparency and quality management.

Based on the effectiveness of the automated design approach, it is recommended to extend the use of automation beyond diagram generation to other aspects of software development such as code generation and testing. Further research could explore the integration of AI and machine learning techniques to predict and model User requirements directly from project documentation, thereby enhancing the adaptability and intelligence of the automation tools. Practical applications might include the development of more sophisticated tools that can dynamically adjust diagrams based on real-time changes in project specifications. These advancements are interesting for elevating the standards of software engineering practices and ensuring that they can efficiently handle increasing complexity in project Requirements.

6.1.4 Use case descriptions

Generalized Terminology in Use Case Descriptions

The descriptive terms in the use case descriptions may not always match the labels or names in the prototype's interface.

Simplification for Clarity and Understanding
One of the primary reasons for using generalized terms in our Use Case Descriptions, such as "user navigates to consent settings" instead of "user navigates to profile," is to simplify and clarify the actions and intentions behind user interactions. By using a more descriptive label, we aim to make the use cases more intuitive and easier to understand for people who may not be familiar with the specific layout or terminology of the prototypes interface.

Broader application
Another reason of using generalized terms is to make the Use Case Descriptions usable in a wider range of scenarios, including future implementation of the software. For example, while the current prototype may have consent settings under the profile section, future iterations of the interface might place these settings under a dedicated menu or a different section.

6.1.5 Architecture Design

Prototype Design Considerations

The choice of React, Node.js, and Google Firebase for the prototype was driven by the need for rapid prototyping. These technologies were selected to demonstrate the feasibility of using user metadata to reduce BIAS in AI datasets, while ensuring GDPR compliance. This setup allowed for quick development and demonstration of core functionalities without extensive analysis required for a full-scale system.

Scalability and Flexibility: The combination of React and Node.js allows the system to handle dynamic content updates efficiently, fitting for the real-time nature of Crowdsourcing platforms. Google Firebase's Scalability ensures that the backend can adjust to varying loads without significant changes to the infrastructure.

Data Security and Compliance: Firebase services come with advanced security tools such as rules and authentication that, even not fully utilized in the prototype, can simplify the compliance with GDPR.

Complexity in Handling Sensitive Data: Despite the robustness of Firebase, additional efforts are required to ensure all aspects of data handling are GDPR-compliant, such as implementing comprehensive data access logs and regular security audits.

Dependency on Third-party Services: Relying on Firebase limits control over data storage and processing, migth lead to challenges in portability and flexibility, potentially complicating compliance with GDPR if policies or service terms change. UC15 and UC46, referred in (Appendix B)

During the development, the platform was engineered to allow users to choose to provide their data in a manner that is useful yet non-identifiable. This was achieved through carefully designed data handling processes that anonymize user data before it was sent to the frontend for processing, ensuring no personal identifiers are part of the statistical data.

Enhanced Data Management Features: Future iterations should include more sophisticated data management capabilities within the application, such as tools for users to view and manage their stored data directly, enhancing transparency and control. UC5, UC12, UC14, UC16, UC18 and UC40, referred in (Appendix B).

Greater Independence from Third-party Providers: Developing in-house solutions for certain backend functionalities could reduce reliance on Firebase, giving more control over data handling and compliance.

The architectural decisions for this prototype highlight the balance between rapid development and the demonstration of privacy-preserving technologies compliant with GDPR. Opting for a serverless architecture during the prototype phase illustrates the trade-offs between rapid development and operational constraints. This choice was made for demonstrating the feasibility of a GDPR-compliant, data handling platform. The architecture allows efficient management of user-generated content and ensures adaptability to evolving regulatory and technological landscapes. Additionally, there is an emphasis on the necessity for further research and development to refine and expand the system's capabilities.

Quality Attributes in Software Engineering

Security, Performance, and Scalability

Security serves as the foundation of trust for users when providing their data. Following this, Performance becomes the attribute to focus on as it ensures users continue to engage with and recommend the platform. Finally, Scalability is essential to support the expanding scope and user base of the platform. These three Quality Attributes: Security, Performance, and Scalability are central in defining the system's architecture and its operational capabilities.

Security is the foundation of user trust, especially given the sensitive nature of some of the data the platform intends to process. Effective Security measures are imperative to maintain the confidentiality, integrity, and availability of user data, safeguarding it against unauthorized access and potential breaches.

Performance should be assessed by how well the platform can handle large amounts of interactions and how the system is able to process data quickly. Essential considerations include optimizing query Performance and employing efficient search algorithms, which are pivotal for maintaining fast response times and promoting sustained user interaction.

Scalability involves preparing the system to handle an increase in user numbers and data volume effectively, without a drop in performance. It requires a strategic approach to resource management, ensuring the platforms stability and responsiveness as demand grows.

The evaluation of these Quality Attributes did not involve active users or real-world testing but was based on theoretical designs and simulated scenarios that mimic potential real-life operations. This theoretical approach helps in understanding how these attributes impact system design and user experience.

Insights and Implications
The theoretical application of these Quality Attributes demonstrates a potential alignment with modern software engineering practices. While no empirical data from actual users was collected, hypothetical analyses suggest that robust security protocols could reduce system vulnerabilities.

This focus on Security, Performance, and Scalability is crucial not only for the user experience but also positions the platform to compete in a market that demands rapid, secure, and scalable digital solutions. Moreover, these attributes provide a solid foundation for future enhancements, allowing the platform to adapt to technological advances and evolving user needs.

Future Directions
Continual improvements in Security, Performance, and Scalability are essential given the rapid evolution of technology and user expectations. Future theoretical research might explore cutting-edge security solutions like AI-driven threat detection, performance enhancements using next-generation databases, and scalability through serverless computing architectures. These advancements could theoretically be integrated into the platform to refine user experiences and operational efficiency.

This discussion underscores the important role of security, performance, and scalability in the theoretical success of software platforms. It offers insights that can guide developers and industry professionals in optimizing their systems within competitive and dynamic markets.

6.1.6 High-level diagrams

The research illustrates that while high-level diagrams are vital for initial stages of system design, they must be dynamically updated and supplemented with detailed diagrams to fully capture the evolving Requirements and complexities of software systems.

Referring to the discussion in the section "Automated Design Approach", the foundational use of high-level diagrams facilitated the subsequent creation of mid-level diagrams. While these high-level diagrams were limited to single use cases, which restricted a broader system overview, they were accurate and well-aligned with use case descriptions and Requirements, serving effectively as a preliminary step in the design process.

However, the limitations of these high-level diagrams cannot be overlooked. Their focus on single scenarios sometimes resulted in an incomplete representation of the system's full functionality, potentially leading to oversights in broader system requirements. This could pose challenges in fully understanding all user interactions and system processes without additional, more detailed diagrams.

Based on these findings, further research should investigate the development of more integrated diagrammatic tools that encompass multiple use cases or complete system functionalities. Future applications could focus on enhancing diagrammatic tools to automatically update and adapt to changes in system requirements or user data. Such advancements could impact the development practices in software engineering, ensuring that complex systems are both understandable and compliant with legal standards.

6.1.7 High- and mid-level diagrams for prototype

From the initial high-level diagrams that outlined the system concept, we transitioned to focus specifically on both high- and mid-level diagrams for the prototype design. This shift was pivotal as it allowed us to refine our conceptual understanding into more detailed plans suitable for prototyping.

After we finished creating the requirements, descriptions, and diagrams for GDPR compliance, we shifted our focus to developing the prototype. We began by designing the prototype using high-level component, use case, and sequence diagrams. We selected use cases from the ones we had already created to develop the prototype, making sure to address the key functionalities and compliance requirements. This approach made sure we addressed both key functionalities and compliance requirements.

Following the high-level component diagram, we made a high-level use case diagram. This diagram was important for the development process because it provided an overview of all possible interactions between the users (both data providers and data consumers) and the system. It helped us identify key functionalities such as image uploading, annotation tasks, consent management, image searching, and data visualization.

After we created the high-level diagrams, we created a mid-level component diagram. The reason we developed the mid-level diagram was to provide a more detailed view of the prototypes architecture, allowing us to go deeper into each components specific responsibilities and interactions. This approach was helpful to accurately implement the prototypes functionalities.

Following the mid level component diagram, we developed detailed mid-level sequence diagrams for the use cases we selected to develop the prototype. These sequence diagrams provided a more detailed view of the interactions within the system, ensuring that each step of the process was clearly mapped and understood. This detailed reasoning about the system was very useful for identifying potential issues and streamlining the development process.

We are aware that moving from high-level diagrams to mid-level diagrams before development in a complete system might limit the opportunity to include user feedback or testing result that could influence the design and lead to a mismatch between user needs and system capabilities. However, since this is a bachelor project with limited time and resources and not a fully operational system we decided not to involve any users or testing, focusing instead on theoretical design validation.

6.1.8 Database

The database is designed in a denormalized fashion to optimize the search process for data consumers. This structure supports rapid retrieval and processing of user tasks, images, and GDPR consents, essential for handling large volumes of metadata efficiently. The system's architecture also ensures that each image can be associated with multiple tasks, and each task can involve several images, facilitating flexible relationships managed through separate collections.

The use of a denormalized structure allows for quick access and manipulation of data, which is crucial for a system relying on extensive data analysis and real-time processing. This design supports the system's ability to leverage user metadata effectively, which support the research question.

While denormalization improves read performance, it can introduce redundancy and increase storage costs. Furthermore, maintaining data integrity might become challenging as the volume of data grow, requiring more sophisticated synchronization and update mechanisms. The separated handling of images and tasks, although beneficial for flexibility, also demands rigorous data management to ensure consistency and integrity.

Further research should explore the balance between normalization and denormalization, particularly how it affects performance and scalability in cloud environments. Alternative database architectures, such as hybrid designs that combine the benefits of both normalization and denormalization, should also be considered. Implementing advanced indexing strategies and exploring further encryption and security measures could enhance both performance and data consistency while maintaining compliance with GDPR.

6.1.9 Prototype

Consent

The GDPR consent functionality, as demonstrated in the prototype, allows users to manage their data preferences, requiring active participation to ensure informed consent. This approach aligns with GDPR requirements, enhancing user trust and ensuring legal compliance. The prototype's reliance on users accurately reading and understanding the consent forms could limit its effectiveness. Future work should explore more interactive and educational consent processes to increase user understanding and engagement.

Tags

For now the prototype only supports the annotation technique for image classification. While mostly for demonstration purposes, this feature is key when developing training data. The potential for creating statistics on the diversity of different elements withing a dataset must be recognised. If relevant, the degree of detail in the tags can be adjusted to allow for better insight and monitoring of potential BIAS. The further development of the project must implement more advanced techniques like referenced in section 2.2, to allow for better quality in the training data. The demanded degree of detail in tags should be dedicated to further research. The way tags are applied affects what information the AI-models obtain from the training data.

An insurance company might want to analyse the occurrence rate of accidents involving a Ford focus vs BMW 3 series. For an AI model to be able to assist such cases, the detail in its training data is crucial. This kind of detail in the tags is also helpful to provide information of the variety of elements represented within a data-set, helping BIAS mitigation. Detailed tagging and monitoring of variety within training data could also allow for intentional BIAS introduction. This is important to address and investigate, as it could present marketing opportunities but also the potential for intentional skewed datasets to implement BIAS in a model that is intended for discrimination.

As this are topics that raises some major ethical dilemmas, this project has not gone into detail here, but there must be done further research on how these cases should be handled. Thus how the system should behave if such filtering is requested in a dataset. If a dataset is requested to contain only BMW cars that would probably be fine, but if a request wants only humans with a specific ethnicity present in a dataset, that might not be okay.

Upload

The image upload feature restricts users to predefined tags, reducing the risk of inappropriate metadata but limiting the richness of the data. This design aims to minimize misspelling, duplicate tags, explicit text and to some degree injection-related security issues. However, while it addresses these aspects, it may not capture all relevant image contexts. The limitation on user-created tags can reduce the utility and accuracy of the annotations. For future work we would investigate the impact of allowing user-generated tags under controlled conditions to enhance metadata diversity without compromising dataset quality.

Filtering explicit content

While the prototype does not have any functionality on filtering explicit content, the exploration of potential strategies to manage explicit content must be implemented for a full-scale system. We will evaluate potential approaches on how to filter explicit content for future work.

Content moderation APIs: To filter out explicit content by screening text and images, we have looked at two different APIs as referred in 3.16. These are some of the most advanced out there. A considerable issue by using such APIs is that the data would potentially be processed by other organizations. These issues will require careful consideration of data privacy considering our system has to comply with GDPR.

Custom Machine learning models: By integrating custom machine learning models to filter explicit content, the system can enhance its capacity for content moderation beyond what is achievable with standard content moderation APIs. This approach would allow for more nuanced control over the data and the criteria applied for content moderation, tailored specifically to the platform's unique requirements. An interesting perspective to consider is that the more explicit content that is uploaded, the better the data foundation becomes for training a custom model. This implies that increased occurrences of explicit content not only challenge the system but also provide valuable data that can improve the model's accuracy and responsiveness over time. This potential data-rich environment could enhance the system's ability to learn and adapt, making it more robust in handling diverse content moderation scenarios. This aspect was not explored within the scope of our project due to the substantial resources required for data collection and model maintenance. However, it represents a strategic avenue for future development, particularly in enhancing the platform's capabilities in automated content moderation and ensuring a safe user environment.

Explicit text: The prototype uses a specific set of tags that users can choose when they upload images. This helps prevent any explicit text from entering the prototype since all tags are chosen and added by us. By managing the words users can use, the system reduces the chances of inappropriate content being tagged or described. In future work we would open up for users to suggest tags by sending requests to us. This gives us the full control of which tags are gonna be in the system or not. However, in a startup phase, this method might have limitations in terms of scalability and user engagement. As the images uploaded will increase, the initial set of tags could quickly become insufficient for the users, potentially limiting users ability to accurately tag their images. Additionally, the lack of user input in tag creation may reduce user engagement.

Ethical considerations are essential when we discuss content moderation, especially user privacy and the transparency of the moderation process. Future implementation of content moderation in the system should be designed with strong ethical guidelines to ensure user data is handled responsibly. This could be done by involving users in the moderation process by implementing a report functionality.

Including explicit content: To allow explicit content for academic or research purposes, we need to implement guidelines and functionalities that enable safe annotation of such content. This includes ensuring that only authorized users have access to this segment, thereby preventing exposure to general users.

Image ownership and property rights: In the prototype we have currently implemented a basic mechanism that tracks which user uploads each image by tracking the user id. This is enough for the prototype but in a fully implemented system it needs to get extended to managing copyrights, licenses and attributes uploaded by users. This would reduce the risk other users using images that is already uploaded by others. We have ensured that there is a minimal traceability of image ownership which is required when dealing with user generated content. For future work a licensing framework called Creative Commons Chooser [151] can be implemented. This tool allows users to select appropriate licenses for their images, which will let them specify how others may use the uploaded images legally. This reduces the legal risks for both the user providing the content, the stakeholders and those who might use it.

Hashing was also something we discussed within the group. In addition to a licensing framework, we discussed that adding hashing to the system would further prevent duplicates being made by users. By creating an unique fingerprint for each image when it is uploaded, the system can identify and prevent duplicate images from being stored in the database, thereby ensuring that each image is unique and prevent the occurrence of duplicates. However, if there is one pixel in the image that is different, the unique hash will not be identical, potentially causing the system to treat nearly identical images as distinct entries [152]. This could lead to inefficiencies and an increased storage of similar images.

Task

The task component functions as the "link" or "portal" through which users connect their consented information to the images they are annotating. The design collects user-generated Annotations, which can be linked to their metadata, enhancing the database with valuable information such as user data, preferences, or other relevant attributes. While it is necessary for the system's operation, the specific design of the task interface is not the primary focus at this stage. However, design aspects will be a factor for Usability and should be considered for future development.

By including user responses along with their metadata, the system improves its dataset. This helps in developing more advanced machine learning models for organizing content or making recommendations.

Based on the functionality and outcomes of the task component in the prototype, the following recommendations are proposed for further research and application:

- **Advanced Questioning Mechanisms:** Investigate the implementation of more complex questioning mechanisms that can adjust dynamically based on the user's previous responses or their profile, thereby personalizing the experience and potentially improving the quality of data collected.
- **Machine Learning Integration:** Explore the integration of machine learning algorithms that can learn from the annotated data to automate some of the simpler Annotation tasks or to refine the questions posed to users.
- **User Feedback System:** Implement a feedback system where users can provide input on the task difficulty or relevance, which can be used to further refine the task generation algorithms.
- **Cross-Validation System:** Develop a cross-validation system where responses from multiple users can be used to verify the accuracy of Annotations, enhancing the reliability of the data collected.

The task component of the prototype serves as a fundamental bridge between user interaction and data collection, playing a key role in the system's overall ability to gather and utilize user-generated content effectively. Further research and development in this area could lead to more sophisticated systems capable of leveraging user input to enhance machine learning models and improve content Personalization and accuracy.

Dashboard

The dashboard is intended to support extensive search and filtering capabilities based on annotator metadata, facilitating targeted data retrieval and analysis. The prototype contains a simple demonstration of this functionality.

As the literature review revealed, challenges related to tracking, understanding, and accessing data for adjusting the dataset for bias mitigation are significant in the field of AI. In this light, the functionality provides direct support for researchers and developers who want to address these challenges. The concept of a dashboard like ours will provide the tools for advanced research and collaboration.
The aggregation and anonymization of user data provide a somewhat unique arena for utilizing user data while preserving rights and privacy. This approach provides insights without exposing individual users, supporting efforts to analyze, understand, and mitigate bias in datasets.

While the dashboard is a functional demonstration of the concept, it lacks many features. Some limitations include:

- Custom dataset creation
- Subscribing to collections
- Ordering more data

These features have to be added before a full scale roll-out of the system.

Future developments should include tools for users to customize datasets, more advanced techniques for search, create "favorite collections", subscribe to data, and request data directly from the dashboard. These enhancements would improve the platform's flexibility and applicability in diverse AI projects.

Filtering dashboard: As of now, the filtering dashboard has been implemented with basic filtering functionalities for demonstration purposes. The reason for this is that we wanted to create the filtering mechanism through user involvement. We want to ensure the filtering functionality meets the practical needs and preferences of end-users. We want the prototype, with its core functionalities, to be a demonstration of the concepts in the project. The architecture of the prototype is designed to support more advanced filtering functionalities, as allowed by the database.

Statistics

The prototype show several ways of visualizing the composition of the search result. The statistics section is an important part of the prototype. This is where the system will differ from others and offer a unique tool for bias-understanding and mitigation. As of now the prototype is capable of visualizing the composition of tags and annotator demographics for a search result.

This feature offers another dimension of information about the images in a search result. By quickly identifying the most common tags and demographics of annotators, developers can gain a better understanding of the biases and strengths in the search result.

As of now the prototype don't allow users to extract a dataset from a search result. This feature must be designed and implemented in a user friendly way for the system to have any value.

The way statistics are displayed as of now is not optimized, user interactions should be arranged to obtain an understanding of how this can be better solved. What statistics to display and how detailed it should be is also something that must be investigated more to ensure the system provides the most valuable information for its users. The UI design regarding layout and coloring is for now not optimised because the prototype is meant to only demonstrate the concept. However it must be adjusted to give a more professional and user friendly impression. A proposed feature is a search/filter-approach that lets the user set the wanted distribution (of e.g age, gender, etc) which the search result will match. To further develop this section, user tests and interviews must be arranged to gain understanding of what information AI developers mostly desire.

6.2 Limitations

Limitiations are mentioned in each of the topics in section 6.1. Additional limitations are mentioned below.

Assumptions about Providers and Consumers of Data: The project is based on certain assumptions about the data providers (annotators) and data consumers (researchers and developers). However, these assumptions may not always be accurate. This could impact how applicable the findings and design of the prototype are in various contexts.

Statistical Inference: The prototype and its findings cannot be used to make statistical inferences. The results can generate hypotheses but not confirm them due to the limited scope and scale of the study.

Sample Size and Scalability: The small sample size used in the study do not provide a realistic test of the prototype's scalability. To address this, future work should include strategies with larger datasets to better evaluate performance at scale.

Time and User Constraints: The study was conducted with time and resource constraints, limiting the extent of user involvement and the depth of testing. Despite these constraints, involving a larger number of users in future iterations could provide more comprehensive insights and validation.

Limited Literature on User Metadata: There are not many papers that focus on using user metadata to reduce bias in AI. This indicate a need for more research in this area to establish a better foundation for future work.

Extensive Literature Review: The literature on other relevant topics was found to be extensive. Due to time constraints, it was necessary to selectively include only certain papers. This limitation suggests that more comprehensive reviews could reveal additional insights and methodologies.

Member state law: To properly comply with GDPR the law of all member states of the EU and EEA must be investigated. I.e GDPR art.6 allows for member states to introduce more specific provisions to adapt the applications of the rules in GDPR.

Data export outside EU: Before the system is rolled out, the regulations regarding data export to countries outside EU must be investigated. A running server in Asia would be preferable if the system gains a user-base there and as we use Googles storage services, proper GDPR compliance here must be further ensured with consultation by legal expertise.

6.3 Implications

Implications are mentioned in each of the topics in section 6.1. Additional implications are mentioned below.

Impact on Bias Mitigation: The study supports the theory that integrating user metadata into AI training processes can directly facilitate bias mitigation. This is a step in the right direction for developing more equitable AI systems and could guide future interventions and technological advancements.

Prototype Utility: The prototype being proposed shows a new way to manage user data that could impact how future AI systems are designed. It includes examples of how bias can be detected and reduced, setting a standard for including user input and ethical concerns in AI development.

Future Research Directions: The findings indicate areas for future research, including more robust methods for user involvement, advanced techniques for dataset customization, and evaluations of system scalability and effectiveness.

6.4 Applications

Recommendations and applications are mentioned in each of the topics in section 6.1. Additional recommendations and applications are mentioned below.

The primary application of this research is to develop a platform for data annotation with user-data management. This platform aims to gather a variety of high-quality data, ensure fair compensation for annotators, and handle their data processing consents in compliance with GDPR regulations. It can be useful for organizations and academic institutions looking for data annotation solutions. The platforms capability to monitor and personalize annotator metadata can provide useful information for improving the quality and diversity of training data, leading to more equitable AI systems.

Chapter 7

Conclusions

This thesis set out to explore the potential of leveraging user metadata to mitigate bias in AI datasets while ensuring compliance with GDPR. The research was guided by the primary question:

> *How can user metadata be leveraged to reduce bias in AI datasets, while ensuring compliance with GDPR?*

Our research concludes with an approach that we believe to be a solid foundation for mitigating bias in AI.

Data and Metadata for Mitigating Bias:
Our results suggest that increasing the diversity of data and annotators can be a significant factor in mitigating bias. The literature review supports the notion that more diverse data and annotators contribute to more equitable AI systems. However, our research also highlights the need for further studies to confirm these findings across larger and more varied populations.

Compliance with GDPR:
Our findings demonstrate that compliance with GDPR, while complex, is achievable with proper measures. Ensuring that users are informed, understand the implications, actively consent, and have the ability to modify their consent easily and without negative consequences is an absolute requirement. The prototype's design aligns with these principles, providing a foundation for GDPR-compliant systems that also address AI bias. The AI Act's requirements for data quality, transparency, and human oversight further supports the robustness of our approach.

Through a thorough examination of all the articles of GDPR, an extensive literature review, the development of a prototype, and analysis of the findings, key conclusions can be drawn:
Bias in AI: The literature review highlighted the widespread issue of bias in AI, particularly how biases in training data often only become apparent after AI models are deployed. While many studies emphasize the impact of disproportionate representation in datasets on AI bias, there is a notable gap in research regarding the demographics of annotators and their contribution to bias. This suggests that managing and utilizing user metadata, including annotator demographics, presents an underexplored avenue for bias mitigation.
Integrating User Metadata: The development and implementation of the prototype demonstrated the practical feasibility of integrating user metadata into AI datasets to reduce bias. The prototype illustrates the concept and serves as a lightweight demonstration of a potential crowdsourcing platform. Importantly, the prototype was designed to demonstrate GDPR concepts to ensure that user data is managed in a manner that respects privacy and consent requirements. Also in aligning with ongoing regulatory developments like the proposed AI Act.

In conclusion:
This thesis provides evidence that leveraging user metadata can be a promising strategy for reducing bias in AI datasets and that it is possible to do so in compliance with GDPR. This approach not only addresses a critical issue in AI development but also aligns with the broader goals of fairness, accountability, and transparency in AI. Future studies should further investigate this intersection, particularly by examining larger-scale applications. They should specify the types of data and user metadata needed for more research on bias in AI systems and ensure compliance with ongoing regulatory developments, such as the proposed AI Act.

Chapter 8

Appendix

A requirements

requirements.xlsx

B Use case description

Use Case Descriptions - All.pdf

C data_preprocessing

data_preprocessing.ipynb

D df_combined_stacked

df_combined_stacked.json

E trace_matrix

trace_matrix.json

F enhanced_trace_matrix

enhanced_trace_matrix.json

G visualization

visualization.ipynb

H requirement_groups

requirement_groups.json

I nlp

nlp.ipynb

J requirement_groups_with_summaries

requirement_groups_with_summaries.json

K enriched_groups_with_summaries

enriched_groups_with_summaries.json

L openAI

openai.ipynb

M use_case_descriptions

use_case_descriptions.ipynb

N diagrams

diagrams.ipynb

O Diagrams Folder

The folder containing the diagrams used in the report.

P manual_diagrams

manual_diagrams.ipynb

Bibliography

[1] Leonie Senn-Kalb and Dev Mehta. *In-Depth Report on Artificial Intelligence*. Statista, Apr. 2023. URL: https://www.statista.com/study/50485/in-depth-report-artificial-intelligence/.

[2] Statista. *Artificial Intelligence - worldwide*. Accessed: May 4, 2024. 2024. URL: https://www.statista.com/outlook/tmo/artificial-intelligence/worldwide%7D.

[3] Business Research Insights. *Ai Training Dataset Market Size, Share, Growth, and Indusry Analysis*. Tech. rep. Accessed: May 5, 2024. Business Research Insights, 2024. URL: https://www.businessresearchinsights.com/market-reports/ai-training-dataset-market-110110.

[4] The Brainy Insights. *AI Training Datasset Market size, Global Industry Analysis*. Tech. rep. Feb. 12, 2024. The Brainy Insights, 2023. URL: https://www.thebrainyinsights.com/report/ai-training-dataset-market-13562#summary.

[5] Javatpoint. *Machine Learning Life Cycle*. Accessed: May 11, 2024. URL: https://www.javatpoint.com/machine-learning-life-cycle.

[6] Alex McFarland. *How AI is Creating Explosive Demand for Training Data*. Accessed: Feb. 12, 2024. 2023. URL: https://www.unite.ai/how-ai-is-creating-explosive-demand-for-training-data/.

[7] Natasha Noy and Omar Benjelloun. *Datasets at your fingertips in Google Search*. Accessed: May 24, 2024. URL: https://research.google/blog/datasets-at-your-fingertips-in-google-search/.

[8] Wikipedia contributors. *List of datasets for machine-learning research*. Accessed: May 24, 2024. 2024. URL: https://en.wikipedia.org/w/index.php?title=List_of_datasets_for_machine-learning_research&oldid=1221075088.

[9] Papadopoulos et al. "Extreme Clicking for Efficient Object Annotation." In: *IEEE* (Oct. 2017). DOI: 10.1109/iccv.2017.528. URL: https://doi.org/10.1109/iccv.2017.528.

[10] R. Schwartz A. Vassilev K. Greene L. Perine A. Burt P. Hall. *Towards a Standard for Identifying and Managing Bias in Artificial Intelligence*. Accessed: May 5, 2024. 2022. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1270.pdf.

[11] European Commission. *A European Strategy for Data*. Accessed: Feb. 12, 2024. 2020. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A52020DC0066.

[12] Thordis Sveinsdottir et al. *The Role of Data in Artificial Intelligence*. Tech. rep. Accessed: Feb. 12, 2024. Dec. 2020. DOI: 10.5281/zenodo.4312907. URL: https://gpai.ai/projects/data-governance/role-of-data-in-ai.pdf.

[13] IBM Data and AI Team. *Shedding light on AI bias with real world examples*. Accessed: April 30, 2024. 2023. URL: https://www.ibm.com/blog/shedding-light-on-ai-bias-with-real-world-examples/.

[14] Judy Wawira Gichoya et al. *AI pitfalls and what not to do: mitigating bias in AI*. Accessed: Feb. 12, 2024. Oct. 2023. DOI: 10.1259/bjr.20230023. URL: https://academic.oup.com/bjr/article/96/1150/20230023/7498925.

[15] Morten Goodwin. "Personal communication." private meeting including our supervisor Arne Wiklund discussing leveraging meta-data to mitigate AI bias.

[16] Lama H. Nazer et al. "Bias in artificial intelligence algorithms and recommendations for mitigation." In: *PLOS Digital Health* 2.6 (June 2023), e0000278. DOI: 10.1371/journal.pdig.0000278. URL: https://journals.plos.org/digitalhealth/article?id=10.1371/journal.pdig.0000278.

[17] Appen. *Top Eight Ways to Overcome and Prevent AI Bias*. Accessed: Feb 10, 2024. 2020. URL: https://www.appen.com/blog/how-to-reduce-bias-in-ai.

[18] B. J. Copeland. "artificial intelligence." In: *Enclypedia Britannica* (2024). accessed April 27, 2024.

[19] Intersoft Consulting. *Artificial Intelligence Act AI Act*. Accessed: May 20, 2024. 2024. URL: https://ai-act-law.eu/.

[20] IBM. *What is artifitial intelligence (AI)?* Accessed: April 27, 2024. URL: https://www.ibm.com/topics/artificial-intelligence.

[21] B. Nancholas. *Narrow artificial intelligence: advantages, disadvantages, and the future of AI*. Accessed: April 27, 2024. URL: https://online.wlv.ac.uk/narrow-artificial-intelligence-advantages-disadvantages-and-the-future-of-ai/.

[22] Shaip. *A Beginner's Guide to Data Annotation: Tips and Best Practices*. Accessed: April 28, 2024. 2024. URL: https://www.shaip.com/blog/the-a-to-z-of-data-annotation/.

[23] Cloudfactory. *Image Annotation for Computer Vision*. Accessed: April 28, 2024. 2020. URL: https://www.cloudfactory.com/image-annotation-guide.

[24] Google. *ML Practicum: Image Classification*. Accessed: April 28, 2024. URL: https://developers.google.com/machine-learning/practica/image-classification.

[25] H. Bandyopadhyay. *Image Annotation: Definition, Use Cases Types*. Accessed: April 28, 2024. 2023. URL: https://www.v7labs.com/blog/image-annotation-guide.

[26] M. .K Pratt C Gonsalves. *Crowdsourcing*. Accessed: April 30, 2024. 2023. URL: https://www.techtarget.com/searchcio/definition/crowdsourcing.

[27] T. Hoßfeld M. Hirth and P. Tran-Gia. "Anatomy of a crowdsourcing platform - using the example of Microworkers.com." In: *Ieeexplore* (June 2011). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5976179.

[28] C. Reffel. *Top 5 Crowdsourced Data Annotation Solutions*. Accessed: April 30, 2024. 2021. URL: https://crowdsourcingweek.com/blog/top-5-crowdsourced-data-annotation-solutions/.

[29] Cambridge University. *Cambridge Dictionary*. URL: https://dictionary.cambridge.org/dictionary/english/bias. Accessed: April 30, 2024.

[30] R. J. Thomas and T. J. Thomson. "Ageism, sexism, classism and more: 7 examples of bias in AI-generated images." In: *The conversation* (July 2023). URL: https://theconversation.com/ageism-sexism-classism-and-more-7-examples-of-bias-in-ai-generated-images-208748#:~:text=There%20were%20also%20notable%20differences,of%20more%20fluid%20gender%20expression.

[31] Suhlle Ahn. *The same biases baked into all our systems are getting baked into AI*. Image accessed May 8, 2024. URL: https://www.tidalequality.com/blog/understanding-and-addressing-ai-bias.

[32] Intersoft Consulting. *General Data Protection Regulation (GDPR)*. Accessed: April 23, 2024. 2016. URL: https://gdpr-info.eu.

[33] RecFaces. *What Is GDPR? Understanding General Data Protection Regulation*. Accessed: May 24, 2024. URL: https://recfaces.com/articles/gdpr-explained.

[34] T. H. Frækaland H. W. Bergsvik and J. A. B. Larssen. "Labor Practices and GDPR Compliance in AI." Our group exam in ING200-G 24V.

[35] *Personvern og innovasjon*. Accessed: Februar 12, 2024. 2022. URL: https://digitalnorway.com/kurs/hvordan-lykkes-med-gdpr-i-innovasjonsprosjekter/.

[36] B. Wolford. *Does the GDPR apply to companies outside of the EU?* Accessed: May 1, 2024. 2023. URL: https://gdpr.eu/what-is-data-processing-agreement/.

[37] Tufts University. *EEA UK General Data Protection Regulation (GDPR)*. Accessed: May 1, 2024. URL: https://access.tufts.edu/eea-uk-general-data-protection-regulation-gdpr.

[38] Intersoft Consulting. *Rights of the data subject*. Accessed: April 23, 2024. 2016. URL: https://gdpr-info.eu/chapter-3/.

[39] B. Wolford. *What is a GDPR data processing agreement*. Accessed: May 1, 2024. 2023. URL: https://gdpr.eu/companies-outside-of-europe/.

[40] Intersoft Consulting. *Art. 5 GDPR, Processor*. Accessed: May 1, 2024. 2016. URL: https://gdpr-info.eu/art-28-gdpr/.

[41] European Commission. *AI Act*. Accessed: May 20, 2024. 2024. URL: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai.

[42] K.E. Wiegers and J. Beatty. *Software Requirements*. Third edition. Best practices. Redmond Washington: Microsoft Press, 2013. ISBN: 9780735679665. URL: https://books.google.no/books?id=40lDmAEACAAJ.

[43] Google. *Data and Security*. Accessed: May 25, 2024. URL: https://www.google.com/about/datacenters/data-security/.

[44] *Software Requirements*. Accessed: May 1, 2024. URL: https://www.inf.ed.ac.uk/teaching/courses/ip/CS2Ah0405-SoftwareRequirements.pdf.

[45] Tatiana Kravchenko, Tatiana Bogdanova, and Timofey Shevgunov. "Ranking Requirements Using MoSCoW Methodology in Practice." In: *Cybernetics Perspectives in Systems*. Ed. by Radek Silhavy. Cham: Springer International Publishing, 2022, pp. 188–199. ISBN: 978-3-031-09073-8.

[46] TheStudio. *Moscow prioritization method model vector illustrattion*. Image accessed May 8, 2024. URL: https://www.freepik.com/premium-vector/moscow-prioritization-method-model-vector-illustrattion_30162650.htm.

[47] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Object Technology Series. Pearson Education, 2018. ISBN: 9780134865126. URL: https://books.google.no/books?id=VTdtDwAAQBAJ.

[48] L. Bass et al. *Software Architecture in Practice, 4th Edition*. SEI series in software engineering. Addison-Wesley Professional, 2021. ISBN: 9780136885979. URL: https://books.google.no/books?id=BWpuzgEACAAJ.

[49] D.C. Schmidt and F. Buschmann. "Patterns, frameworks, and middleware: their synergistic relationships." In: *25th International Conference on Software Engineering, 2003. Proceedings.* 2003, pp. 694–704. DOI: 10.1109/ICSE.2003.1201256.

[50] Microsoft. *Design the infrastructure persistence layer*. Accessed: May 11, 2024. 2023. URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design.

[51] Microsoft. *Design the infrastructure persistence layer*. Accessed: May 11, 2024. URL: https://www.vmware.com/topics/glossary/content/configuration-management.html.

[52] P. S. Aithal and Shubhrajyotsna Aithal. "New Research Models under Exploratory Research Method." In: *Emergence and Research in Interdisciplinary Management and Information Technology*. Ed. by P.K. Paul et al. New Delhi, India: New Delhi Publishers, 2023. Chap. 7, pp. 109–140. DOI: 10.2139/ssrn.4490827. URL: https://ssrn.com/abstract=4490827.

[53] Anna-Maria Teperi, Nadezhda Gotcheva, and Kirsi Aaltonen. "16 - Design thinking perspective for developing safety management practices in nuclear industry." In: Woodhead Publishing Series in Energy (2021). Ed. by Anna-Maria Teperi and Nadezhda Gotcheva, pp. 309–326. DOI: https://doi.org/10.1016/B978-0-08-102845-2.00016-8. URL: https://www.sciencedirect.com/science/article/pii/B9780081028452000168.

[54] Jan Vom Brocke Alan Hevner Alexander Maedche. *Introduction to Design Science Research.* Accessed: May 1, 2024. 2020. URL: https://www.researchgate.net/publication/345430098_Introduction_to_Design_Science_Research.

[55] Python Software Foundation. *What is Python? Executive Summary.* Accessed: April 30, 2024. URL: https://www.python.org/doc/essays/blurb/.

[56] Benjamin Zeman. *What is Google Colab?* Accessed: April 30, 2024. 2023. URL: https://www.androidpolice.com/google-colab-explainer/.

[57] Google. *googleapis/google-api-python-client: The official Python client library for Google's discovery based APIs.* Accessed: April 30, 2024. URL: https://github.com/googleapis/google-api-python-client.

[58] Google. *google-auth.* Accessed: April 30, 2024. 2016. URL: https://google-auth.readthedocs.io/en/master/#:~:text=google%2Dauth%20is%20the%20Google,for%20signing%20and%20verifying%20JWTs%20..

[59] Anton Burnashev. *gspread.* Accessed: April 30, 2024. URL: https://docs.gspread.org/en/v6.0.0/.

[60] Robin Thomas. *Welcome to gspread-dataframe's documentation!* Accessed: April 30, 2024. 2017. URL: https://gspread-dataframe.readthedocs.io/en/latest/#:~:text=1-,gspread_dataframe,a%20worksheet%20using%20a%20pandas..

[61] Robin Thomas. *Welcome to gspread-formatting's documentation!* Accessed: April 30, 2024. 2017. URL: https://gspread-formatting.readthedocs.io/en/latest/.

[62] pandas. *pandas.* Accessed: April 30, 2024. URL: https://pandas.pydata.org/#:~:text=pandas%20is%20a%20fast%2C%20powerful,of%20the%20Python%20programming%20language..

[63] Numpy developers. *What is NumPy?* Accessed: April 30, 2024. URL: https://numpy.org/doc/stable/user/whatisnumpy.html.

[64] NetworkX developers. *Software for complex networks.* Accessed: April 30, 2024. URL: https://networkx.org/.

[65] The Matplotlib development team. *Matplotlib: Visualization with Python.* Accessed: April 30, 2024. URL: https://matplotlib.org/.

[66] *What is Plotly?* Accessed: April 30, 2024. URL: https://domino.ai/data-science-dictionary/plotly.

[67] *What is NLP?* Accessed: April 30, 2024. URL: https://www.ibm.com/topics/natural-language-processing.

[68] Nimra Shahzadi. *Unleashing the Power of Natural Language Processing: A Journey into Text Understanding and Generation.* Image accessed May 8, 2024. URL: https://medium.com/@nimrashahzadisa064/unleashing-the-power-of-natural-language-processing-a-journey-into-text-understanding-and-2d4d46223619.

[69] *What is spaCy?* Accessed: May 1, 2024. URL: https://domino.ai/data-science-dictionary/spacy.

[70] $en_core_web_trf$. Accessed: May 1, 2024. URL: https://spacy.io/models/en#en_core_web_trf.

[71] *Trained Models Pipelines.* Accessed: May 1, 2024. URL: https://spacy.io/models.

[72] *What are Hugging Face Transformers?* Accessed: May 1, 2024. 2024. URL: https://docs.databricks.com/en/machine-learning/train-model/huggingface/index.html#:~:text=Hugging%20Face%20Transformers%20is%20an,tune%20them%20to%20maximize%20performance..

[73] *CodeBERT*. Accessed: May 1, 2024. URL: https://paperswithcode.com/method/codebert#:~:text=CodeBERT%20is%20developed%20with%20a%20Transformer%2Dbased%20neural%20architecture%2C%20and,plausible%20alternatives%20sampled%20from%20generators..

[74] Guo et. al. *GraphCodeBERT model*. Accessed: May 1, 2024. URL: https://huggingface.co/microsoft/graphcodebert-base.

[75] *T5 (language model)*. Accessed: May 1, 2024. 2019. URL: https://en.wikipedia.org/wiki/T5_(language_model)#:~:text=T5%20(Text%2Dto%2DText,that%20they%20were%20pretrained%20for..

[76] Greg Brockman Mira Murati Peter Welinder OpenAI. *OpenAI API*. Accessed: May 1, 2024. 2020. URL: https://openai.com/blog/openai-api.

[77] Accessed: May 1, 2024. URL: https://platform.openai.com/playground/compare?models=gpt-4-turbo&models=gpt-3.5-turbo.

[78] Kirill Fakhroutdinov. *UML 2.5 Diagrams Overview*. Accessed: May 25, 2024. URL: https://www.uml-diagrams.org/uml-25-diagrams.html.

[79] *PlantUML at a Glance*. Accessed: May 1, 2024. URL: https://plantuml.com/.

[80] postman. *What is Postman?* Accessed: May 1, 2024. URL: https://www.postman.com/product/what-is-postman/.

[81] *Why did we build Visual Studio Code?* Accessed: May 1, 2024. URL: https://code.visualstudio.com/docs/editor/whyvscode.

[82] *About GitHub Copilot*. Accessed: May 1, 2024. URL: https://docs.github.com/en/copilot/about-github-copilot.

[83] *Run JavaScript Everywhere*. Accessed: May 1, 2024. URL: https://nodejs.org/en.

[84] *Node.js NPM*. Accessed: May 1, 2024. URL: https://www.w3schools.com/nodejs/nodejs_npm.asp#:~:text=What%20is%20NPM%3F,to%20run%20on%20your%20computer!.

[85] StrongLoop IBM. *Fast, unopinionated, minimalist web framework for Node.js*. Accessed: May 1, 2024. URL: https://expressjs.com/.

[86] *A JavaScript library for building user interfaces*. Accessed: May 1, 2024. URL: https://legacy.reactjs.org/.

[87] React-Bootstrap Team. *React Bootstrap*. Accessed: May 4, 2024. 2024. URL: https://react-bootstrap.netlify.app/.

[88] *What is react-router-dom ?* Accessed: May 1, 2024. 2024. URL: https://www.geeksforgeeks.org/what-is-react-router-dom/.

[89] *Bootstrap Icons*. Accessed: May 1, 2024. URL: https://icons.getbootstrap.com/.

[90] *What is Axios?* Accessed: May 1, 2024. URL: https://axios-http.com/docs/intro.

[91] *What is JavaScript?* Accessed: May 1, 2024. URL: https://www.w3schools.com/whatis/whatis_js.asp.

[92] *Introduction to CSS*. Accessed: May 1, 2024. URL: https://dhis2-app-course.ifi.uio.no/learn/essential-front-end/css/introduction/.

[93] Google. *Google Cloud console*. Accessed: May 1, 2024. URL: https://cloud.google.com/storage/docs/cloud-console.

[94] Google. *Make your app the best it can be*. Accessed: May 1, 2024. URL: https://firebase.google.com.

[95] Google. *Firebase security rules*. Accessed: May 2, 2024. URL: https://firebase.google.com/docs/auth.

[96] Google. *Firebase security rules*. Accessed: May 1, 2024. URL: https://firebase.google.com/docs/rules.

[97] Google. *Cloud Firestore*. Accessed: May 2, 2024. URL: https://firebase.google.com/docs/firestore.

[98] Googl. *Cloud Storage for Firebase*. Accessed: May 2, 2024. URL: https://firebase.google.com/docs/storage.

[99] Github. *CI/CD explained*. Accessed: May 3, 2024. URL: https://resources.github.com/devops/ci-cd/.

[100] Coursera. *What Is GitHub and Why Should You Use It?* Accessed: May 3, 2024. 2023. URL: https://www.coursera.org/articles/what-is-git.

[101] Microsoft. *Content Moderator documentation*. Accessed: May 4, 2024. URL: https://learn.microsoft.com/en-us/azure/ai-services/content-moderator/.

[102] Google. *Extract insights from images, documents, and videos*. Accessed: May 4, 2024. URL: https://cloud.google.com/vision.

[103] T. George. *Exploratory Research | Definition, Guide, & Examples*. Accessed: Jan. 31, 2024. 2024. URL: https://www.scribbr.com/methodology/exploratory-research/.

[104] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research." In: *Journal of Management Information Systems* 24 (2007), pp. 45–77.

[105] Google. *Google Scholar*. Accessed: April 30, 2024. URL: https://scholar.google.com/.

[106] *Universitetsbiblioteket*. Accessed: April 30, 2024. URL: https://www.uia.no/bibliotek/.

[107] Allen institute for AI. *Semantic scholar*. Accessed: April 30, 2024. URL: https://www.semanticscholar.org/.

[108] Connected papers. *Connected papers*. Accessed: April 30, 2024. URL: https://www.connectedpapers.com/.

[109] OpenAI. *chatGPT*. Accessed: April 23, 2024. URL: https://chat.openai.com/.

[110] Xingwei He et al. *AnnoLLM: Making Large Language Models to Be Better Crowdsourced Annotators*. Accessed: May 10, 2024. 2024. arXiv: 2303.16854 [cs.CL]. URL: https://arxiv.org/pdf/2303.16854.

[111] Chao Xue et al. *OmniForce: On Human-Centered, Large Model Empowered and Cloud-Edge Collaborative AutoML System*. Accessed: May 10, 2024. 2023. arXiv: 2303.00501 [cs.LG]. URL: https://arxiv.org/pdf/2303.00501.

[112] Shu Yang et al. "A Web 3.0-Based Trading Platform for Data Annotation Service With Optimal Pricing." In: *IEEE Transactions on Network Science and Engineering* (2023). Accessed: May 10, 2024, pp. 1–13. DOI: 10.1109/TNSE.2023.3322817. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10274096&tag=1.

[113] Venkata Satya Sai Ajay Daliparthi et al. "ViSDM 1.0: Vision Sovereignty Data Marketplace a Decentralized Platform for Crowdsourcing Data Collection and Trading." In: *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*. GoodIT '23. Accessed: May 10, 2024. Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 374–383. ISBN: 9798400701160. DOI: 10.1145/3582515.3609556. URL: https://doi.org/10.1145/3582515.3609556.

[114] Jingru Yang et al. "A game-based framework for crowdsourced data labeling." In: *2022 ACM Conference on Fairness, Accountability, and Transparency*. Accessed: May 10, 2024. The VLDB Journal, Nov. 2020. DOI: 10.1007/s00778-020-00613-w. URL: https://doi.org/10.1007/s00778-020-00613-w.

[115] Mark Díaz et al. "CrowdWorkSheets: Accounting for Individual and Collective Identities Underlying Crowdsourced Dataset Annotation." In: *2022 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '22. Accessed: May 10, 2024. ACM, June 2022. DOI: 10.1145/3531146.3534647. URL: http://dx.doi.org/10.1145/3531146.3534647.

[116] *Open data and AI: A symbiotic relationship for progress | data.europa.eu*. Accessed: May 10, 2024. June 2023. URL: https://data.europa.eu/en/publications/datastories/open-data-and-ai-symbiotic-relationship-progress.

[117] Mike Jin et al. *Expert-Level Annotation Quality Achieved by Gamified Crowdsourcing for B-line Segmentation in Lung Ultrasound*. Accessed: May 10, 2024. 2023. arXiv: 2312.10198 [cs.CY]. URL: https://arxiv.org/ftp/arxiv/papers/2312/2312.10198.pdf.

[118] Kevin R. McKee. *Human participants in AI research: Ethics and transparency in practice*. Accessed: May 9, 2024. 2024. arXiv: 2311.01254 [cs.CY].

[119] Tim Rädsch et al. *Labeling instructions matter in biomedical image analysis*. Accessed: May 10, 2024. 2022. arXiv: 2207.09899 [cs.CV]. URL: https://arxiv.org/pdf/2207.09899.

[120] *Data Annotation in 2024: Shaping the future of Computer Vision*. Accessed: May 10, 2024. URL: https://www.basic.ai/blog-post/data-annotation-in-2024:-shaping-the-future-of-computer-vision.

[121] Jinfei Liu et al. "Dealer: an end-to-end model marketplace with differential privacy." In: *Proc. VLDB Endow.* 14.6 (Feb. 2021), pp. 957–969. ISSN: 2150-8097. DOI: 10.14778/3447689.3447700. URL: https://doi.org/10.14778/3447689.3447700.

[122] Infosys Limited. *Top 7 trends for data annotation market in 2023 | Infosys BPM*. Accessed: May 9, 2024. URL: https://www.infosysbpm.com/blogs/master-data-management/top-7-trends-for-data-annotation-market-in-2023.html.

[123] Jian Pei. "A Survey on Data Pricing: From Economics to Data Science." In: *IEEE Transactions on Knowledge and Data Engineering* 34.10 (Oct. 2022), pp. 4586–4608. ISSN: 2326-3865. DOI: 10.1109/tkde.2020.3045927. URL: http://dx.doi.org/10.1109/TKDE.2020.3045927.

[124] Lingjiao Chen et al. "Data Acquisition: A New Frontier in Data-centric AI." In: (2023). arXiv: 2311.13712 [cs.AI]. URL: https://www.semanticscholar.org/reader/8137c3bb7c1908d62addf07e0

[125] Najafabadi M.M. et al. "Deep learning applications and challenges in big data analytics." In: *Journal of Big Data* 2 (Feb. 2015). ISSN: 2196-1115. DOI: 10.1186/s40537-014-0007-7. URL: https://doi.org/10.1186/s40537-014-0007-7.

[126] Rodrigo Benenson and Vittorio Ferrari. *From colouring-in to pointillism: revisiting semantic segmentation supervision*. 2022. arXiv: 2210.14142 [cs.CV].

[127] Amy K. Heger et al. *Understanding Machine Learning Practitioners' Data Documentation Perceptions, Needs, Challenges, and Desiderata*. Accessed: May 10, 2024. 2022. arXiv: 2206.02923 [cs.HC]. URL: https://arxiv.org/pdf/2206.02923.

[128] Farzan Erlik Nowruzi et al. *How much real data do we actually need: Analyzing object detection performance using synthetic and real data*. Accessed: May 10, 2024. 2019. arXiv: 1907.07061 [cs.CV]. URL: https://arxiv.org/pdf/1907.07061.

[129] J. Solé-Casals et al. *Machine Learning Methods with Noisy, Incomplete or Small Datasets*. MDPI AG, 2021. ISBN: 9783036512887. URL: https://mdpi-res.com/bookfiles/book/3727/Machine_Learning_Methods_with_Noisy_Incomplete_or_Small_Datasets.pdf?v=1715302893.

[130] Xinyu Zhang, Vincent CS Lee, and Feng Liu. *From Data to Insights: A Comprehensive Survey on Advanced Applications in Thyroid Cancer Research*. Accessed: May 10, 2024. 2024. arXiv: 2401.03722 [cs.LG]. URL: https://arxiv.org/pdf/2401.03722.

[131] Xu Yan et al. *Forging Vision Foundation Models for Autonomous Driving: Challenges, Methodologies, and Opportunities*. Accessed: May 10, 2024. 2024. arXiv: 2401.08045 [cs.CV]. URL: https://arxiv.org/pdf/2401.08045.

[132] Wenjun Qiu, David Lie, and Lisa Austin. *Calpric: Inclusive and Fine-grain Labeling of Privacy Policies with Crowdsourcing and Active Learning*. Accessed: May 9, 2024. 2024. arXiv: 2401.08038 [cs.CL]. URL: https://arxiv.org/pdf/2401.08038.

[133] Zicun Cong et al. "Data pricing in machine learning pipelines." In: *Knowledge and Information Systems* 64 (2021), pp. 1417–1455. URL: https://api.semanticscholar.org/CorpusID:237194666.

[134] ITRex. *Machine Learning Costs: Price factors and Real-World Estimates*. Accessed: May 9, 2024. Oct. 2023. URL: https://hackernoon.com/machine-learning-costs-price-factors-and-real-world-estimates.

[135] Nestor Maslej et al. *Artificial Intelligence Index Report 2023*. 2023. arXiv: 2310.03715 [cs.AI].

[136] Amanda Napitu. *150+ Artificial Intelligence Statistics You Need to Know in 2024 8211; Who is Using It amp; How?* Accessed: May 9, 2024. Jan. 17, 2024. URL: https://www.techopedia.com/artificial-intelligence-statistics.

[137] Nengfeng Zhou et al. "Bias, Fairness and Accountability with Artificial Intelligence and Machine Learning Algorithms." In: *International Statistical Review* 90.3 (2022), pp. 468–480. DOI: https://doi.org/10.1111/insr.12492. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/insr.12492. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12492.

[138] Cloudfactory. *Creating data annotation guidelines that drive accurate ML models*. Accessed: May 11, 2024. Oct. 2023. URL: https://blog.cloudfactory.com/data-annotation-guidelines-for-accurate-ml-models.

[139] Helena Matute and Lucía Vicente. "Humans inherit artificial intelligence biases." In: *Scientific Reports* 13.15737 (2023). Accessed: May 10, 2024. DOI: 10.1038/s41598-023-42384-8. URL: https://doi.org/10.1038/s41598-023-42384-8.

[140] Chao Yu et al. "Summary of Data Label Research under Smart Finance." In: Accessed: May 10, 2024. EAI, Apr. 2023. DOI: 10.4108/eai.28-10-2022.2328436. URL: https://eudl.eu/pdf/10.4108/eai.28-10-2022.2328436.

[141] Manil Maskey. "Rethinking AI for Science: An Evolution From Data-Driven to Data-Centric Framework." In: *Perspectives of Earth and Space Scientists* 4.1 (2023). e2023CN000222 2023CN000222, e2023CN000222. DOI: https://doi.org/10.1029/2023CN000222. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023CN000222. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023CN000222.

[142] Pranav Kulkarni et al. *One Copy Is All You Need: Resource-Efficient Streaming of Medical Imaging Data at Scale*. 2023. arXiv: 2307.00438 [cs.CV].

[143] Krit Gangwanpongpun, Ekapol Chuangsuwanich, and Proadpran Punyabukkana. "Analysis of the Trade-Off in Data Quality Management for Crowdsourcing." In: *2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 2023, pp. 155–160. DOI: 10.1109/JCSSE58229.2023.10201975. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10201975&tag=1.

[144] Goran Radanovic, Boi Faltings, and Radu Jurca. "Incentives for Effort in Crowdsourcing Using the Peer Truth Serum." In: *ACM Trans. Intell. Syst. Technol.* 7.4 (Mar. 2016). ISSN: 2157-6904. DOI: 10.1145/2856102. URL: https://doi.org/10.1145/2856102.

[145] Angelina Wang et al. "REVISE: A Tool for Measuring and Mitigating Bias in Visual Datasets." In: *International Journal of Computer Vision* 130 (2020). Accessed: May 11, 2024, pp. 1790–1810. URL: https://api.semanticscholar.org/CorpusID:221112230.

[146] Brianna Richardson and Juan E. Gilbert. "A Framework for Fairness: A Systematic Review of Existing Fair AI Solutions." In: *ArXiv* abs/2112.05700 (2021). Accessed: May 11, 2024. URL: https://api.semanticscholar.org/CorpusID:245117903.

[147] Chad Boutin. *There's more to AI bias than biased data, NIST report highlights*. Accessed: May 5, 2024. Mar. 16, 2022. URL: https://www.nist.gov/news-events/news/2022/03/theres-more-ai-bias-biased-data-nist-report-highlights.

[148] Intersoft Consulting. *Art. 5 GDPR, Principles relating to processing of personal data*. Accessed: April 23, 2024. 2016. URL: https://gdpr-info.eu/art-5-gdpr/.

[149] Firebase. *Cloud Firestore Data model*. Accessed: May 11, 2024. URL: https://firebase.google.com/docs/firestore/data-model#documents.

[150] G. Ishaya, I. Rhoda, and I. Olaronke. "An Appraisal of Software Requirement Prioritization Techniques." In: *Asian Journal of Research in Computer Science* 1: 1-16 2018: Article no.AJRCOS.40763 (Apr. 2018). DOI: 10.9734/AJRCOS/2018/40763.

[151] Creative commons. *BETTER SHARING, BRIGHTER FUTURE*. Accessed: May 6, 2024. URL: https://creativecommons.org/.

[152] L. Chen, F. Xiang, and Z. Sun. "Image deduplication based on hashing and clustering in cloud storage." In: *Transactions on Internet and Information Systems* 15.4 (Apr. 2021). DOI: 10.3837/tiis.2021.04.014.