NTNU

Kunnskap for en bedre verden

Department of Computer Science (IDI)

IT2901 - Informatics Project II

Connectivity and data management for environmental monitoring

Group 7

Authors:

Name	**Email**
Siri Arnesen	siriarn@stud.ntnu.no
Johanne Burns Bergan	johannbb@stud.ntnu.no
Sander Stenbakk Ekanger	sandesek@stud.ntnu.no
Marius Bølset Gisleberg	mariubgi@stud.ntnu.no
Jacob Gullesen Hagen	jacobgh@stud.ntnu.no
Ingrid Helene Kvitnes	ingrihkv@stud.ntnu.no
Noah Lund Syrdal	noahls@stud.ntnu.no

Date

Table of Contents

List of Figures

List of Tables

1 Introduction

This report presents the development and implementation of our project, completed as part of the IT2901 Informatics Project II course at NTNU spring 2025. Conducted in collaboration with the Norwegian Institute for Nature Research (NINA), the project aims to support biodiversity research through technological solutions.

The report begins with the prestudy phase, outlining the problem definition and key requirements. It then details the system design, covering the user interface, architecture, and testing, followed by an overview of the process, including methodology, sprint diaries, and risk analysis. Finally, we discuss future work for ongoing development.

1.1 Background

Monitoring biodiversity and ecological changes is crucial for nature research. Traditional data collection relies heavily on fieldwork, which is time-consuming and constrained by seasonal and geographical limitations. To address these challenges, NINA is integrating new technologies to improve data collection and analysis. This project contributes to that effort by enhancing automated data processing and visualization, ultimately supporting more efficient and accurate biodiversity monitoring.

1.2 Motivation

The motivation is driven by the need to improve the efficiency and accuracy of ecological monitoring. The current reliance on traditional methods of data collection can be limiting, both in terms of time and resources. By adopting more modern, automated approaches, we aim to streamline the monitoring process, reducing manual workload and improving data accessibility for researchers.

Beyond the direct benefits to NINA, the project also contributes to a larger effort to advance biodiversity research, with the potential to impact global scientific communities. For our team, it offers an opportunity to apply our technical skills to a meaningful real-world problem while gaining valuable experience in developing user-centered solutions that can have a tangible impact.

1.3 Scope of the Final Product

The scope of the final product strives to be a solution for managing and visualizing environmental data collected from acoustic sensors. The primary objective is to create a system that integrates seamlessly with existing infrastructure, ensuring compatibility with current data systems and meeting NINA's specific research needs.

The focus will be on designing an intuitive and user-friendly system for data management, organization, and visualization. Key aspects include ensuring efficient presentation of environmental data, facilitating easy access and analysis. Additionally, the project will prioritize system stability, performance, and scalability, allowing for future updates. We are developing a prototype, aiming for future use by NINA researchers as a fieldwork tool.

The scope of the project has evolved throughout the project, with the original scope outlined in section 2.1.5 Findings on Prestudy. Adjustments and the final project scope are described in more detail in the section 2.3 Adjustment of Project Scope.

1.4 Stakeholders

The Customer

The customer is the Norwegian Institute for Nature Research (NINA), with Julia Wiel as our primary point of contact. She is a PhD candidate at NINA and provides guidance and clarifications on requirements. While other personnel at NINA are available for additional support if needed.

End Users

The end users of the product are researchers at NINA, who will utilize the dashboard to analyze and visualize environmental monitoring data. As this project is part of an international collaboration, parts of our code could also be used by researchers outside of NINA, contributing to a broader effort in biodiversity monitoring. By providing a reliable and efficient system, the project aims to support both national and international research communities in making informed decisions and improving ecological monitoring workflows.

The team

The team consists of seven students from BA informatics, responsible for extending an existing system to better support NINA needs.

Other developers

There are other developers involved in the project that contribute to its development. We have direct contact with Julian Evans, the lead developer of the ARISE project, which our work builds upon. Dr. Evans, currently based at the University of Amsterdam, serves as a key resource for our team, providing guidance on the existing system and ensuring a smooth transition for further development. Future developers will be responsible for maintaining and expanding the system as research needs to evolve. To make this easier, we are focusing on clear code structure, modular design, and good documentation. By building a flexible and scalable system, we aim to create a solid foundation that future teams can improve on.

The supervisor

The supervisor for this project is Claudia Maria Cutrupi. Her primary role is to support the team's process, providing guidance, and ensuring that the project stays on track throughout its development.

1.5 Project Timeline

Figure 1 provides an overview of the project's timeline, outlining key phases from prestudy to reflection. The timeline is divided into distinct stages, where each stage highlights the tasks performed. The figure shows the current state of the project, but it will be updated as the work progresses.

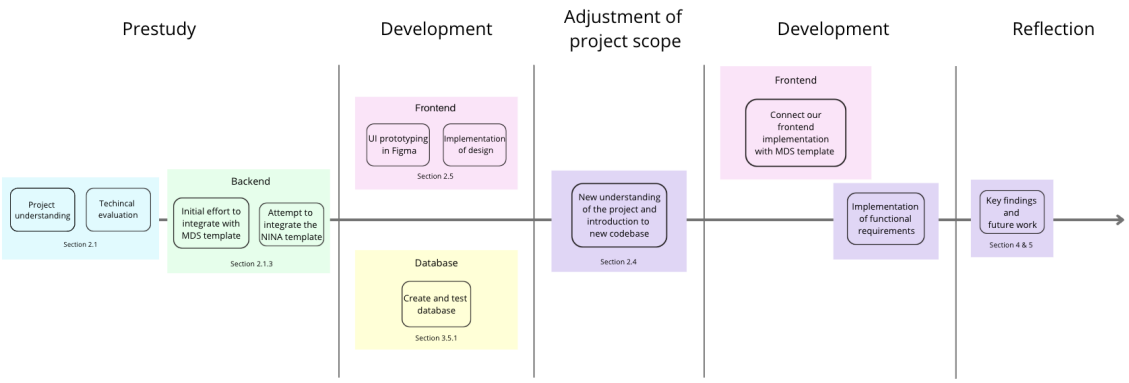

Figure 1: Project Timeline

2 The Product

2.1 Prestudy

To ensure a solid foundation for the project, we conducted a prestudy before starting the development phase. This work focused on understanding the project and its purpose, reviewing similar solutions to gain insight and inspiration, and evaluating relevant technologies to choose the best fit for our needs. The prestudy process included several meetings with NINA over a two-week period, where we discussed their expectations, existing challenges, and how our solution could support their work. These meetings were crucial to ensure that we and NINA had a shared understanding of the goals and direction of the project.

2.1.1 Understanding of the problem

In order to fully understand the project NINA assigned to us, it was crucial that we also gained a clear understanding of the underlying problem NINA is facing and how this project could help address it. During our initial meeting with the customer contact, she presented a few concrete scenarios to illustrate how their work processes could become more efficient and how specific challenges in the field could be handled in a better way going forward.

Scenario 1: Automating Species Monitoring in Remote Areas

A team of ecologists is tasked with monitoring bird populations in a vast forested region in northern Norway. Traditionally, this would require field teams to visit multiple sites, conduct point counts, and manually record species observations - a time-consuming process limited to specific times of year and weather conditions.

Instead, the team places small, weather-resistant acoustic sensors throughout the area. These sensors continuously record the soundscape, capturing bird calls, insect activity, and even amphibian vocalizations. Instead of spending weeks in the field, researchers receive large datasets automatically processed by machine learning algorithms that identify species and estimate their abundance. The team can now track changes in biodiversity throughout the year, even in areas that are difficult or impossible to access during winter. [2]

Scenario 2: Detecting and Responding to Human Disturbance

In a national park, conservationists are concerned that increased snowmobile tourism can disturb wildlife, particularly birds during their breeding season. Rather than relying on occasional field observations, they install a sensor system along popular snowmobile routes and in nearby undisturbed areas.

The system continuously records both natural sounds and human-generated noise. Using automated analysis, researchers can detect when snowmobiles pass by, and whether bird vocal activity changes immediately after. Over time, the data reveal clear patterns, showing that certain species reduce their singing near busy trails. Armed with this information, park managers can make informed decisions about zoning or restricting snowmobile access during critical periods, all without requiring researchers to be present in the field every day. [4]

PAM- Passive Acoustic Monitoring

In these scenarios, a sensor-based system was presented as a potential solution, known as Passive Acoustic Monitoring (PAM). Our role is to help implement this system for NINA, so it is essential for us to understand what it involves. PAM is a non-intrusive and cost-effective method where small microphones are placed in natural areas to capture environmental sounds. These recordings are then analyzed using AI to detect species, monitor breeding periods, estimate populations, and uncover ecological patterns. [15]

Through these scenarios, we saw how this system could reduce both the workload for NINA's field researchers and the disturbance to nature by enabling remote data collection. With fewer field

visits, researchers can spend more time analyzing data, leading to deeper insights and better use of the collected information.

2.1.2 Understanding our scope in a bigger project

Our project is part of the larger TABMON (Targeted Acoustic Biodiversity Monitoring Network) initiative, which is coordinated by NINA. TABMON is focused on improving biodiversity monitoring in Europe using advanced acoustic technologies. The goal of this project is to gather detailed information on species presence, population sizes, breeding periods, and environmental changes, which are essential for both conservation efforts and understanding human impacts on ecosystems.

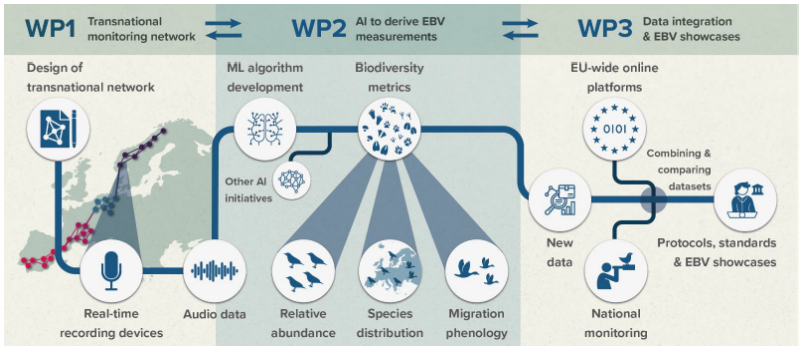

Figure 2: TABMON work packages. Source: [17]

As part of TABMON, our role falls between WP1 and WP2, where WP1 deals with hardware development, such as the placement and maintenance of microphones in the field, and WP2 focuses on the analysis and presentation of the data. [17] Our responsibility is focused on developing the software that will handle the data collected through PAM, which is a core component of the TABMON initiative.

2.1.3 Exploring similar product

As NINA collaborates with several partners in the TABMON project, including the University of Amsterdam [17], we learned about a project called the ARISE-MDS sensor portal, led by Dr. Evans, which focuses on goals similar to ours. [7] NINA suggested we explore this project as a potential foundation, either to build upon or take inspiration from.

We saw several reasons for both continuing with ARISE or starting from scratch. On one hand, building upon the project could save time, as much of the required functionality might already be implemented, allowing us to focus on additional features for NINA. Moreover, working with an existing project increases the likelihood that our work would be adopted, as it could be more easily integrated with the existing system. Additionally, this approach provided valuable learning opportunities, since working with a pre-existing project is a common real-world scenario that none of us had experienced before.

On the other hand, it is challenging to adapt to another person's ongoing work, especially when the code utilized older technologies that we were unfamiliar with. Additionally, the code was poorly documented, making it difficult to understand and posing challenges for future developers working on the project. NINA also provided a Python template for the backend, which could be a cleaner starting point. [16]

After exploring the ARISE project and encountering difficulties running it, we discussed the situation with the customer and ultimately decided to start fresh with our own project, basing it on the template provided by NINA.

2.1.4 Selection of product technology

At our very first group meeting, we mapped out each member's skills, preferences, and familiarity with relevant technologies and tools. Since we had taken many of the same courses, most of the group members had experience with similar technologies.

For the frontend, there was broad consensus that React was one of the most relevant technologies on the market. Furthermore, most of the group members had used React in previous projects and were eager to apply it again, provided that it aligned with the client's preferences.

The preferences of the backend technology were more varied, and no specific framework or language stood out. We were therefore open to the client's input and felt that we needed a clearer understanding of the project requirements before making a final decision.

During our initial customer meeting with NINA, we asked about their technology preferences. NINA mentioned that they frequently use Python with Django in their existing systems and encouraged us to consider it for this project. After discussing this within the group, we concluded that this would be a valuable opportunity to work with a technology some of us were less experienced with while also leveraging Python, which all group members were familiar with from previous coursework.

Based on this, we proposed using React with TypeScript for the frontend and Python with Django for the backend. These choices were driven by their strong industry adoption, extensive community support, and compatibility with NINA's existing infrastructure. When we presented this proposal to NINA, they were highly supportive, emphasizing the importance of seamless integration with their current systems.

2.1.5 Findings on prestudy

We established several key points for the project during the prestudy phase. The technology stack was defined, primarily based on customer preferences as well as our own experience. More information about the technology stack can be found in section 2.4 Technologies. At the end of the prestudy phase, we also worked together with the customer to define the project requirements, further details on this are provided in section 2.2.1 Functional Requirements. Additionally, based on internal team reflections and with approval from the customer, we decided to initiate the project from scratch.

Defining the scope

Through the prestudy phase, we mapped out the overall task and, in agreement with the customer, established a clear project scope. The objective of this project is to develop a dashboard for managing and visualizing data collected from acoustic sensors used in NINA's biodiversity monitoring projects. These sensors capture audio data in natural environments, which is then processed to support biodiversity research. The dashboard will provide researchers with an intuitive tool to access, browse, and manage both device information and collected audio files.

As part of a larger initiative, the project focuses not only on delivering functionality tailored to current needs but also on ensuring that the solution remains scalable and easy to further develop. The scope covers a first version where core functionality related to data handling and visualization will be prioritized, based on requirements identified and refined together with NINA. The goal is to provide a solid and flexible tool that supports NINA's ongoing work with biodiversity monitoring, while leaving room for future adaptations and extensions.

2.1.6 Reflection on prestudy

Even though we were eager to start developing, we are pleased with the decision to dedicate the first two weeks to a prestudy phase. We see the value in spending time thoroughly understanding NINA's challenges, as well as gaining insight into parts of the project that fall outside our immediate

scope.

This broader understanding proved particularly useful later when new requirements emerged, which significantly impacted how we viewed and defined our scope, more on this in section 2.3 Adjustment to Project Scope. Having a solid foundation allowed us to faster reorganize as a team and respond to the new requirements more effectively. We also feel that it gave us a better basis for discussing and evaluating how to design the best possible solution for NINA.

2.2 Requirements

To ensure that the product met customer expectations, a comprehensive set of requirements was formulated. NINA had a set of predefined requirements, which they expanded upon at project initiation. From this we created a set of functional and non-functional requirements. Then we organized collaborative sessions with the customer to further refine and prioritize these elements, ensuring that the product satisfies both operational and functional targets.

2.2.1 Functional Requirements

User requirements and system requirements

Functional requirements describe both the system's goals and the user's needs [9]. When we received the requirement list from NINA, it included several system-specific demands that were not directly related to the user. To ensure a comprehensive understanding of our application, we decided to separate the requirements into user requirements and system requirements. This approach made customer meetings more efficient, as it allowed NINA to clearly see that we were addressing their key concerns. To organize these requirements, we categorized user requirements as FUR (Functional User Requirements) and system requirements as FSR (Functional System Requirements).

Since we categorize functional requirements into user-specific and system-specific requirements, we determined that including user stories would be redundant. A user story is a short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system [21], but since our approach already distinguishes between user and system needs in a structured manner, maintaining separate user stories would introduce unnecessary overlap.

Prioritization

To clearly outline tasks and ensure shared understanding, each functional requirement was assigned a priority level, categorized as "Very low" "Low", "Medium", "High" and "Very High", to direct attention and resources to the most crucial features first. The list was reorganized with the customer to display the highest priority items at the top, with requirements of the same category further compared to establish their order of importance.

Explanation of the table

Table 1 lists all the functional requirements for the product. The main focus is to view devices in a table display and access basic information about audio files, hence the "Very High" priority placed on these requirements, in addition to uploading audio data, which is essential for our product. Getting an overview and retrieving specific pieces of information about devices and audio files is the primary objective of the product, which is why every functional requirement relating to this is "High" or above. "Medium" priority items are less critical and often stand-alone features. Requirements receiving 'low' priority mainly focus on analyzing audio data and modifying the data. "Very Low" priority includes features deprioritized in agreement with the client due to lack of access or necessary resources, and are not planned for completion.

ID	Description	Priority
FSR-1	The system shall support uploading audio data, linking them to the correct deployment.	Very High
FUR-8	Users shall be able to view basic info about audio files, like sample frequency, length of the file, and size of the file.	Very High
FUR-10	Users shall be able to view a lookup table for devices, with information about the last uploaded file.	Very High
FUR-11	Users shall be able to view a lookup table for browsing audio files retrieved from each device, with additional filtering options.	High
FUR-12	Users shall be able to see the total size (GB) of all audio files retrieved from each device.	High
FUR-9	Users shall be able to view a map displaying device locations, with information about site name, country and last uploaded file. Device-markers shall be green for active devices and red for inactive ones.	Medium
FUR-19	Users shall be able to upload information about devices through a form.	Medium
FUR-7	Users shall be able to download and upload audio files.	Medium
FUR-16	Users shall be able to listen to an audio file.	Medium
FUR-14	Users shall be able to filter audio files on date and time.	Medium
FUR-3	Users shall be able to filter deployments based on country.	Low
FSR-6	The system shall support integration with external cloud storage solutions.	Low
FSR-2	The system shall produce observation files, linking them to the correct media entries.	Low
FUR-15	Users shall be able to see if an audio file is associated with observations, and download the observation table.	Low
FUR-13	Users shall be able to view the sound quality of the audio files.	Low
FUR-18	Users shall be able to log in to the system.	Low
FSR-5	The system shall connect to the NIRD database.	Very low
FSR-4	The system shall allow editing of audio files, including silencing human voices and logging modifications.	Very low

Table 1: Overview of Functional Requirements

2.2.2 Non-Functional Requirements

Non-functional requirements (NFRs) define the qualities and constraints of a system beyond the basic, functional requirements. According to ISO/IEC 25010, the quality model defines nine key

characteristics that reflect how well a system meets stakeholders' needs [10]. In the early phases of the project, we encountered some NFRs from the customer. After discussing this further, we agreed on prioritizing the NFRs and identified the most critical areas to focus on. The table below addresses the requirements we have chosen to focus on and emphasize as most important, providing a concise description of our approach for each.

Area	**Description**	**Priority**
Functional Suitability	Functional Suitability assesses whether the system's set of features is complete, relevant, and operates as intended. The customer emphasized the importance of delivering a system in which every feature should be complete and fully operational. Instead of initiating numerous features that might remain unfinished, our approach should focus on developing a limited set of functionalities to a high standard. To achieve this, we invest time in prioritizing our work—completing the highest-priority tasks first and ensuring they function correctly before moving on. Each feature should be well-designed and intuitive, aligning closely with the customer's specific needs. This approach should guarantee that each component meets the defined requirements and consistently delivers real value to the end users.	High
Performance Efficiency	The system should perform efficiently, making sure that pages load quickly and data is processed without unnecessary delays. This is important because researchers should be able to utilize our dashboard out in the field without encountering prolonged delay. For example, when a user navigates to a new page or when data is being collected, these actions should happen smoothly and promptly. While it is not critical to achieve performance at the millisecond level, the system should still be capable of processing data in near-real time so that users don't experience any noticeable lag. To ensure this efficiency the system should make use of strategies like optimized database queries and caching frequently used data to speed up retrieval.	Low
Compatibility	The system we develop should integrate seamlessly with NINA's existing infrastructure and tools, as well as with the broader TABMON initiative. This means that our solution should be compatible with the current hardware, such as acoustic sensors, and the software already in use. To achieve this, the system should be designed using standardized interfaces and data formats that align with the existing environment, ensuring smooth communication between our solution, other components, and external systems.	Medium
Maintainability	Given that the system will be part of a larger initiative and further developed by others in the future, maintainability should be prioritized. It's essential that the system we build is modular and that the components we develop can be reused without significant changes. In this context, our work should be reusable and serve as an asset for future developers. Therefore, we should make as few changes as possible to the database and leverage existing logic—for example, when fetching data. We should emphasize that what we create can be used later and will deliver real value.	Medium

Area	Description	Priority
Flexibility	While flexibility is not emphasized, installability was important for the customer. Docker containerization ensures quick, consistent deployment, and modular architecture allows scaling if needed. Standard protocols and clear documentation ensure smooth integration and future updates. Actual deployment and operation in production or in the field are the customer's responsibility and outside our scope.	Low

Table 2: Overview of Non-Functional Requirements

2.3 Adjustment to Project Scope

This section explains the adjustment to the project scope following NINA's new requirement to build upon Dr. Evans' ARISE project, including the impact on the project's direction and requirements.

2.3.1 Introduction of New Requirement

A month into the project, after our first sprint, NINA introduced a new requirement: our work had to build upon Dr. Evans' ARISE project. Following internal discussions among NINA researchers involved in the TABMON project, they concluded that integrating with ARISE would provide the most value.

Initially, we attempted to set up ARISE, but encountered persistent errors that could not be resolved. Dr. Evans later confirmed that the available version contained a critical bug, making it impossible to run at the time. Although the customer mentioned ARISE as a relevant foundation, we did not interpret this as a requirement and believed we could develop an equally effective solution from scratch. Our impression is that NINA's preference for building upon ARISE became clearer over time through internal discussions rather than being a firm initial directive.

This decision was ultimately based on the fact that ARISE already contained much of the functionality and logic required to meet several of NINA's specifications. Additionally, it aligned with their goal of centralizing efforts within the TABMON project. By building on ARISE, our work would not only support NINA's research but also enhance Dr. Evans' project, which is designed to be easily adopted by other institutions using their own sensors and cameras. This integration could potentially expand its reach beyond NINA researchers, benefiting a wider audience.

The ARISE project also includes additional functionality not initially requested by the customer, offering the flexibility to utilize these features in the future if needed. In retrospect, while we do not believe we misjudged the customer's expectations, we recognize that earlier discussions about ARISE and direct engagement with Dr. Evans could have clarified its significance sooner. Seeking more structured feedback when deciding to start from scratch might have also helped align expectations earlier in the process.

The process of adapting to this new requirement, along with our reflections on the transition, is further detailed in 3.7.2.

2.3.2 Aligning Requirements

Functional Requirements

Since Dr. Evans' project already included some of the functionality outlined in NINA's requirements, we needed to ensure that the functional requirements remained unchanged from our initial agreement. After discussions with the customer, we confirmed that the original functional requirements would stand, allowing us to focus on implementing them thoroughly. Any additional features would only be considered if time allowed.

We thoroughly reviewed the project existing status in collaboration with Dr. Evans. Some features were already in place, while others were incomplete or missing. The following functionality was available at the time of takeover:

- **FR-1 (API Upload & External Storage Import)**: Supported, but frontend-based uploads were missing.
- **FR-11 (Lookup Table for Audio Files)**: Implemented with basic functionality.
- **FR-12 (Total Size of Audio Files)**: Displayed, but required minor fixes.
- **FR-9 (Map Visualization)**: Partially available, but unfinished.

Additional information on the status of the remaining functional requirements at takeover can be found in Table X in the appendix.

Non-Functional Requirements

The prioritization of the non-functional requirements remained largely unchanged. However, the decision to build upon Dr. Evans' project made compatibility even more critical. To ensure that what we develop can be integrated with his project even after our work is completed, it is crucial to focus on seamless integration with the existing infrastructure and make it easy for future developers to adopt and extend our solution.

2.3.3 The Final Project Scope

The scope of this project is to extend and integrate with the existing ARISE system to enhance NINA's biodiversity monitoring capabilities. The primary objective is to develop a custom frontend that provides an intuitive and efficient dashboard for managing and visualizing environmental data collected from acoustic sensors. This solution will ensure compatibility with ARISE's backend while aligning with NINA's specific research needs.

The project scope includes:

- Designing and implementing a standalone frontend that integrates with ARISE, ensuring usability and efficiency for researchers.
- Developing functionality for handling, organizing, and presenting environmental monitoring data in a clear and structured manner.
- Ensuring seamless communication between the new frontend and ARISE's backend while maintaining system stability and performance.
- Structuring the solution to support future adaptations and extensions.

2.3.4 Out of scope

tabell hvor vi lister opp ting som er out of scope, vise til future work om mer begrunnelse og beskrivelse av det videre arbeidet.

2.4 Technologies

When we transitioned to working on Dr. Evans' project, our choice of technologies was influenced by his previous work and the tools required for further development. Below is an overview of the key technologies we utilized throughout our development process.

Docker

Docker is utilized to containerize the application, ensuring a standardized development and deployment environment. By encapsulating dependencies within containers, Docker eliminates configuration inconsistencies between different environments. This approach allows for rapid deployment of the fullstack project with a single command, improving workflow efficiency. Containerization also enhances scalability and portability, making the system adaptable for various hosting environments [6].

Django

Django serves as the foundation for the backend, providing a high-level Python framework that emphasizes security, maintainability, and rapid development. The modular architecture of Django and the built-in features, such as authentication, security mechanisms, and request handling, allow the team to focus on the core business logic without reinventing the wheel [8].

PostgreSQL

PostgreSQL serves as the relational database management system (RDBMS), offering high performance, reliability, and support for complex queries [2]. Its ACID compliance, indexing, and optimization features ensure efficient data retrieval. Seamless integration with Django's ORM enhances data integrity and performance.

React

The frontend is built with React and TypeScript, offering a component-driven architecture that improves code reliability and maintainability [14]. React's virtual DOM and declarative UI paradigm enhance performance and user experience. Vite is used as the build tool, enabling instant hot module replacement (HMR) and efficient bundling [3]. Tailwind CSS complements this setup by providing a utility-first styling framework, reducing the need for custom CSS while ensuring a responsive and consistent design [13].

2.5 User Interface

This section presents the user interface of the initial design, design discussions during adjustment to project scope and final web design.

2.5.1 Initial design

When deciding on a user interface, we focused on the specific needs of the customer. Their previous solution was using Google Cloud's interface for viewing audio files, without the possibility of customising the views to their liking. Based on the set of requirements we received, we began working on an initial design.

First, we established the overall design direction. Using Figma, we made design drafts of the most important web pages relating to the functional requirements. At this point, the project scope was not entirely set, so investing numerous hours into designing pages that would not be needed was deemed unnecessary. During the prestudy, the customer heavily indicated that functionality was the most important element of the frontend, not the aesthetic. Therefore making an MVP of the device lookup table was the main priority.

Since the researchers at NINA are the primary users of the end product, we maintained close communication with the customer and held meetings to discuss the design. The feedback indicated

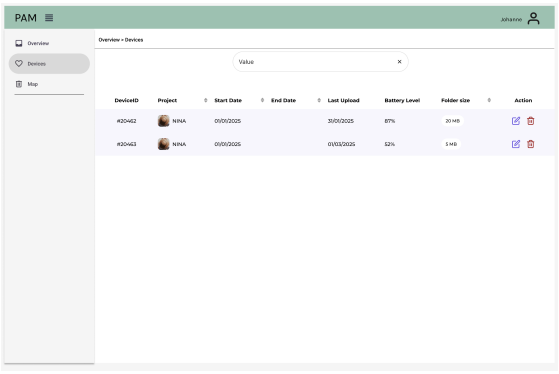

Figure 3: Figma design for device lookup table

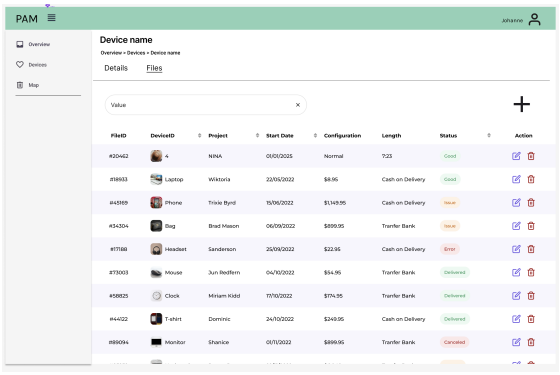

Figure 4: Figma design audio files lookup table

that our initial design aligned well with their expectations, so we used it as a foundation moving forward. To ensure consistency across the application, we designed similar functionalities to have a uniform look and feel, for example, the similarities between the device lookup table and the file browsing table.

During design presentations, we explained how the different pages were structured and how users would navigate between them. To provide a more interactive experience, we created a "paper prototype" online on figma, where we physically swapped out design sketches when the customer "clicked" on a button. This hands-on approach helped visualize the navigation flow and allowed the customer to better understand how the system would function. The feedback from these sessions confirmed that both the design and navigation worked as intended, enabling us to proceed confidently with development.

2.5.2 Design discussions during adjustment to project scope

As mentioned in section 2.4, our project scope underwent an adjustment, requiring us to decide whether to continue developing based on our initial design or to refine the existing ARISE design. Given the key differences between the two, we carefully discussed the best course of action internally. We ultimately concluded that our initial design was the better choice, as it aligned more effectively with the project requirements.

To validate this decision, we set up a meeting with the customer, where we presented our findings and reasoning. We demonstrated how our design better structured the elements most important to the customer, as outlined in the requirements list, making the interface clearer and more user-friendly. To our satisfaction, the customer agreed with our approach, allowing us to proceed with our original design direction.

2.5.3 Final web design

2.6 Architecture

In this section, we outline the architecture of our product employing Kruchten's 4+1 model [12]. This framework divides the system into several perspectives, logical, development, process, physical, and use case—that together provide a thorough understanding of the system's core functions and performance expectations. Although Kruchten's work centers on object-oriented architectures, and our approach may call for different diagrammatic representations for some perspectives, the 4+1 model still offers a solid foundation for describing our system.

2.6.1 The Logical View

The logical view illustrates how users interact with the system and how the different pages and functionalities of the system connect to each other. In our application, the user's journey typically begins on the homepage, which serves as the central hub for navigation. From there, the user can either proceed to view all available deployments or access a map displaying deployment locations, using the sidebar. Once a user opts to view all deployments, they are presented with a comprehensive list and can sort these deployments by various different attributes. It is also possible to initiate the process of adding a new deployment by opening the form. There, a user can add information about a new deployment that is being deployed, which subsequently creates a new entry in the database. Selecting a particular deployment in the table view leads to a detailed deployment view, where further information about that deployment is provided. In addition, a user can view site information and device information related to the deployment.

After accessing the details of a deployment, the user can see the audio files associated with the device on the deployment. Each audio file can be sorted by its attributes, allowing users to quickly find the files they need. They can also navigate to a specific audio file, where more information is displayed. When viewing an audio file, the user can check the audio quality of said audio file. This also generates observations, which are sound snippets which the system detects as interesting.

In 2.6.3, we show how the logical view is built up with components.

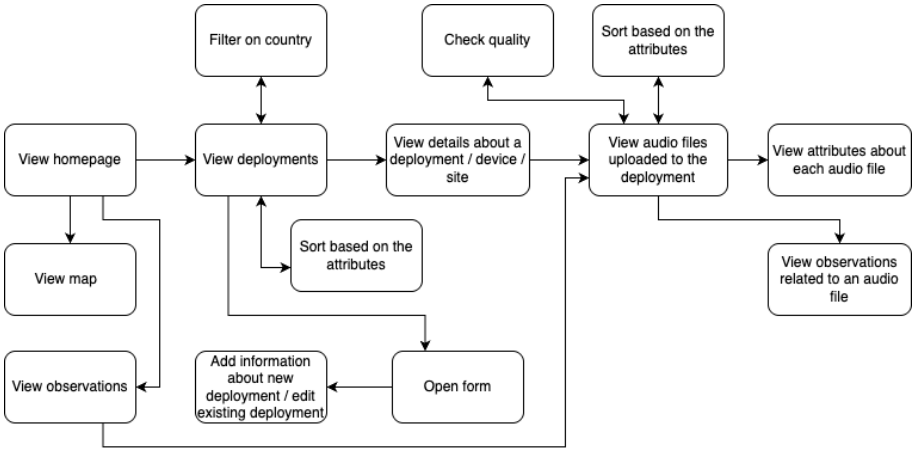

Figure 5: System flow diagram

In the figure above, the relationships between these pages and actions are clearly depicted. It shows the primary routes the user can take, moving from the homepage to view devices or the map, as well as managing and viewing audio files associated with each device. The diagram emphasizes how the system logically supports different use cases: from simple browsing of existing devices and their files, to adding new entries and sorting them according to relevant attributes.

2.6.2 The Process View

The process view shows *how* the logical view operates in the physical system. This process view illustrates non-functional requirements like performance and availability [12].

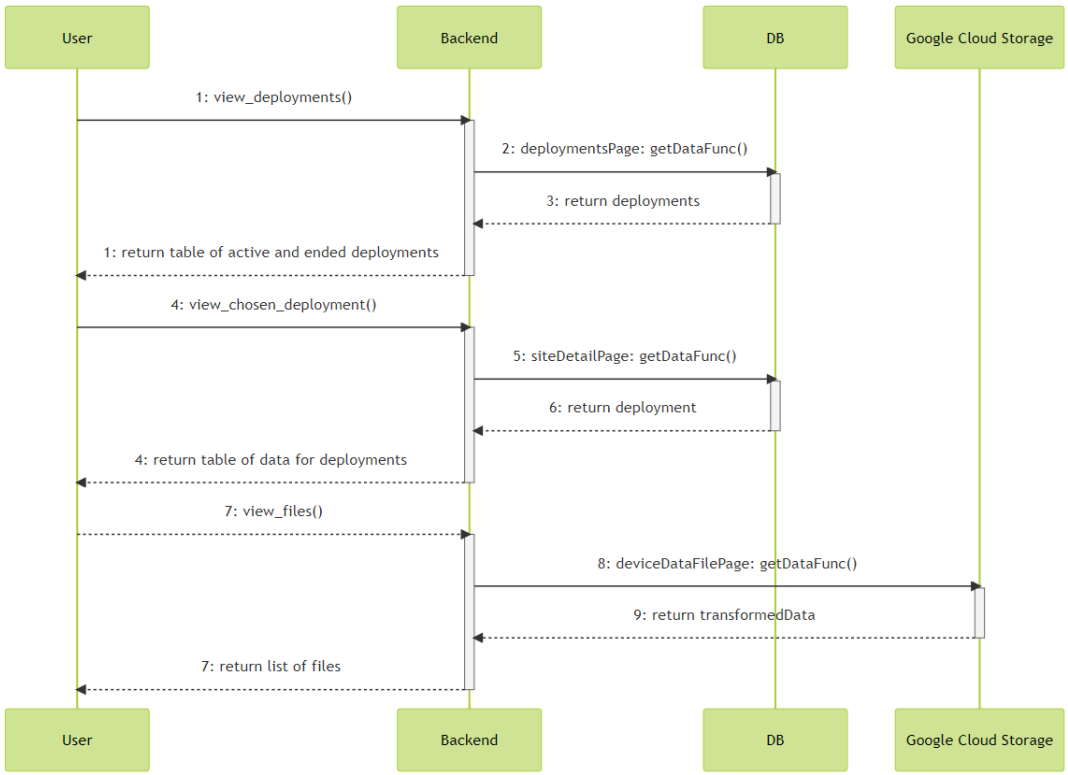

Figure 6: Sequence diagram

The sequence diagram in Figure 6 shows the abstracted call sequence for when a user first wants to see all available deployments. The click from the user fires a call to the backend to show all registered deployments. The backend then queries the database to get the deployments. The database returns a list of deployments stored in the database to the backend. The backend returns a table of deployments to the user interface for the user to see.

The next call from the user, shows the abstracted call sequence after a click where the user wants to see the related data to a chosen deployment. This fires a call to the backend to get data for the chosen deployment. The backend queries the database for the related data, which returns the data of the deployment to the backend. The backend returns the information to the user interface in the frontend.

When the user has chosen which deployment they want to display, they can click a button to access all the files in this deployment. This fires a call to the backend to get all the files of this deployment. The backend then sends a HTTP request to the Cloud Storage to get the files associated with the chosen device. These files are sent from the Cloud to the backend, before the backend sends a list of the files back to the user.

2.6.3 The Development View

In this project, we built upon a pre-existing backend infrastructure, allowing us to concentrate primarily on the design and implementation of the frontend structure. The frontend was developed using React, a framework that encourages a modular approach by dividing the user interface into independent, reusable components. This method enhances both the maintainability and scalability of the codebase, ensuring that future updates or expansions can be managed efficiently. During development, we also made incremental updates to the backend; however, the primary focus of this section is on the frontend architecture. As such, the component diagram provided illustrates only the structure and relationships of the frontend components, without delving into backend modifications. This approach offers a clear overview of how the application's user interface is

organized and how different components interact to deliver a cohesive user experience.

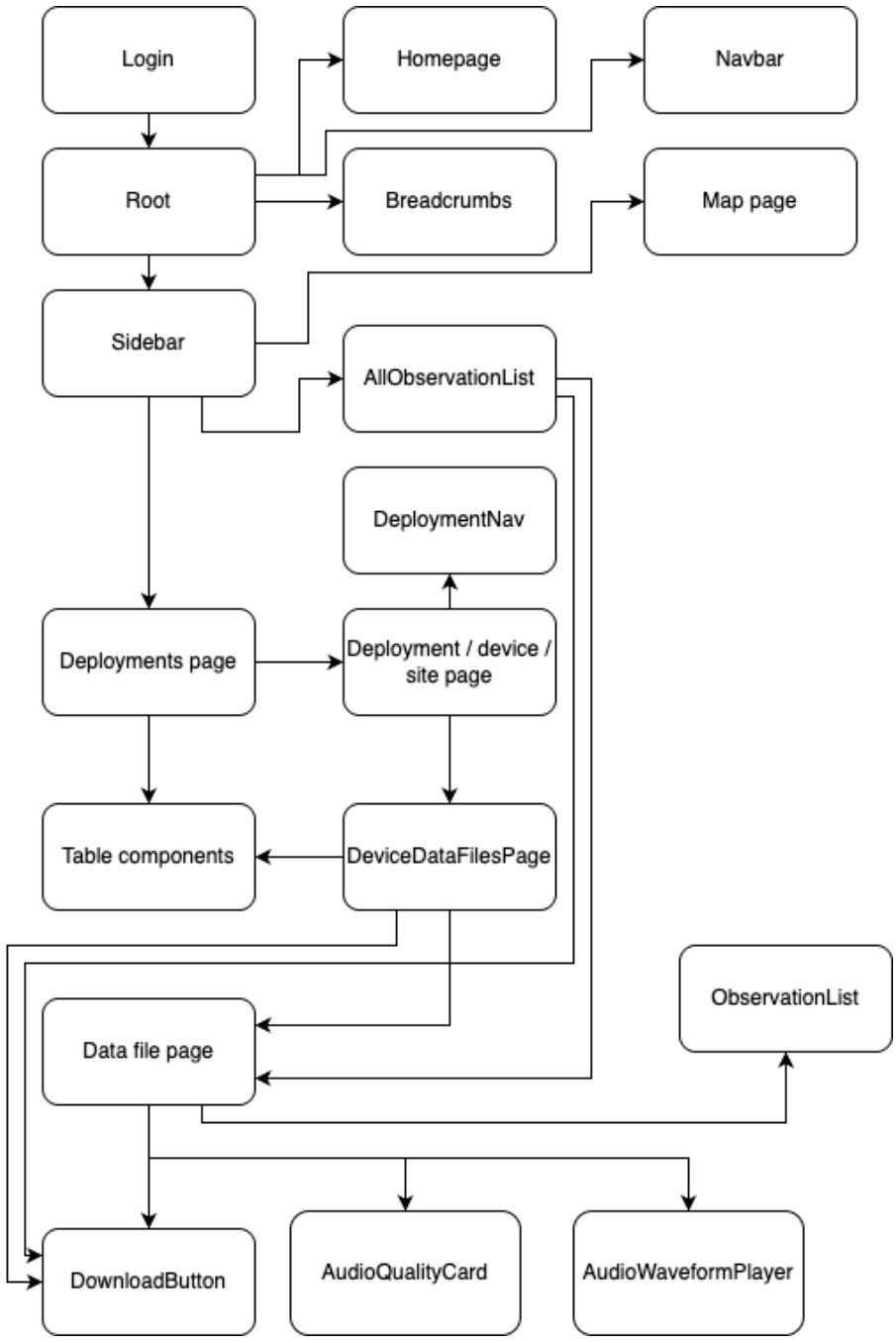

Figure 7: Component diagram

2.6.4 The Physical View

The physical view showcases the deployment of hardware and infrastructure of the project. Figure 7 (Physical view) shows in diagram form that the project will be hosted on a server that runs a docker daemon. The project runs a NGNIX reverse proxy that allows communication between the users browser and backend and frontend servers, and the Django server (backend) is getting information from a Postgis setup. The redis caching image feeds the database whatever relevant information it has cached. The diagram also shows what ports the different components run on, and that the user browser can connect via the port 8080.

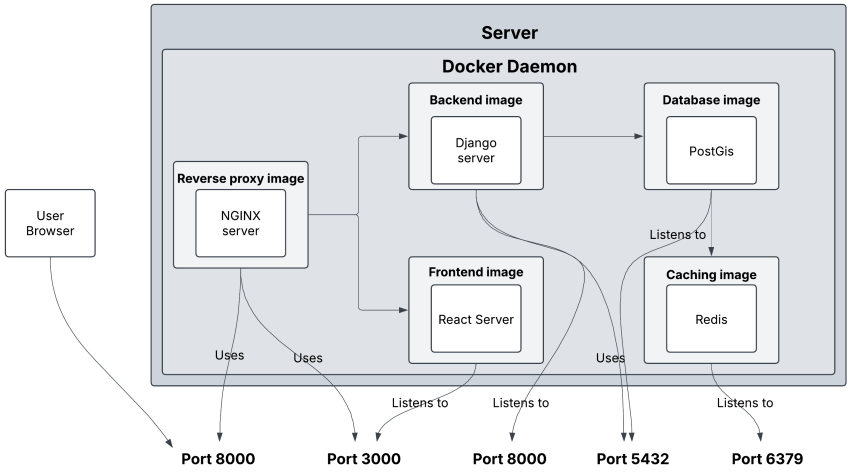

Figure 8: Physical view - How the project runs in it's environment

2.7 Testing

Testing is an essential part of software development, ensuring that the application functions as intended, remains stable, and delivers a high-quality user experience. By implementing a combination of unit tests, manual tests, end-to-end (E2E) testing, and automated code quality checks, our goal is to identify and resolve errors early in the development cycle. This section outlines our testing strategy and priorities.

From the outset of the project, our customer has emphasized that quality outweighs quantity. Rather than aiming to deliver a broad array of features, our priority is to ensure that every function we develop is complete, reliable, and performs flawlessly. This customer perspective drives our testing strategy, which is designed not only to validate that the application functions as intended but also to guarantee that each critical feature is thoroughly tested and robust when when we are finished.

To achieve this, we utilize different testing methodologies as explained below.

2.7.1 Unit Testing

Unit testing involves testing individual functions or components in isolation to validate their accuracy. We have implemented unit tests on the backend, frontend, and database to ensure that main functions behaves as expected.

Backend

Unit tests ensure the correctness of core functionality by verifying critical API endpoints, database interactions, and business logic. Instead of testing every minor detail, we focused on the most important components that directly impacted the system's stability and correctness. By testing these elements in isolation, we could catch logical errors early before they propagate into larger issues. This approach ensures that essential workflows function as expected while keeping the testing effort efficient and maintainable.

Frontend

In line with the client's preference for maintaining a high-level testing approach, we have implemented basic component tests in the frontend to ensure that the most important parts of the user interface function as expected. Using React Testing Library and Vitest, we tested key components with a focus on correct data rendering and the handling of typical user interactions and state changes.

Database

Unit tests for the database check that queries execute correctly and return the expected results. These tests are crucial for ensuring data integrity and consistency across the application.

Unit testing is performed using industry-standard frameworks, including Vitest for frontend, Pytest for Python, and JUnit for Java. These tests provide fast feedback during development and prevent regressions when changes are introduced.

2.7.2 Manual Testing - User Tests

User testing involves manually interacting with the product to evaluate usability, functionality, and user experience. By conducting user tests, we aim to identify issues that may not be apparent through automated tests. Testers perform key workflows and provide feedback on design, navigation, and overall ease of use.

User testing sessions involve structured test scenarios, in which users attempt specific tasks while their interactions and feedback are recorded. This process helps uncover usability bottlenecks and informs design improvements.

2.7.3 End-to-End (E2E) Testing

E2E testing ensures that the entire application works as expected in an integrated environment. These tests simulate real-world user interactions by testing workflows from start to finish, covering:

- Navigation across the application.
- User authentication
- Data retrieval and updates
- UI interactions.

E2E tests are automated to reduce manual effort and increase reliability. There are many frameworks available to implement E2E tests, but we are using Cypress.

2.7.4 Automated Testing - Code Quality

Maintaining high code quality is essential for long-term maintainability and collaboration. Automated testing tools ensure code consistency and adherence to best practices.

- Prettier: Ensures consistent code formatting.
- ESLint: Identifies and fixes potential errors in JavaScript/TypeScript code.

2.7.5 Test Coverage

Test coverage measures the percentage of code executed during tests. High test coverage indicates that most functionalities are verified by tests, reducing the risk of undetected bugs. We track:

- Line coverage: Percentage of lines executed by tests.
- Branch coverage: Coverage of different execution paths in conditional statements.
- Function coverage: Percentage of functions tested. Although achieving 100 % coverage is not always practical, we aim for a balance between coverage and effort, focusing on critical components.

2.7.6 Reasons for Testing

- Bug Detection: Early identification and resolution of software defects.
- Regression Prevention: Ensures new changes do not break existing functionality.
- Code Reliability: Increases confidence in the correctness of the application.
- Improved Design: Encourages modular, maintainable, and testable code.
- Enhanced User Experience: Detects usability issues before they affect users.

2.7.7 Testing Prioritization

While testing is crucial, certain aspects are deprioritized to optimize resource allocation:

- Security Testing: While important, security audits will be conducted separately rather than within standard test cycles.
- Performance Testing: Load and stress testing are not primary concerns during early development but may be explored in later stages.
- Cross-Browser Testing: Focus is placed on the most commonly used browsers initially, with further testing planned for broader compatibility.
- By structuring our testing strategy effectively, we ensure that our application remains functional, reliable, and user-friendly throughout its lifecycle.

2.7.8 Reflection on Testing

To be continued.

2.8 Quality of Documentation

2.8.1 Challenges with Limited Documentation in the ARISE Project

When we forked Dr. Evans' ARISE project, we encountered significant challenges due to the lack of documentation. The code contained few comments explaining its logic, making it difficult to understand and build upon. Additionally, the README.md file was minimal, providing only instructions on how to run the project without detailing its structure or key components. As a team tasked with extending this project, we found the absence of comprehensive documentation particularly challenging. We lacked guidance on how different components were connected, which aspects were intended for customization, and how we could effectively adapt the project for NINA's specific needs.

2.8.2 Our Documentation Approach

The difficulties we faced in working with an under-documented codebase slowed our progress considerably. While Dr. Evans was available to clarify certain aspects, future developers may not have the same level of access to support. Since we will no longer be involved with the project after completing our bachelor's degree, ensuring clear documentation became a priority for both us and the customer. To address this, we focused on creating thorough documentation.

To ensure high-quality documentation that is both user-friendly and maintainable for future developers, we have prioritized thorough documentation of all newly written code. This has been achieved through:

Frontend README: blablabla

Backend README: blablabla

Database README: blablabla

Test README: blablabla

In-Code Comments: blablabla

It is important to note that while we have carefully documented all code we have written, we have not revised or improved Dr. Evans' existing documentation. Given the scope of his project, this would have been too time-consuming within our available timeframe. Additionally, documenting another developer's code without full insight into its design and functionality presents significant challenges.

2.9 State of the Product at Project Completion

At the conclusion of the project, we successfully transitioned from our initial independent development effort to fully integrating with the overtaken project. Despite the significant mid-project change, we adapted quickly and delivered a robust, functional system that addressed nearly all the agreed-upon requirements.

Initially, we had developed our own solution, including a fully structured and operational database, a backend featuring map plotting for device locations, audio file handling, and communication logic with the frontend, as well as a complete frontend that supported most of the Very High, High, and Medium Priority requirements. When the project direction changed, we abandoned our original backend and database in favor of the overtaken project's architecture, but we continued to work with and refine our own frontend.

Throughout the transition, the team encountered several challenges, including adapting our frontend to an unfamiliar backend structure, resolving inconsistencies in database schemas, and reworking previously planned functionalities to fit the new system. Through close collaboration, regular communication, and iterative problem solving, we successfully overcame these obstacles and maintained steady development progress.

Upon taking over the new project, we thoroughly reviewed its existing status in collaboration with the original developer. Some features were already in place, while others were incomplete or missing. The following functionality was available at the time of takeover:

- **FR-1 (API Upload & External Storage Import)**: Supported, but frontend-based uploads were missing.
- **FR-11 (Lookup Table for Audio Files)**: Implemented with basic functionality.
- **FR-12 (Total Size of Audio Files)**: Displayed, but required minor fixes.
- **FR-9 (Map Visualization)**: Partially available, but unfinished.

Several key features were solved or adapted during development:

- **FR-5 (NIRD Database Integration)**: Not completed due to lack of access to the NIRD database.
- **FR-6 (Cloud Storage Integration)**: Implemented during the project.
- **FR-7 (Download Audio Files)**: Extended to allow bulk downloads.
- **FR-8 (Basic Audio File Metadata)**: Metadata extraction implemented.
- **FR-16 (Listening to Audio Files)**: Playback functionality added to the frontend.

- **FR-15 (Audio Files Connected to Deployments)**: Observation linkage completed.
- **FR-2 (Producing Observations for Audio Files)**: Completed based on uploaded audio files.
- **FR-3 (Manual Verification of Classified Sound Events)**: Solved through automatic observation creation and manual editing by users.
- **FR-4 (Editing of Audio Files)**: Solved differently by allowing users to download, edit externally, and re-upload audio files.
- **FR-14 (Graph for Uptime Visualization)**: Not implemented within the project scope.

Throughout the remaining project period, we prioritized integrating our frontend with the inherited backend, adapting API requests, adjusting data handling, and refining the authentication mechanisms. In parallel, we dedicated time to documenting the backend structure, database schema, and API functionalities, aiming to support ongoing development after the project's completion. We also verified and aligned the database structure to ensure that it fully supported the intended functionality.

We completed the following functional requirements:

Uploads and Storage:

- **FR-1**: Finalized API and frontend upload functionality.
- **FR-6**: Implemented cloud storage integration.
- **FR-7**: Enabled bulk downloading of audio files.

Audio Features and Playback:

- **FR-16**: Implemented playback of audio files directly from the frontend.

Metadata and Data Handling:

- **FR-8**: Extracted and displayed audio file metadata (sample frequency, length, size).
- **FR-15**: Linked audio files to deployments.
- **FR-2**: Completed production of observation files based on audio file data.
- **FR-3**: Solved by creating observations automatically, with users able to manually edit them.

Visualization and Browsing:

- **FR-9**: Refined and completed the map visualization.
- **FR-11**: Improved and finalized the lookup table for audio files.
- **FR-12**: Corrected display of total size for retrieved audio files per device.

While the system is fully functional, some technical debt remains, particularly regarding backend code optimization and expanding test coverage, which should be addressed in future development stages.

Three requirements were not fully realized. **FR-5 (NIRD Database Integration)** could not be completed due to a lack of access to the NIRD database, **FR-14 (Uptime Graph Visualization)** was left outside the current scope, and **FR-4 (Editing of Audio Files)** was solved differently by allowing download, external editing, and reuploading.

Although these elements remain for future development, the team delivered a functional, maintainable, and extensible system that meets the vast majority of the original goals. In the following section, we outline future work suggestions to further enhance and complete the system, ensuring it continues to support NINA's research needs effectively.

The system was developed with a strong focus on modularity, maintainability, and documentation, ensuring that future developers can easily continue extending and improving the platform (see Section 4).

2.9.1 Changes from vision

3 The Process

3.1 Project Overview

We started our process with a focus on clear and flexible planning. This section gives an overview of how we used Scrum to plan the project and includes a timeline showing how the work was organized throughout the project period.

3.1.1 Project Planning

Project planning involves setting goals, outlining tasks, assigning responsibilities, and creating a timeline. It ensures clarity about what needs to be done, when, and by whom. (kilde).

To ensure a structured yet flexible approach to project planning, we adopted Scrum as our framework. It provided both a method for organizing daily work and the overall planning of the project. In the prestudy, we defined the project goal and scope, and clarified requirements with the customer, forming the initial product backlog. We then divided the work into short, iterative sprints, each serving as a milestone and chance for reassessment. This helped us prioritize tasks based on customer needs and handle uncertainties as they arose, rather than resolving everything upfront.

Our approach followed a key agile value: responding to change over following a plan. (kilde) Instead of trying to decide everything at the beginning, we accepted that changes would happen as the project moved forward. We set clear long-term goals early, but kept the way we would reach them flexible, so we could adjust priorities and solutions based on new insights and feedback from the customer. This planning approach laid the foundation for how we later organized our practical work during the sprints, which will be described in the following sections.

An overview of how Scrum elements supported our project planning, along with references to where more details can be found in the report, is provided in Table 3.

Scrum Element	**How it Supported Project Planning**	**Reference in Report**
Product Backlog	Helped prioritize tasks and define project scope early based on customer needs.	Requirements
Sprints	Broke down the project into manageable iterations with regular reassessments.	Sprint diary
Sprint Planning	Structured planning sessions ensured clear goals and workload for each sprint.	Scrum
Customer Meetings / Demos	Provided feedback opportunities as needed, allowing flexible adjustments during the project.	Scrum
Sprint Retrospectives	Supported continuous improvement of processes and planning methods.	Scrum, sprint diary
Roles: Scrum Master and Development Team	Maintained structure and responsibility within the team, even without a dedicated Product Owner.	Scrum, roles

Table 3: Overview of how Scrum elements supported project planning

3.1.2 Project Timeline

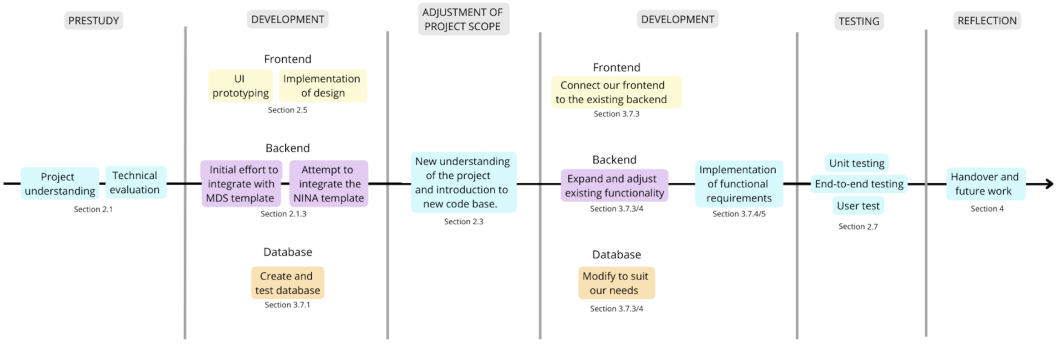

Figure 9: Project Timeline

3.2 Risk Assessment

A Risk analysis was created to identify, evaluate and mitigate potential risks we could encounter during our project. The template used was Digitaliseringsdirektorates "Risk assessment - Support tool template" [5].

3.2.1 Risk Identification

To identify all potential risks, the team worked together to compile a comprehensive list, following these steps outlined by [5]:

- **Identification and Description**
  Each potential risk was thoroughly analyzed and clearly defined.
- **Severity Evaluation**
  The severity of each risk was assessed based on its potential impact on project goals, deadlines, and deliverables.
- **Probability Evaluation**
  The probability of each risk occurring was assessed, taking into account past experiences and project-specific factors.
- **Severity Justification**
  The rationale for assigning each risk's severity level was documented to ensure a clear understanding of its potential impact.
- **Probability Justification**
  The rationale for the probability assessment of each risk was provided, highlighting the factors influencing its likelihood of occurrence.

Both likelihood and severity were classified into the categories of "Low," "Medium," "High," or "Very High."

3.2.2 Risk Measurements

The risk measurements were adjusted to reduce the probability and impact of the potential risks. Any remaining risks were assessed and accepted by the team.

3.2.3 Risk Matrix

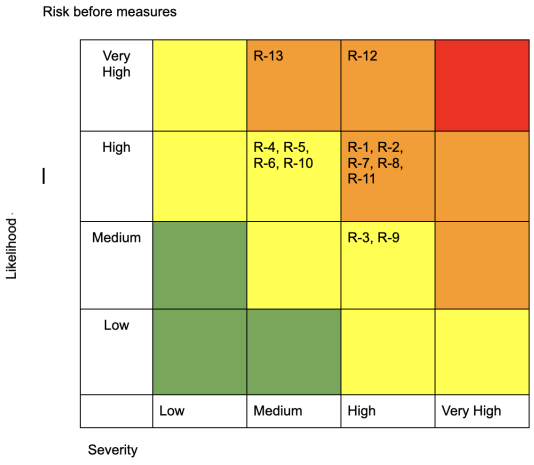

Figure 10: Risk before measures

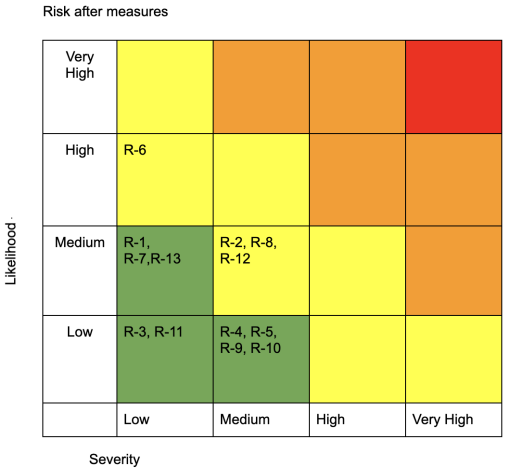

Figure 11: Remaining risk

3.2.4 Remaining risk

- **R-2 - Long term sickness or longer absence of members**
  We cannot control when team members fall ill or need to take extended absences. Therefore, the group has acknowledged this risk. If these absences were to impact the project's progress, we would initiate a discussion with both the customer and the course supervisor to determine the necessary steps forward.
- **R-8 - Unclear or changing project scope**
  During the initial phase, it became evident that the project description did not fully align with our assigned tasks, requiring multiple adjustments to the project scope throughout its duration. To address this, we adopted an iterative approach and implemented agile work methods, allowing us to adapt efficiently to evolving requirements.
- **R-12 - The project is dependent on external factors**
  Our project is closely tied to the employees at NINA and the larger initiative we are part of. This reliance on external expertise and knowledge often made us dependent on others. To mitigate potential delays or obstacles, we maintained proactive and responsive communication while also considering alternative solutions in advance. This approach allowed us to adapt effectively and ensure steady progress despite dependencies.
- **R-6 - Remote or physically absent team members**
  Given the minimal impact of a team member joining remotely, we have determined that remote participation is acceptable if they have communicated this in advance and there is a valid reason. However, physical attendance remains preferable whenever possible.

3.2.5 Reflections on Risk Assessment

At the start of the project, the risk assessment allowed the team to identify potential issues and proactively consider strategies to mitigate them throughout the development process.

3.3 Working with Scrum

Scrum provides a framework for the work process consisting of planning, meetings, role definition and reflection. Scrum is intentionally incomplete, and leaves it up to the users of it to fill in the

blanks [22]. Scrum focuses on being agile, flexible and responsive, to make software that works, through collaboration [1].

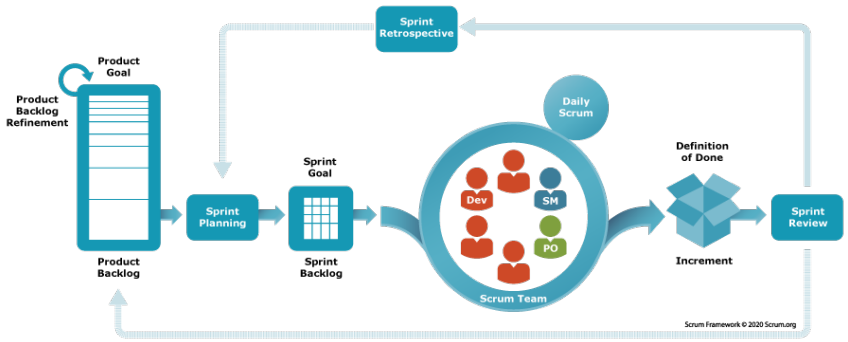

Figure 12: Sprint activities in Scrum [19]

The team wanted a flexible and agile working method to be able to adjust tasks and workload according to need. Frequent check-ins were considered necessary due to the large size of the team. Since the customer's requirements and priorities were expected to evolve over time, the ability to respond quickly and adjust accordingly was critical. Flexibility and rapid feedback became key aspects of the team's workflow.

As a result, Scrum was adopted as a framework, with selected features tailored to fit the specific needs of the project and the team.

This section will describe how the team implemented elements from Scrum, and discuss the benefits and challenges this has posed.

Sprint: The work, including Scrum activities, is planned for a set period of time. When the time period is over, a new period begins. Each period makes up a sprint [22]. The sprints were set to two weeks. Due to continuous adjustments, two weeks were considered sufficient time to work towards overall goals without risking working too long before accommodating the course to the customer's feedback.

Sprint planning: Planning the sprint is a Scrum activity, where next sprints tasks are decided [20].

As a part of planning the sprint, all task in the backlog were considered for the next sprint. Where tasks were not completed during the sprint, they were evaluated in case *a part* of the task was completed. In these cases the tasks were marked as done, and new tasks were made for the specifics still missing. This could leave a false impression that a greater quantity of the tasks were completed within the planned sprint, when parts of some tasks actually did stretch over to the next. See Gantt chart in Appendix.

An overall goal for each sprint were set. This was helpful as a guideline to prioritize tasks. See examples of this in the sprint diaries, starting at section 3.7.1.

Where the feedback or change in requirements or priority from the customer were given, the adjustments were done straight away, regardless of where the team was in the sprint. For instance, when the customer shared their changed preference regarding the ARISE project, as explained in section 2.4, the team immediately started the work of implementing it. This could arguably be contradictory to how the sprint should be handled, since changes are not to be introduced during a sprint [11]. However, this change was not an addition to existing development plan, but a complete change of foundation. The Agile manifesto clearly states to respond to change over sticking to a plan [1].

Daily Scrum: The Daily Scrum is a short update on each team members progress and work plan

for today [22]. The Daily Scrum was a part of the fixed weekly meetings to continuously make sure the whole team was updated on status and current issues. This contributed to synchronize the team, and to include and involve all members. Having to report previous work also put pressure on the members to do the work that had been planned. Having to formulate today's plan also made each member having to think through their own planned progress and how to achieve it. This was a good exercise, especially when progress was halted due to, for instance, problems with the technology or in periods awaiting answers from the customer.

Scrum task board / Sprint backlog: Tasks with time estimates were registered in ClickUp, as a form of Scrum task board, which is the sprint backlog [23].

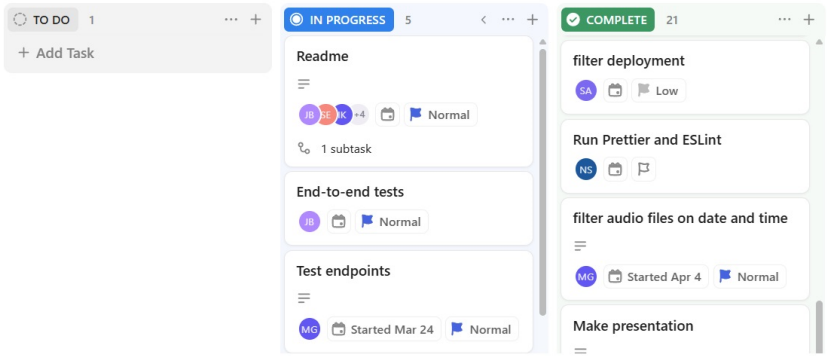

Figure 13: Task board of active sprint.

A new board was created for each sprint. Being able to see all tasks on a board, contributed to a better overview of current status through a tidy visualization.

The benefits of this would still be somewhat reduced when the lists of tasks were very long. The visual representation could then be overwhelming, and tasks could drown in the masses. The different level of familiarity to the tool, and the routine of using it, also limited the benefits of it. A task that needed to be done, and was registered and assigned in Clickup, were duplicated in Clickup and completed by someone that was not initially assigned to the task. This was revealed due to close communication within the team. Collisions like this could potentially lead to friction within the team if it were to happen often. As a mitigation measure, no new tasks were to be made in Clickup before all existing tasks were checked.

Scrum Master: The Scrum Master is operating as the teams coach, facilitating the work so that the team can perform their best. The Scrum Master is not a part of the developing team, but has a role closer to a managers, without any direct power over the member, only the work process [24].

The Scrum Master on the team *was* a part of the developing team, making the difference between Scrum Master and team member less obvious. The choice to still have someone with responsibility of having an overview was considered valuable, as a measure to avoid all members being locked in on their own assignments, loosing track of the big picture. This also contributed to the team taking responsibility as a collective.

The Scrum Master did not assert control over the team, allowing everyone in the team to influence the team work. A softer lead lowered the threshold for participation and welcomed all voices, which has been appreciated and necessary for a safe work environment. On the other hand, this made it less obvious who were to take initiative in coordination during times where some members were absent or non-responsive. It was up to each member of each team to reach out, since the Scrum master did not have a clear managing role. An outreach from the Scrum master could have had a more rapid effect, rather than a team member on the same level in the hierarchy reaching

out. A clearer defined and more visible Scrum Master role could therefor have served the team more beneficially.

Retrospective: After each sprint the team reflects on the recently ended sprint in a designated retrospective meeting. Each team member contributes with their experience on what worked well for them, what did not, and if there is something that needs to be done differently [25].

The team conducted retrospective meetings after each sprints. See "Retrospective insights" in section 3.5.2.

Product owner: The product owner is a part of the team, directing production towards the correct product [18]

TODO: ingen prodeier som forhandlet krav/ønsker med kunde Ingen prodeier

The team lacked a product owner, and during the process, the tasks of the product owner needed to be identified and handled by the team as a whole. In practice, this meant that the whole team participated in discussing and concluding on prioritizing tasks for each sprint, where the team collectively had to have eyes on the end goal for the customer.

This also meant that there was no one person negotiating tasks and timelimits

This did not pose great difficulties, but became more apparent when starting to work with the ARISE project. There was some uncertainty regarding the coordination of communication between Dr. Evans and the customer contact when issues that needed to be handled by the customer arised. This would be a natural assignment for the product owner, and would have made the transition smoother.

Overall, the elements used from Scrum gave the team a guideline on how to operate, but freedom to adjust and adapt to the teams needs.

3.3.1 Tools

A range of collaboration tools were incorporated into the project to streamline workflows, enhance communication, and improve task management. Each tool was carefully chosen to address key challenges faced during development, ensuring that the team could coordinate effectively, track progress, and maintain productivity across different phases of the project. By leveraging these technologies, the team was able to work more efficiently while keeping all stakeholders aligned on project goals.

Github

GitHub was used as the centralized version control system, allowing seamless collaboration, code review, and issue tracking. Through GitHub repositories, all team members had access to the latest codebase, commits, and branch updates, ensuring that development remained synchronized and conflict-free. Pull requests were actively used to review and discuss changes before merging, promoting code quality and team accountability. The built-in GitHub Actions were also utilized for automating workflows, such as continuous integration (CI) and deployment processes.

ClickUp

ClickUp was chosen as the primary project management tool, providing a structured and visual approach to tracking progress. The platform enabled the team to create task lists, assign responsibilities, set deadlines, and monitor overall project progression. The use of task statuses, such as "To Do," "In Progress," "Blocked," and "Completed," helped maintain a clear and transparent workflow. ClickUp's integrated time tracking feature was particularly useful for monitoring work hours and optimizing team efficiency.

Legge inn bilde av clickup her?

Figma

Figma served as the primary design and prototyping tool, allowing collaborative UX / UI development. As a cloud-based platform, Figma allowed the team to work on wireframes and interface designs in real-time, ensuring that all team members had instant access to the latest versions. The ability to add comments directly on designs facilitated feedback loops and iterative improvements. This tool was essential in bridging the gap between developers and product stakeholders, facilitating the transition from concept to implementation.

pgAdmin

pgAdmin was used for database management, providing an intuitive and efficient interface for interacting with the PostgreSQL database. The tool allowed team members to visualize, query, and modify database tables with ease. Its user-friendly UI and support for multiple database connections made debugging and database monitoring more efficient. By leveraging pgAdmin, the team could perform real-time database operations, ensuring data integrity and quick access to structured information throughout the development process.

3.4 Group Organization

(få inn alt av gruppegreier her, collaboration inne i teamet osv også)

3.4.1 Group Contract

To establish a shared understanding of expectations and responsibilities, the team developed a contract at the beginning of the project, which can be found in the appendix. This served as a framework for work effort, coordination routines, roles, ambition and conflict resolution, ensuring a structured and efficient working environment.

While the contract initially provided clear guidelines, such as aligning on ambition levels and expected work effort, we recognize in retrospect that we did not consistently follow or update it. Some aspects became outdated, for example, our planned use of Linear for sprint organization, which was replaced by ClickUp based on client preferences. Other elements were not strictly adhered to, such as task prioritization, which was meant to follow a dot-voting system but instead aligned with client priorities. Decision-making was formally set for Tuesday meetings, but in practice, many decisions were made continuously. Additionally, a policy requiring latecomers to bring snacks was not enforced, leading to a more relaxed attitude toward punctuality, ultimately causing minor inefficiencies and occasional frustration.

Reflecting on the contract, it was valuable in establishing team cohesion and a common understanding. However, we could have leveraged it more effectively to set clearer internal structures, enhance efficiency, and foster a stronger team environment. Regular updates to reflect evolving team dynamics would have further strengthened its impact.

3.4.2 Roles

We established roles within the group to ensure a structured workflow. One member took on the role of Scrum Master, and was responsible for facilitating meetings, keeping discussions on track, and ensuring that everyone contributed to the project. This helped the group maintain a healthy collaboration and accountability within the group.

We divided the team into specific work areas. Some members focused on frontend development, handling the user interface and user experience. Others were responsible for the backend, ensuring functionality and logic were implemented effectively. Additionally, two members worked on both the backend and database, managing data storage and retrieval to support the backend.

This division of responsibilities initially worked well, as it provided each member with a clear sense of ownership over their tasks while still having a shared responsibility for meeting project deadlines. By distributing work based on expertise, each team member could focus on specific

aspects of the project, which led to efficient task completion and specialization in key areas. The setup allowed individuals to take full responsibility for their assigned sections, minimizing bottlenecks and ensuring steady progress throughout the development process. Additionally, it encouraged accountability, as team members knew that others relied on their contributions to move the project forward.

However, while this structure worked in theory, it also introduced several challenges in practice. The separation of tasks created isolated teams, making it difficult for others to step in when issues arose outside their area of focus. When a team encountered roadblocks, other members often lacked the necessary context or knowledge to provide immediate assistance, leading to delays in problem-solving. The lack of cross-team familiarity with different project components resulted in inefficiencies, as troubleshooting required additional time to understand unfamiliar sections of the system before any meaningful contribution could be made.

Another significant issue was the lack of a holistic overview of the project. Since everyone was primarily focused on their specific tasks, few had a comprehensive understanding of the entire system. This created difficulties when teams needed to coordinate, as communication often became fragmented. Misalignments arose when different sections of the project required integration, and the absence of shared knowledge made it harder to foresee potential conflicts between different components. Additionally, when discussing progress with the customer or supervisor, the team sometimes struggled to provide a unified, detailed status update, as each member could only speak confidently about their own area of responsibility.

3.4.2.1 Reflections and Actions Taken

During the midterm evaluation, we identified that the strict separation of roles, while efficient for individual work, led to collaboration bottlenecks, integration difficulties, and limited flexibility in responding to unexpected issues.

Recognizing these challenges, we adjusted our approach for the second half of the project. Updates and knowledge sharing were actively incorporated into our daily standups, where team members not only reported progress but also explained their current work and discussed how it fit into the overall system. This helped improve awareness and communication across the team.

In addition, team members were rotated across different types of tasks, which encouraged everyone to become more familiar with multiple areas of the project. Over time, many of the tasks evolved into full-stack responsibilities that inherently required a comprehensive understanding of both frontend and backend systems. As a result, all members developed a deeper knowledge of the full architecture and data flow within the system.

This approach helped dissolve the silos that initially formed and supported a more integrated team dynamic, making it easier for members to assist each other, adapt to changes, and work collaboratively on complex problems.

3.4.2.2 Positive Outcomes

By the final stages of the project, these changes had a visible positive impact:

- Team members became more invested in the whole project, not just their specific tasks.
- Issue resolution became faster, as more people could meaningfully contribute outside their original areas of expertise.
- Customer demos and supervisor meetings became more coherent and confident, with multiple team members able to present the system as a unified whole.

Ultimately, balancing specialization with cross-collaboration proved to be one of the key lessons for the team in managing a complex, integrated project. This shift helped the team work as a more

cohesive unit rather than isolated subgroups, ensuring smoother collaboration, faster problem-solving, and a better final product.

3.4.3 Daily Workflows and Meetings

The group met regularly three times a week; on Mondays, Tuesdays and Wednesdays. Additionally we had meetings with the customer contact and supervisor. The internal meetings began with a daily standup where each member got to show what they have been working on since the last meeting, share potential problems they are encountering and what they will be working on until next meeting.

The documentation for new tools, for instance Django and pgAdmin, and forums such as Stack Overflow, was valuable in acquiring new competencies. We also consulted ChatGPT and each other.

Where there were difficulties in the development, each meeting was a chance to ask for help and support. It was the responsibility of each team member to voice their need, and the group as a whole to make sure the challenge was overcome. Members with available capacity helped and contributed to progress. When no member had available capacity it could be difficult to prioritize, especially where the tasks had the same priority.

If needed, we sought help from the resources at NINA or Dr. Evans. We aimed to not contact them unnecessarily, collecting questions to avoid sending multiple e-mails from several team members at the same time, like we experienced in the start of the development. This consideration could slow the progress down.

If there were no obvious tasks to take on, there would be a group discussion on what plan would be ahead. We would often also discuss questions we had to the project and what questions we could bring to the customer contact. We always had focus on having concise and informative meetings and avoid any bloat.

We did not experience great challenges within the team. The communication could still be more constructive. Hours used on different task were not always logged, and some members had greater difficulties logging their hours in the shared time sheet. This member would get personal reminders from one team member, in addition to general reminder to the whole team in meetings.

3.5 External collaboration

3.5.1 Coordination with the Customer

We maintained consistent communication with the customer through scheduled meetings. Weekly in-person meetings at the NINA house provided an opportunity for in-depth discussions, while additional virtual meetings via Teams were used for progress updates and resolving uncertainties. To streamline communication, we also established a dedicated Slack channel with the customer, enabling quick responses to questions and clarifications.

Both physical and virtual meetings proved valuable. In-person meetings facilitated detailed discussions and decision-making, whereas virtual meetings offered an efficient way to address minor issues without the need for travel. At the start of the development process, we were assigned a workspace at NINA, which significantly improved communication. However, we later lost access to this space, which posed challenges in maintaining the same level of seamless collaboration.

One of the main challenges in coordinating with the customer was managing their input and prioritizing tasks. Initially, they were unfamiliar with our areas of expertise, leading to the assignment of tasks that fell outside of our specialization. Additionally, some requirements were not essential to the project's core objectives. To address this, we held constructive discussions with the customer to clarify our expertise and collaboratively categorized tasks into low, medium, high and very high.

Another challenge we encountered was that our client contact was frequently working in the field for extended periods. This made quick and immediate communication difficult, requiring us to rely more heavily on well-planned weekly meetings, which were now conducted online. While this approach generally worked well, it led to delays during critical phases, such as when we adjusted the project scope.

Through reflection, we recognized the importance of maximizing client meetings to gather sufficient information to work effectively until the next scheduled interaction. This experience improved our ability to structure discussions efficiently and ultimately strengthened our communication with the customer throughout the project.

3.5.2 Coordination with the Supervisor

During the initial meeting with our supervisor, we agreed to have biweekly guidance meetings. During these meetings, we reviewed a status report that included a progress summary, open and closed problems, planned work for the next period, an updated risk analysis, and a time sheet.

The meetings have taken place both on Teams and in person, ensuring that we could choose the most effective form of communication for each session. In addition, we have maintained regular communication with our supervisor on Slack through both a shared channel and private chats, where we received frequent responses whenever needed.

At the beginning of the project, our supervisor ensured that we got off to a strong start by setting clear expectations for how we should communicate with the customer. When significant changes occurred in our project, she provided valuable guidance and encouraged us to take an active role in shaping these changes to align with our desired direction.

3.6 Process Adjustments

3.6.1 Adaptations During the Project

3.7 Sprint Diary

tabell av alle sprintene med navn og mål feks.

Burndownchart inn under denne section et sted. Enten i introen her eller som en oppsummeringsgreie til slutt.

3.7.1 Sprint 1: 11.02 - 03.03

Sprintgoal: During this sprint, our objective was to deliver a working MVP for a device lookup table while establishing the necessary project foundations on the frontend, backend, and database layers.

Workflow: On the frontend, we finalized a Figma draft, aligned the design with the customer, and developed a functional MVP, including a sidebar for navigation and an audio file table. The backend team focused on integrating templates, shifting from ARISE to the more viable NINA template. Meanwhile, the database team designed schemas, set up the environment, and implemented CRUD operations with automated testing.

Retrospective insights: Our retrospective highlighted key improvement areas, especially given the impact of sickness and absences:

- All members must log their hours. Inconsistent time logging made it difficult to monitor progress, particularly when team members were absent.

- Use screen sharing and demonstrations during standups to improve transparency. The lack of detailed updates made it difficult to follow progress, especially when people were out.
- Better planning on what to do on the report, so that everyone contributes.
- Team members need to stay updated on what is happening within their own subteams (frontend, backend, database) to ensure they can lead meetings and answer questions if others are absent.

This sprint provided valuable deliverables and insights, guiding us to improve efficiency, communication, and workload management moving forward.

3.7.2 Sprint 2: 04.03 - 17.03

Period of Adjusting to New Requirement

At the start of the sprint, we had a meeting with the customer contact and Dr. Evans, where we were informed about the new requirement of NINA to build on Julian's ARISE project. This unexpected change created challenges in reorganizing and planning our next steps.

Challenges and First Week

The sudden change in project requirements made it difficult for us to ask the necessary questions during the meeting, leading to uncertainty regarding requirements, project scope, and user interface choices. Additionally, the customer contact was mostly unavailable during the first week, which further reinforced our uncertainty. Given the uncertainty, we decided to pause sprint planning and spend a week understanding Dr. Evans project, setting it up, and clarifying key points.

During a meeting with our supervisor, we were reassured that such changes are common in both projects and professional work. We were also encouraged to propose how we wanted to proceed. This motivated us to pitch the idea of developing our own user interface.

Second Week and Clarifications

After the initial week, it became clear that one week was insufficient to fully understand the product and clarify requirements. To better align the project with NINA's expectations and avoid disrupting Dr. Evans' ongoing work, we extended this phase by an additional week.

During the second week, we confirmed the functional requirements, enabling more efficient task distribution: some team members continued to familiarize themselves with the ARISE project, while others focused on documenting our findings. We also proposed the development of a new user interface, which the customer supported after reviewing our design sketches and comparisons. In parallel, we agreed with Dr. Evans to fork the project to avoid conflicts in development.

Retrospective Insights

During the retrospective, we reflected on our reactions to the project changes and how we could have handled things better. It became clear that the team experienced a decrease in motivation due to uncertainty about the project's direction, slow progress, and concerns about the project's value. Based on these reflections, we identified several key areas for improvement. While most are particularly relevant if new uncertainties arise, the first point is also crucial for maintaining quality in regular, ongoing work.

- Stand-ups should ensure that everyone has concrete tasks and prevent periods of idleness. We realized the importance of ensuring that every team member has clear and meaningful tasks to avoid disengagement.
- In case of uncertainty, immediately ask the customer or relevant stakeholders for clarification. By responding more quickly to project changes and reviewing functional requirements early, we could have avoided stagnation.

- Use of external perspectives when relevant. Meetings with Dr. Evans and our supervisor proved valuable in providing external perspectives.

Despite the challenges, the retrospective provided us with valuable insights. We now have a clearer direction, renewed motivation, and a more structured approach to handling project changes. Moving forward, we will focus on proactive communication and better planning to ensure steady progress.

3.7.3 Sprint 3: 04.03 - 17.03

Sprint goal: Connect the database, backend, and frontend as an extension of Julian's project, and deliver a draft of the report.

Workflow: To facilitate the integration of the frontend and backend, we prioritized setting up Swagger before connecting the two. However, this process took longer than anticipated, delaying the start of API integration. While waiting, we focused on developing the frontend for the form, drafting the report, and conducting a review.

The extended time required for setting up Swagger and writing the report prevented us from fully achieving the sprint goal of connecting all system components. Despite each team member dedicating the planned 15–20 hours per week, the delays impacted our progress.

Retrospective insights:

We began our retrospective by reviewing whether the action points from sprints 1 and 2 had been followed. Overall, we were satisfied with the improvements, but time tracking remained an issue. To address this, we clarified the action point:

- All members must log their hours from the previous week before the Monday meeting, where this will be checked.

We also discussed our estimation process, given that the sprint goal was not met. We underestimated the time required for report writing, which took priority over development since it was a submission.

As the reporting workload was the main issue, we saw no need to adjust estimations for development tasks. However, we were disappointed about missing the sprint goal and will be more cautious about setting ambitious development targets when other significant tasks are expected.

We identified further improvements:

- Tasks in progress or completed should be marked accordingly in ClickUp, to be reviewed during stand-ups.
- Sprint planning should involve more discussion as a full group and less time in subteams to ensure alignment toward shared goals.

3.7.4 Sprint 4: 18.03 - 31.03

Sprint goal: Finish very high and high functional requirements.

Workflow: The sprint involved two planning sessions due to overlapping tasks and uncertainty about how long it would take to integrate the backend and frontend. Initially, there weren't enough tasks defined, which caused some confusion about priorities. However, as the sprint progressed, we gained a clearer understanding of what needed to be done. Despite the early challenges, we successfully completed the sprint goal and delivered a few medium-level functional requirements. The team adapted well and made steady progress throughout the sprint.

Retrospective insight: We began the retrospective by reviewing the action items from the previous sprint, which we were pleased with in terms of follow-up. Overall, all team members were satisfied with the sprint and noticed significant progress in the product development.

We identified several action points:

- Make sprint goals and action items available on ClickUp To make it easy for everyone to remember.
- Set task priorities in ClickUp Priorities should align with the functional requirements or any dependencies within the project.
- Implement Planning Poker for all development tasks To ensure that everyone has a say and that no one overrides the planning process, we will introduce Planning Poker for development tasks.

3.7.5 Sprint 5: 01.04 - 14.04

Sprint goal: Finish medium and low functional requirements.

Workflow: På grunn av at medlemmer

- skulle bli ferdig med kode på onsdagen, det skjedde ikke

- vanskelig når folk ikke er samlet

- masse småendringer fra kunde

Retrospective insight:

- på grunn av påske ble den flyttet til rett etter påske,

3.7.6 Sprint 6: 15.04 - 28.04

Sprint goal: Finish the product, including testing and documentation.

Workflow:

Retrospective insight:

4 Future work

In light of the current state of the project and the feedback gathered during the final stages, several areas for further development have been identified. These include addressing limitations found during user testing, implementing additional functionalities requested by the customer, completing remaining deployment-related tasks, and facilitating potential reuse of individual elements by external collaborators. This section outlines the main points for future work.

4.1 Problems Identified During User Testing

During the final demonstration of the system, several additional improvement areas and potential future needs were identified. Although this was the final demo before project completion — a stage where new requirements are usually avoided — it is common in agile development that stakeholders recognize new needs when seeing the system in its near-final form. As noted in *Scrum and XP from the Trenches* (pp. 80–82), final demos often attract crucial customer feedback that helps improve product quality.

While these new suggestions were not included in the agreed project scope, they provide valuable direction for future development. The improvements discussed include:

- **File Metadata Accuracy:** It was observed that sample rate and file length are currently retrieved from the database during datafile filtering, rather than being extracted directly from the audio files. To ensure data consistency and reliability, the system should be improved to read these values directly from the source files.
- **Date-Proximity File Retrieval:** Users requested the ability to easily locate files closest in time to a selected date — both the nearest file before and after — to support more efficient temporal browsing of data.
- **Map Display Improvements:** In the current deployment map, the location is labeled by country. Stakeholders expressed a preference for displaying the unique deployment ID instead, to improve clarity when multiple deployments exist within the same country.
- **Deployment Editing Enhancements:** There was a strong interest in allowing users to edit deployment information directly within the deployment detail view, including uploading pictures associated with each deployment.
- **Sidebar Layout Adjustment:** It was noted that the map should appear above the deployment list in the sidebar, reversing the current layout to better match user expectations and improve navigation.

All additional feedback and new feature requests identified during the final demonstration have been carefully documented and handed over to NINA for consideration in future development planning.

4.2 Additional Functionalities

Based on the state of the product at project completion (Section 2.10), several originally intended functionalities were not completed within the project timeframe. These are listed below for future follow-up and prioritization.

Functional Requirements Not Completed

The following functional requirements (FRs) were not fully realized:

- **FR-5 (NIRD Database Integration):** Integration with the external NIRD database was not completed due to access limitations.
- **FR-14 (Uptime Graph Visualization):** Visualization of device uptime through graphs was deprioritized and left outside the project scope.
- **FR-4 (Editing of Audio Files):** Instead of implementing direct audio file editing within the system, an alternative solution was provided, allowing users to download files, perform external editing, and reupload them.

These functionalities remain important for the long-term goals of the system and should be considered for future development, even though they were not prioritized during the project.

Non-Functional Requirements Not Fully Addressed

Several non-functional aspects also require further work:

- **Performance:** According to the non-functional requirements, "The system should perform efficiently, with minimal loading times for pages and data processing."
- **Scalability:** Currently, the observation job is executed within the Django-based SensorPortal backend. Since the creation of observations is resource-intensive, offloading this task to a dedicated Celery worker would significantly reduce the load on the backend and improve overall scalability.

Additional Ideas from Collaboration

During the project, several additional ideas emerged in collaboration with the customer that, while not part of the formal requirements, could be valuable in the long term. These include improving local development tooling (e.g., Docker support for easier onboarding), lightweight administrative utilities, and user feedback mechanisms integrated into the interface. These improvements could enhance the maintainability and usability of the system going forward.

4.3 Deployment – Remaining Work

To transition the system from a development prototype to a fully operational solution, several deployment-related tasks must still be completed. These primarily involve infrastructure integration, authentication and access control, security hardening, and coordination with NINA's internal server environment.

Integration with NIRD

At present, the application stores files locally using a Google Cloud Storage solution. However, the long-term objective is to migrate all data storage to NIRD. This will require establishing a reliable connection between the application and NIRD's file system. As part of this migration, file uploads and observation generation must be redirected to operate entirely within the NIRD environment.

User Authentication Integration

The system is currently not connected to NINA's internal authentication infrastructure. For production use, it is necessary to integrate with NINA's centralized login system. In addition, role-based permission checks should be implemented throughout the application to enforce access control and ensure consistent and secure user behavior.

Backend Security Configuration

Prior to deployment, several backend security settings must be configured. This includes defining an appropriate Cross-Origin Resource Sharing (CORS) policy to restrict access to trusted hosts, as well as setting up security middleware in the reverse proxy. These configurations should be adapted to NINA's internal security policies to ensure compliance and mitigate risk.

Deployment to NINA Servers

NINA operates its own internal server infrastructure for hosting production systems. During deployment, it will be essential to configure container parameters correctly and establish reliable communication between application services and Celery worker processes. Proper orchestration will be key to ensuring a stable and maintainable deployment environment.

4.4 Integration with NINA's Long-Term Project

This project's outputs will primarily support NINA's ongoing research activities. It is NINA that will maintain, extend, and operationalize the system after project completion. The codebase, database schemas, and system design were developed to be modular and easily extendable to support future research needs.

Our transition to the overtaken system midway through the project, as outlined in Section 2.10, required careful adaptation to a new architecture. This background highlights the importance of clear documentation and structured code, which have been prioritized to facilitate smooth handover to NINA's future developers.

NINA's developers are encouraged to use the provided documentation and to build on the current system, completing the deferred tasks and implementing enhancements based on their evolving needs.

4.5 Potential Reuse by Dr. Evans

The system developed in this project was originally forked from Dr. Julian Evans' ARISE-MDS codebase, which is widely used in environmental data management by NINA and other institutions in the field. While our solution evolved to meet the specific needs of NINA, it still shares structural similarities and follows the same general coding style as the original ARISE project. This provides a basis for potential code-level reuse, even though merging the projects directly would likely be complex due to their diverging purposes and architecture.

Despite this, some of the improvements made during our work may be relevant for Dr. Evans and could be selectively incorporated into his system if desired. These include:

- Enhancements to the frontend interface, particularly in terms of usability, filtering, and file browsing workflows.
- Refinements to how audio file metadata (such as sample rate, length, and size) is handled and displayed.
- Additional audio-related functionality, such as audio quality checks and observation file generation.

Any decision to reuse or adapt elements of our implementation rests entirely with Dr. Evans, based on the direction and requirements of his own project. The continued development and maintenance of the system described in this report is the sole responsibility of NINA. Should Dr. Evans choose to incorporate any part of our work into his own solution, we would view that as a positive outcome and a recognition that our contributions offer value beyond their original context.

5 Summary

6 References

References

[1] K. Beck et al. *Manifesto for Agile Software Development*. Accessed: 12.03.2025. 2001. URL: https://agilemanifesto.org/.

[2] I. Avery Bick et al. 'National-scale acoustic monitoring of avian biodiversity and migration'. In: *bioRxiv* (2024). DOI: 10.1101/2024.05.21.595242. eprint: https://www.biorxiv.org/content/early/2024/12/01/2024.05.21.595242.full.pdf. URL: https://www.biorxiv.org/content/early/2024/12/01/2024.05.21.595242.

[3] Vite Contributors. *Vite Documentation*. Accessed: 12.03.2025. 2025. URL: https://vitejs.dev/guide/.

[4] Benjamin Cretois et al. 'Snowmobile noise alters bird vocalization patterns during winter and pre-breeding season'. In: *bioRxiv* (2023). DOI: 10.1101/2023.07.13.548680. eprint: https://www.biorxiv.org/content/early/2023/07/15/2023.07.13.548680.full.pdf. URL: https://www.biorxiv.org/content/early/2023/07/15/2023.07.13.548680.

[5] Digitaliseringsdirektoratet. *Risikovurdering - støtteverktøy*. Accessed: 04.03.2025. 2025. URL: https://www.digdir.no/informasjonssikkerhet/malar-og-eksempel/3164#handtering_av_risiko.

[6] Inc. Docker. *Docker Best Practices*. Accessed: 12.03.2025. 2025. URL: https://docs.docker.com/develop/dev-best-practices/.

[7] Joseph Evans and contributors. *ARISE MDS Sensor Portal*. https://github.com/jevansbio/ARISE-MDS-sensor-portal. Accessed: 2025-03-06. 2025.

[8] Django Software Foundation. *Django Documentation*. Accessed: 12.03.2025. 2025. URL: https://docs.djangoproject.com/en/stable/.

[9] Pavlo Gorbachenko. *What Are Functional and Non-Functional Requirements and How to Document These*. Accessed: 14.04.2025. URL: https://enkonix.com/blog/functional-requirements-vs-non-functional/.

[10] International Organization for Standardization (ISO). *ISO 25010 - System and Software Quality Models*. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. Accessed: 2025-03-10. 2025.

[11] Henrik Kniberg and Mattias Skarin. *Kanban Scrum, making the most of both*. C4 Media, 2010.

[12] Philippe Kruchten. 'Architectural Blueprints—The "4+1" View Model of Software Architecture'. In: *IEEE Software* (1995).

[13] Tailwind Labs. *Tailwind CSS Documentation*. Accessed: 12.03.2025. 2025. URL: https://tailwindcss.com/docs.

[14] Inc. Meta Platforms. *React Documentation*. Accessed: 12.03.2025. 2025. URL: https://react.dev/.

[15] Norwegian Institute for Nature Research (NINA). *Acoustic Monitoring - Norwegian Institute for Nature Research (NINA)*. Accessed: 2025-03-06. 2025. URL: https://www.nina.no/english/Fields-of-research/Environmental-monitoring/Acoustic-monitoring.

[16] Norwegian Institute for Nature Research (NINA). *NINA Python Template*. https://github.com/NINAnor/python-template. Accessed: 2025-03-06. 2025.

[17] Norwegian Institute for Nature Research (NINA). *TABMON - Norwegian Institute for Nature Research (NINA)*. Accessed: 2025-03-06. 2025. URL: https://www.nina.no/english/TABMON.

[18] Scrum.org. *What is a Product Owner?* Accessed: 12.03.2025. 2025. URL: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-team/what-is-a-product-owner.

[19] Scrum.org. *What is Scrum?* Accessed: 12.03.2025. 2025. URL: https://www.scrum.org/learning-series/what-is-scrum/what-is-scrum.

[20] Scrum.org. *What is Sprint Planning?* Accessed: 12.03.2025. 2025. URL: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-events/what-is-sprint-planning.

[21] Mountain Goat Software. *The Scrum Framework*. Accessed: 14.04.2025. 2024. URL: https://www.mountaingoatsoftware.com/agile/user-stories.

[22] Mountain Goat Software. *The Scrum Framework*. Accessed: 12.03.2025. 2024. URL: https://www.mountaingoatsoftware.com/agile/scrum.

[23] Mountain Goat Software. *The Scrum Framework*. Accessed: 27.04.2025. 2024. URL: https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/task-boards.

[24] Mountain Goat Software. *The Scrum Framework*. Accessed: 12.03.2025. 2024. URL: https://www.mountaingoatsoftware.com/agile/scrum/roles/scrummaster.

[25] Mountain Goat Software. *The Scrum Framework*. Accessed: 12.03.2025. 2024. URL: https://www.mountaingoatsoftware.com/agile/scrum/meetings/sprint-retrospective.

Appendix

article graphicx lscape subcaption

A Risk Assessment

ID	Risk description	Justification for the severity of the risk	Severity	Likelihood	Justification for the likelihood of the risk	Risk impact	Mitigation of risk management	Severity after measurers	Likelihood after measures	Remaining risk impact
R-1	Conflicts or disagreements within the team	If disagreements arise within the team, productivity may decline, potentially causing team members to produce results below the expected quality based on the scope of the project	High	High	As a new team, with members bringing different individual perspectives and approaches, we can experience diverse opinions, unclear roles, and communication challenges that can easily lead to disagreements.	High	The team holds regular meetings to review progress and address issues collectively. Clear roles and responsibilities will be established, and open communication will be encouraged to address issues early. Unresolved conflicts are escalated to the supervisor.	Low	Medium	Low
R-2	Sickness or longer absence of members	Unexpected illnesses or personal situations could arise, potentially reducing the group's overall productivity and capacity.	High	High	Due to the unpredictability of illnesses and the team's limited experience in handling absences effectively.	High	Regular status updates will help the team stay informed, making it easier for others to step in if needed. Key tasks will be documented to ensure continuity during absences. Team members must inform the group as early as possible if an absence is expected. Contagious members stay at home.	Medium	Medium	Medium
R-3	Communication failure within the team	Communication failure within the team can lead to misunderstandings, misaligned tasks, delays, and reduced overall productivity, impacting the product.	High	Medium	Being a new team can lead to misunderstandings, unclear messages, and unestablished communication routines.	Medium	We have established dedicated channels for different types of communication, including Slack for information sharing and Messenger for quick updates. Regular stand-up meetings three times a week ensure consistent communication and coordination between subteams.	Low	Low	Low
R-4	Uneven workload or avoidance of work	Reduced work effort from a team member can affect overall team performance, causing delays and potentially lowering the quality of the final product.	Medium	High	The team is newly formed and may face challenges in task allocation. Individual differences in work pace or motivation could lead to uneven distribution and possible task avoidance.	Medium	The group will regularly review and compare the hours worked by each member after each sprint. Standup meetings will include updates on individual progress since the last meeting. During sprint planning, tasks will be clearly assigned to ensure balanced workloads and accountability.	Medium	Low	Low
R-5	Poor planning	Poor planning, including both underestimating and overestimating tasks, can lead to uneven workload distribution, delays, wasted resources, and difficulties in meeting project goals effectively.	Medium	High	Since the team is new, we may struggle with setting clear goals, timelines, and task distribution, leading to planning issues.	Medium	We will organize simple sprint planning sessions to set clear tasks and deadlines. After each sprint, sprint reviews will evaluate the hours worked and tasks completed, helping us improve future time estimates and planning accuracy. Regular progress checks will ensure the team stays aligned and adaptable.	Medium	Low	Low

R-6	Remote or Physically Absent Team Members	Team members working remotely or being physically absent from the group can lead to reduced collaboration, slower communication, and potential misalignment in tasks and project progress.	Medium	High	While it is hard to predict when members may be remote or absent, such situations are typically infrequent and rarely cause significant disruptions based on past experience.	Medium	Team members must inform the group early about planned absences, which will be added to the shared calendar. If possible, members can participate digitally. If unable to work, tasks will be delegated, but the absent member should still aim to complete the agreed-upon hours.	Low	High	Medium
R-7	Insufficient Technical Expertise	Insufficient technical expertise can lead to delays, implementation challenges, and a higher risk of errors, affecting the quality and efficiency of the project's development.	High	High	As students, we are still developing our skills, making it likely that certain tasks will exceed our current expertise.	High	We will encourage knowledge sharing and collaboration within the team to bridge skill gaps. External resources, such as online materials, will be used when needed. Tasks will be assigned based on individual strengths and motivation.	Low	Medium	Low
R-8	Unclear or Changing Project Scope	Can cause confusion and delays, impacting overall project delivery.	High	High	In an agile approach, scope changes are common, making the likelihood of unclear or shifting project requirements relatively high.	High	We hold weekly meetings with the product owner to receive feedback on progress, clarify any questions, and address new or changing requirements to ensure alignment and minimize confusion.	Medium	Medium	Medium
R-9	Decreased Product Quality Due to Development Errors	During development, errors or missteps can cause a decline in product quality, potentially affecting functionality or performance. The impact will depend on the severity of the issue and where it occurs within the development process.	High	Medium	Development errors can occur due to limited experience, unfamiliar tools, or complex tasks.	Medium	Code reviews and pair programming will be implemented to identify errors early. Automated testing and frequent testing cycles will help catch issues during development.	Medium	Low	Low
R-10	Technical Debt	Technical debt arises from quick fixes or suboptimal solutions, leading to maintenance issues, reduced code efficiency, and delays in future development.	Medium	High	Due to time constraints and learning curves there is a high chance of accumulating technical debt during development.	Medium	To manage technical debt, we will set clear testing and completion requirements, conduct regular code reviews, and prioritize documentation and comments to ensure sustainable, maintainable code.	Medium	Low	Low
R-11	The customer comes with impossible demands and requiremen	Requests that exceed scope, resources, or feasibility can result in delays, scope creep or team burnout.	High	High	The customer may not fully understand the team's capabilities or the time required for tasks, making unrealistic demands more likely.	High	Weekly meetings will be used to clarify scope	Low	Low	Low
R-12	The progress is dependent on external factors	External factors such as the performance of the other bachelor group working on the project or external systems can impact project timelines.	High	Very hig	Another bachelor group is working on the hardware, making it highly likely that parts of our task will be affected by their progress.	High	Regular progress monitoring of dependencie	Medium	Medium	Medium