

A close-up of a logo

Description automatically generatedGroup project research report – CY2

# Contents

[Contents 2](#_Toc167478263)

[Introduction 3](#_Toc167478264)

[The stakeholder 3](#_Toc167478265)

[The project 3](#_Toc167478266)

[The research 3](#_Toc167478267)

[Q1: What is the current system’s architecture? [Yordan] 4](#_Toc167478268)

[Source 1: Project’s internal documentation 4](#_Toc167478269)

[Source 2: Developer interview 6](#_Toc167478270)

[Conclusion: 9](#_Toc167478271)

[Q2: How can an existing solution help with system monitoring? [Nikolay] 9](#_Toc167478272)

[Source 1: Available product analysis 9](#_Toc167478273)

[Source 2: Domain modelling 13](#_Toc167478274)

[Conclusion: 14](#_Toc167478275)

[Q3: What is a good way to track system operation? [Tudor] 15](#_Toc167478276)

[Source 1: Good and bad practices 15](#_Toc167478277)

[Source 2: Expert interview (Marcel) 17](#_Toc167478278)

[Conclusion: 19](#_Toc167478279)

[Q4: What is an efficient way to visualize system operation data?[Kiril] 19](#_Toc167478280)

[Source 1: User interview 19](#_Toc167478281)

[Source 2: Prototyping 20](#_Toc167478282)

[Source 3: Research on Data Visualization Best Practices 22](#_Toc167478283)

[Conclusion 22](#_Toc167478284)

[Q5: What is the quality and stability of the system [Yordan + Kiril] 23](#_Toc167478285)

[Source 1: Code review 23](#_Toc167478286)

[Source 2: System test 25](#_Toc167478287)

[Conclusion: 27](#_Toc167478288)

[Q6: How can the system be improved further [Nikolay + Tudor] 27](#_Toc167478289)

[Source 1: Peer review 27](#_Toc167478290)

[Source 2: Product review 29](#_Toc167478291)

[Conclusion: 30](#_Toc167478292)

[Conclusion 31](#_Toc167478293)

[References: 32](#_Toc167478294)

# Introduction

## The stakeholder

CY2 has more than 15 years of experience in assistance of Higher Education by extending and improving Campus Solutions. As an official Oracle development partner, the company primarily focuses on PeopleSoft campus solutions and chatbots.

What sets CY2 apart is their approach to solutions, which are custom-made in close collaboration with clients. This ensures that their products meet not only the current needs of universities but also anticipate future requirements. As a result, they are able to offer tailored solutions that cater specifically to the unique challenges and objectives of each educational institution.

## The project

This project concerns CY2’s university FAQ system which provides chat bots that can be integrated into universities’ 3rd party systems. This system exists to allow universities to train chatbots on custom documents. As of the beginning of this project, a baseline for the project has already been created by a previous group.  
This project is a near-greenfield project which was only worked on by one group. The previous group managed to implement some critical functionalities like combining an LLM with a document source into a pipeline. However, the system is still quite barebones. User management does not exist, and administrators need to add users to the database manually. Moreover, system observability is not present and administrators don’t know if there are errors in the system.

This system leverages pre-existing internal knowledge bases like SharePoint and generative AI, enabling it to deliver tailored FAQ answers to the user with reduced maintenance overhead for educational institutions.

In order to address this issue, CY2 is seeking to build an Insights solution for our FAQ system. This solution should provide system monitoring, as well as give administrators with clear and understandable graphs into the decisions and actions undertaken by the AI model used in the FAQ system.

Is more information to be desired about the context of the application, it can be found as part of the *Project Plan[1]* which has been created as a starting point of the application’s development.

## The research

After analyzing the context of the situation, we, as the current developers of the application are left with a question meant to encompass the entire contents of what has been assigned to us. As such, we have created the following research question:

**How do we allow administrators of CY2 and organization developers to gain more insight about the system operation?**

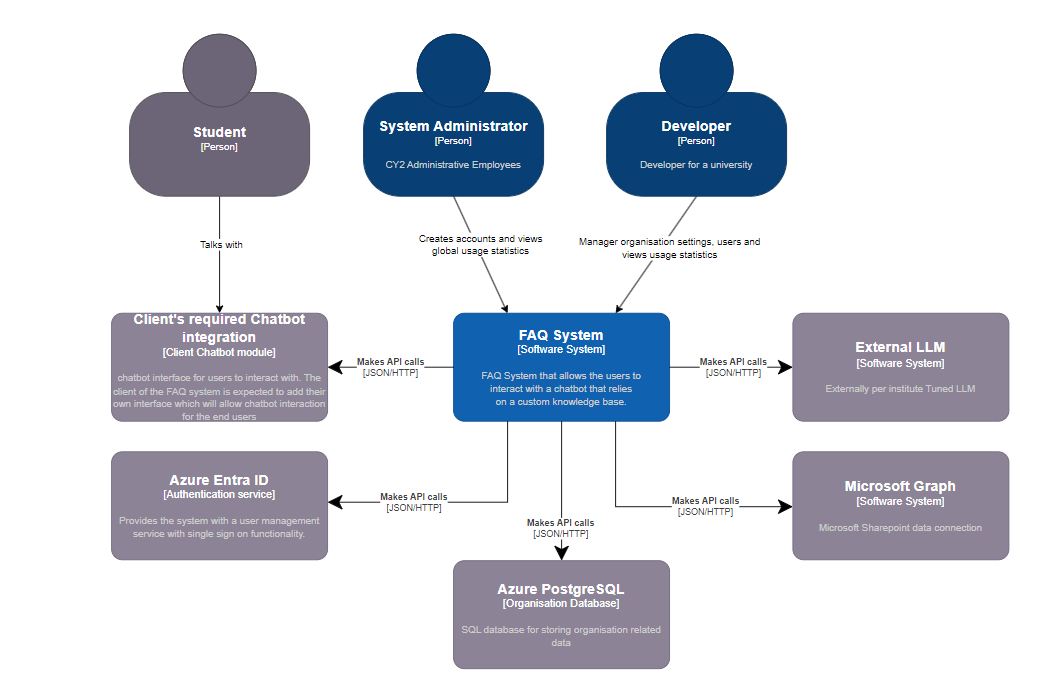
Below can be found the research that was done for the group project in semester 6 documented question by question. This is done to convey the steps taken in answering the questions, along with the conclusions that have been drawn. The DOT framework is used as an inspirational skeleton and is not followed too strictly. This research is based on most of the methods depicted in the DOT framework, the triangulation method, as well as common sense.

# Q1: What is the current system’s architecture? [Yordan]

For the first sub question, we have decided that we require more insight into the inner workings of the application at a system level. Thus, we have decided to analyze the way the system itself is structured. In order to answer this, we have decided to conduct Field (Document analysis) as well as Library (Expert Interview) research (as they are classified by the *DOT Framework [6]*).

## Source 1: Project’s internal documentation

To answer this question, we can refer to the documentation left behind by the previous team that worked on the project. Going into the project’s GitHub repository, a document named “architecture.md” can be seen, which contains the application’s C1 (Figure 1 - C1 Diagram of FAQ System), C2 (Figure 2 - C2 Diagram of FAQ System) and C3 diagrams.

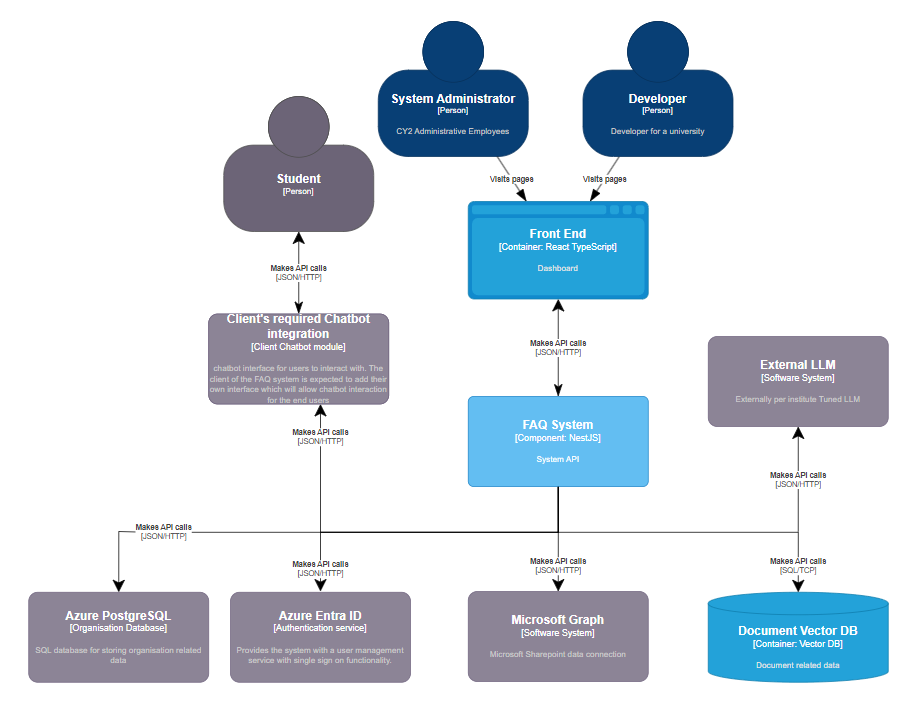


To be implemented

Figure 1 - C1 Diagram of FAQ System

From the C1 diagram it can be seen that there are 3 main actors for the system – the System administrator, the developer and the student. System administrators are CY2 employees that keep track of the system’s health and develop it further. Developers are employees in partner universities that integrate the system into their own infrastructure and leverage it. The students seem to chat with an Oracle Digital Assistant, which in term uses the FAQ system to obtain answers for their university-related questions.

The FAQ system itself relies on a few external systems like Azure Entra ID, Azure PostgreSQL, external LLM models and Microsoft Graph.



To be implemented

Figure 2 - C2 Diagram of FAQ System

Diving deeper into the C2 diagram, it can be seen that the FAQ system itself consists of a React frontend, a NestJS Rest API and a Pinecone vector database, as can be directly observed in Figure 2 - C2 Diagram of FAQ System.

In order to gain even more knowledge regarding the architecture of the application, we have also gone over the C3 diagram of the system, as can be observed below in Figure 3 - C3 Diagram of FAQ System

A diagram of a company

Description automatically generated

To be implemented

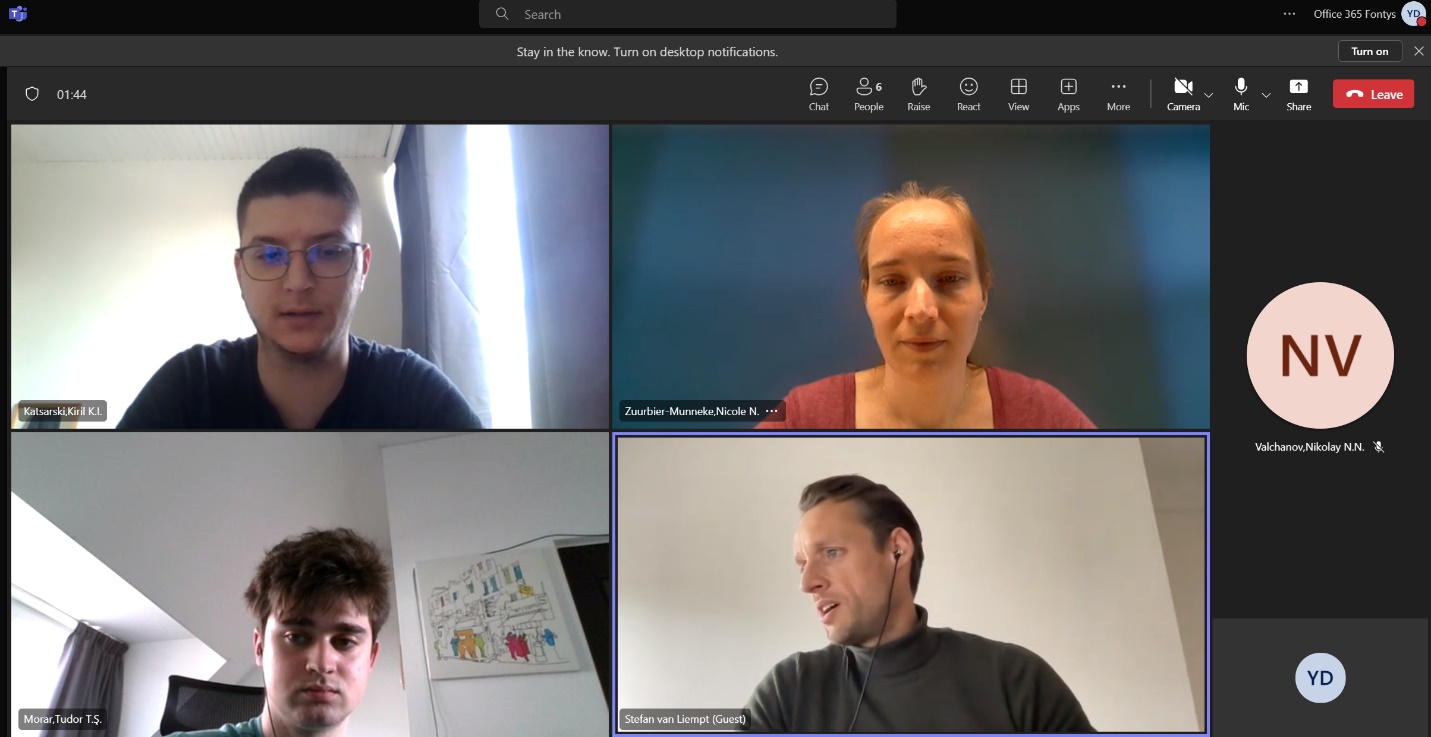
Figure 3 - C3 Diagram of FAQ System

As part of the C3 diagram, the services of the application can be analyzed. A first observation can be made regarding the possible responsibility of each component based on the attributed names, which is that, from the SOLID principles, the *Single responsibility principle* appears to be generally respected.

## Source 2: Developer interview

In order to confirm the previous findings, an interview with a developer of the CY2 system was scheduled. His name is Koen Janssen and he is a student currently doing his graduation at CY2. He was tasked with making improvements on the LLM side of the system, so there was a good chance that he knew whether the documentation left behind by the previous team was relevant.

The interview occurred on MS teams.



Koen mentioned the following:  
“The documentation left by the previous team is mostly relevant. However, there are some discrepancies that we can go over”.

As he explained, the Oracle Digital Assistant was not actually a part of the system and was a scrapped idea. The current idea was that the system administrators and developers can directly interact with the chat bots on CY2’s frontend for testing purposes, and the developer would need to integrate their own chat interface within their systems. There, students would be able to chat with the bots provided by CY2’s backend. As a result of these insights, the C4 diagrams required updating. In that sense, the outdated diagrams can be observed in the following figures: Figure 4 - Outdated C1 Diagram, Figure 5 - Outdated C2 Diagram, Figure 6 - Outdated C3 Diagram.

A diagram of a system

Description automatically generated

Figure 4 - Outdated C1 Diagram

A diagram of a computer system

Description automatically generated

Figure 5 - Outdated C2 Diagram

A diagram of a company

Description automatically generated

Figure 6 - Outdated C3 Diagram

Furthermore, Koen noted:  
“As of right now the Microsoft Graph SharePoint functionality is not present. There are some documents present in Pinecone, but that’s it. Also, only 1 LLM model is supported right now – OpenAI's ChatGPT”

As it appeared, the project’s Sharepoint functionality served just as a placeholder for future implementation and there was only one LLM model integrated. That meant that the C1 and C2 diagrams left behind the previous team were early drafts on how the system was first envisioned and were not entirely matching with the situation at the time.

## Conclusion:

The system’s architecture consists of a NestJS backend making use of Prisma as an ORM of a Postgres database, a React Frontend and a Pinecone vector database. There are 3 types of users – Administrators, Developers and Students.

Administrators would track the health and statistics of the CY2 system. Developers would integrate the system within their universities’ own infrastructure and students would use that integration to get assistance with their questions.

Some parts of how the system was originally envisioned are missing like the Oracle Digital Assistant and the Microsoft SharePoint integration. In the case of Oracle’s Digital assistant, these integrations were not even planned. It is worth noting that, while the Sharepoint integration is not yet present, yet we have decided that the relevant components of the C4 architecture ought to be kept due to the intended implementation as well as already existing components which aim to aid in the implementation of such feature such as specific entities already present within the database as well as existing, but not fully functional components (such as the MS Graph Repository component).

# Q2: How can an existing solution help with system monitoring? [Nikolay]

## Source 1: Available product analysis

System monitoring is a crucial aspect of maintaining the health, performance, and security of applications. With numerous tools available, choosing the right one can significantly impact the efficiency and effectiveness of the monitoring process. This chapter explores various alternatives for system monitoring, providing an analysis of their features and benefits. Ultimately, it will conclude with a justification for the selection of Sentry as the most viable option for CY2.

Since system monitoring is a common problem for many developers, it is possible for an already existing product to help resolve this issue. Searching on google for “monitoring for REST APIs” yield some results, with the most popular one being “Sentry”. Other alternatives which have been identified include New Relic, Datadog, the Prometheus and Grafana as a package and Splunk.  
We have first decided to go over the less popular alternatives.

**1. New Relic [8]**

New Relic is a comprehensive observability platform that offers detailed insights into application performance and infrastructure health. It provides end-to-end monitoring, including real user monitoring, application performance monitoring (APM), infrastructure monitoring, and synthetics. New Relic's robust dashboard and alerting mechanisms allow developers to quickly identify and resolve issues.

Pros: Extensive features for full-stack monitoring, real-time analytics, customizable dashboards, and robust alerting.

Cons: Higher cost, complexity in setup, and may be more feature-rich than necessary for smaller applications.

**2. Datadog [9]**

Datadog is another prominent player in the system monitoring space, offering a unified platform for monitoring, security, and analytics. It integrates seamlessly with various services and technologies, providing comprehensive metrics, traces, and logs. Datadog's strengths lie in its scalability and the ability to monitor cloud-based infrastructure efficiently.

Pros: Scalable, user-friendly interface, extensive integration options, and comprehensive observability features.

Cons: Pricing can become expensive with scale, and it may require significant configuration for optimal use.

**3. Prometheus [10] and Grafana [11]**

Prometheus, an open-source monitoring solution, coupled with Grafana for visualization, offers a flexible and powerful monitoring stack. Prometheus excels in collecting and querying time-series data, making it ideal for infrastructure and application monitoring. Grafana complements Prometheus by providing rich, customizable dashboards and alerting.

Pros: Open-source and free, highly customizable, strong community support, and excellent time-series data handling.

Cons: Requires more setup and maintenance, and integration can be complex compared to other solutions.

**4. Splunk [12]**

Splunk provides operational intelligence by collecting, indexing, and visualizing machine-generated data. It is highly effective for log management and analysis, offering real-time monitoring, search, and correlation capabilities. Splunk's strengths are its powerful search functionalities and scalability.

Pros: Powerful search and analysis capabilities, scalability, and strong support for log management.

Cons: High cost, steep learning curve, and potentially more suitable for larger enterprises.

**Sentry: A Focused Solution [2]**

After evaluating various system monitoring solutions, Sentry stands out as a focused tool for error monitoring and performance tracking, specifically tailored to meet the needs of the CY2 application.

**Features and Integration:**

Referring to their documentation, Sentry is a third-party system that captures errors during application runtime and stores them safely for developers to reference. It also offers performance monitoring and visualization tools. Integration with the system is technology-specific; for CY2, the NodeJS integration would be required.

**Dashboard and Monitoring:**

Once set up, Sentry provides a dashboard where different parts of the connected application can be monitored, depending on the specific setup. This allows developers to have a centralized view of the application’s health and performance.

**Error Capturing and Beyond:**

The most popular use of Sentry is error capturing and investigation. However, CY2 also requires insights about chatbot usage, referenced documents, and user feedback, which are quite specific and cannot be directly integrated into Sentry. Moreover, the product owner, Stefan, has specifically requested for these insights to be visible within the Frontend of CY2, rather than an external system. Nevertheless, for error logging and runtime exception monitoring, Sentry remains an invaluable tool.

While New Relic, Datadog, Prometheus with Grafana, and Splunk each offer robust monitoring capabilities, Sentry's specialized focus on error monitoring and performance tracking makes it an ideal choice for CY2. Its ease of integration with NodeJS, user-friendly dashboard, and effective error capturing features align well with the needs of the project. Therefore, Sentry is selected as the viable option for system monitoring in CY2, ensuring efficient tracking of application errors and performance issues.

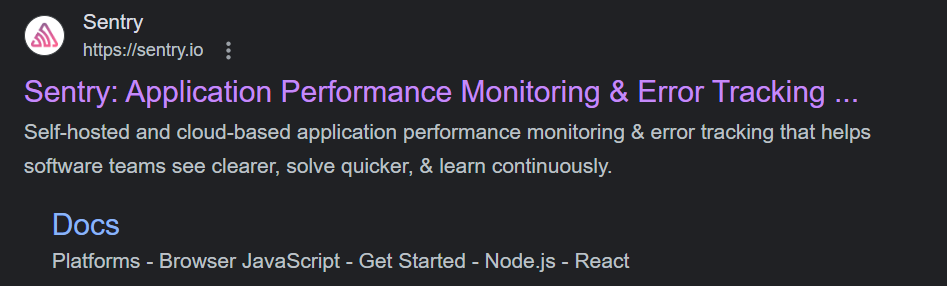


Figure 7 - Sentry website

Referring to their documentation, it appeared that Sentry was a 3rd party system that captures errors occurred during application runtime and stores them safely for the developers to reference. It also offers some performance monitoring and tools for visualization.

This tool does require integration with the system which is specific for each technology. In the case of CY2, the NodeJS integration would be required.

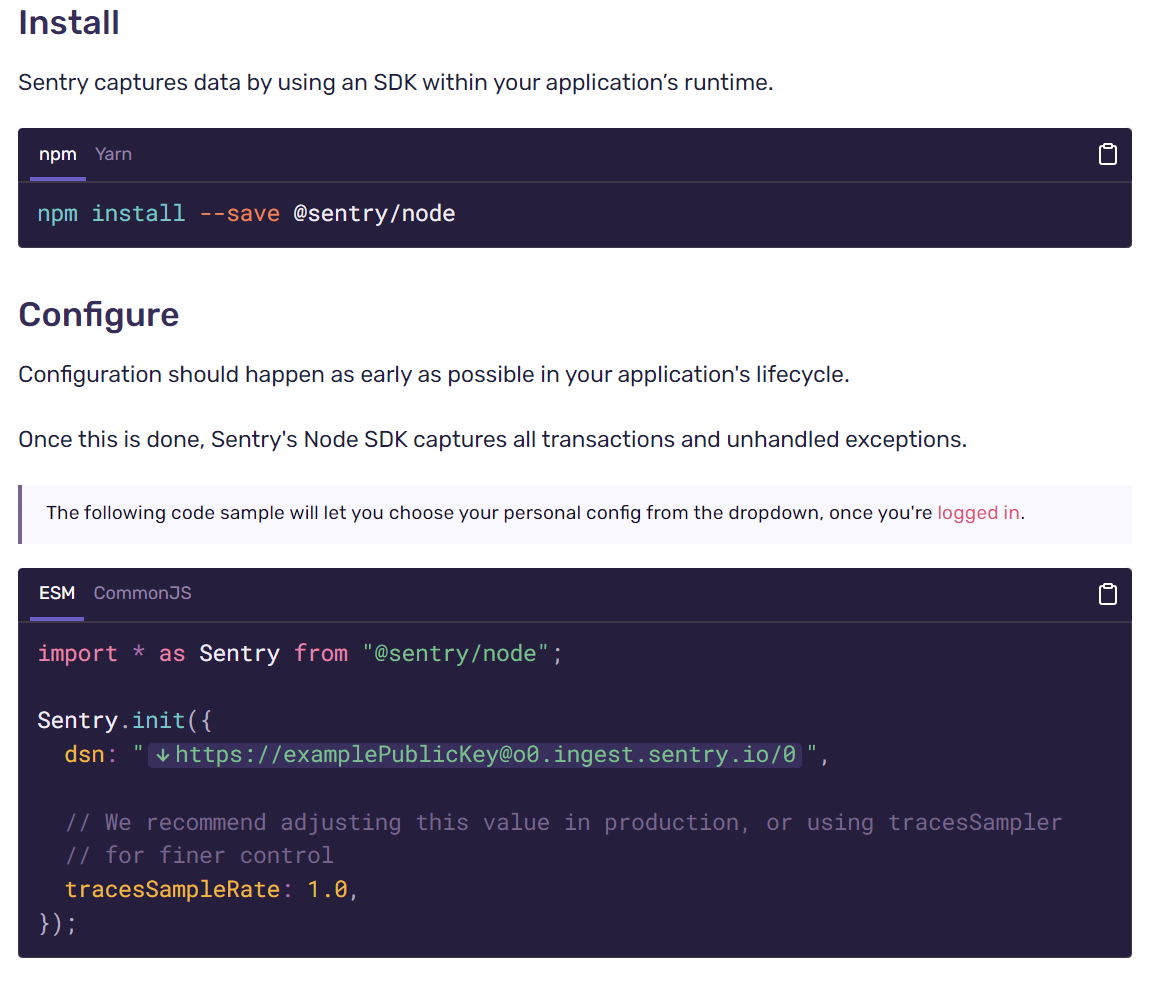


Figure 8 - Installation instructions for Sentry

Once set up, Sentry has a dashboard where different parts of the connected application can be monitored, depending on the specific setup.

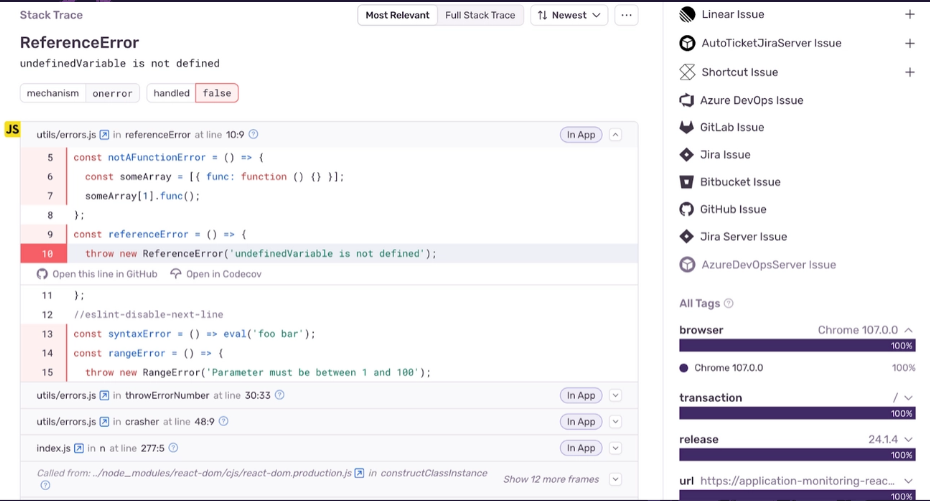


Figure 9 - Sentry error capturing

The most popular use of Sentry seemed to be error capturing and investigation, which in the case of CY2 is only a part of the problem. CY2 also requires a lot more insights about chat bot usage, referenced documents and user feedback, which are quite specific and cannot be directly integrated into Sentry. Moreover, the product owner Stefan has specifically requested for these insights to be visible within the Frontend of CY2, rather than an external system. However, in the scenario of error logging, Sentry could be used to monitor runtime exceptions.

## Source 2: Domain modelling

In order to see if Sentry could fit with the current project’s architecture, some domain modelling had to be done. The easiest way to portray this would be to modify the existing C1 diagram to include sentry as an external system.

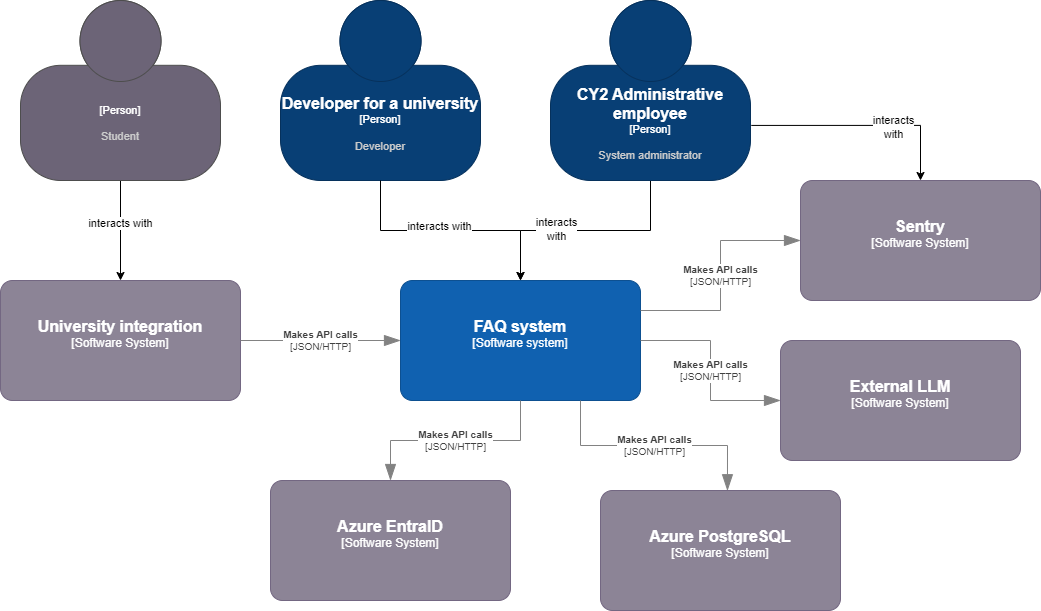


Figure 10 - C1 Diagram with integrated Sentry component

In this updated version of the C1 diagram(Figure 10 - C1 Diagram with integrated Sentry component), the FAQ system makes API calls to Sentry whenever an error occurs so that sentry can capture it. From there, the System Administrators would need to go to sentry directly to investigate the errors. While this flow is a possible solution, it makes it impossible for the administrators to reference the errors within the FAQ system itself. They would need separate accounts for Sentry, which also has a separate UI.

The benefits include the fact that the safekeeping of the errors does not need to be handled by the developers. There is an entire team of developers at Sentry taking care of safekeeping and making sure that their system is secure.

The drawbacks, however, are the fact that System Administrators of CY2 cannot directly reference captured errors in their own system. Instead, they have to go to an entirely different with different authentication and accounts to monitor the backend’s state.

## Conclusion:

In this project’s scenario, an external system like Sentry can help with application monitoring through the capture of runtime exceptions that happen while users are interacting with the system. There are both benefits and drawbacks to using an external system for error logging. In the end, it can be stated that during the early lifecycle of the project it might be considered as a non-necessity, but is the project to gain traction, an addition such as sentry as a monitoring tool will become indispensable.

# Q3: What is a good way to track system operation? [Tudor]

## Source 1: Good and bad practices

Effective system operation tracking is essential for maintaining security, performance, and reliability in any application. Various frameworks and best practices guide developers in implementing robust logging and monitoring systems. This chapter explores several notable guidelines and methodologies, including but not limited to OWASP, to ensure comprehensive system operation tracking for CY2.

**Alternative Guidelines and Best Practices**

**1. NIST (National Institute of Standards and Technology) [13]**

The NIST provides a detailed framework for logging and monitoring, particularly within its Computer Security Incident Handling Guide (NIST SP 800-61). This guide emphasizes the importance of maintaining detailed logs to support incident response and outlines key events to log, including user authentication, access control changes, and system errors.

* **Pros:** Comprehensive, government-endorsed standards, detailed guidance on incident response.
* **Cons:** Can be complex to implement, primarily aimed at larger organizations with extensive security needs.

**2. CIS (Center for Internet Security) [14]**

CIS offers a set of controls known as the CIS Controls, which provide a prioritized set of actions to protect organizations and data from cyber threats. Control 6 specifically deals with Maintenance, Monitoring, and Analysis of Audit Logs, offering practical steps to ensure effective logging and monitoring practices.

* **Pros:** Prioritized and practical, widely adopted across various industries.
* **Cons:** May require adaptation to fit specific application contexts.

**3. ISO/IEC 27001 [15]**

ISO/IEC 27001 is an international standard for information security management systems (ISMS). It includes guidelines on maintaining and monitoring logs to ensure the security of information systems. This standard covers the creation, protection, and retention of logs as well as the use of logging mechanisms to detect and respond to incidents.

* **Pros:** Internationally recognized, comprehensive coverage of information security practices.
* **Cons:** Implementation can be resource-intensive and may require certification.

**OWASP: A Trusted Source [16]**

As a certified source of truth, OWASP is an online community that produces freely available articles, methodologies, documentation, tools, and technologies in the fields of IoT, system software, and web application security. OWASP is also highly revered as one of the best, if not the practice by many developers worldwide.

Investigating their logging cheat sheet, they note that the best way to track system operation is to log different types of events. They can be logged to a file system or a database depending on the application context and requirements. Such logs help with identifying security incidents, monitoring policy violations, establishing baselines, etc.

The recommended events to log are as follows:

* **Input validation failures** e.g. protocol violations, unacceptable encodings, invalid parameter names and values
* **Output validation failures** e.g. database record set mismatch, invalid data encoding
* **Authentication successes and failures**
* **Authorization (access control) failures**
* **Session management failures** e.g. cookie session identification value modification
* **Application errors and system events** e.g. syntax and runtime errors, connectivity problems, performance issues, third-party service error messages, file system errors, file upload virus detection, configuration changes
* **Application and related systems start-ups and shut-downs**, and logging initialization (starting, stopping, or pausing)
* **Use of higher-risk functionality** e.g. network connections, addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens, use of systems administrative privileges, access by application administrators, all actions by users with administrative privileges, access to payment cardholder data, use of data encrypting keys, key changes, creation and deletion of system-level objects, data import and export including screen-based reports, submission of user-generated content - especially file uploads
* **Legal and other opt-ins** e.g. permissions for mobile phone capabilities, terms of use, terms & conditions, personal data usage consent, permission to receive marketing communications

In the case of CY2, these events could be logged to the database that is already integrated into the system. Afterwards, they can be referenced by System Administrators, so that the system can be monitored.

A bad practice, as described by OWASP, is to not log the correct events, which are described above. More bad practices also include:

* Access control violation for logged events
* Not having log indexes
* Incorrect log safekeeping

**Conclusion**

While NIST, CIS, and ISO/IEC 27001 offer valuable guidelines for system operation tracking, OWASP's detailed and widely respected best practices make it a highly suitable choice for CY2. OWASP provides a comprehensive framework for logging key events and maintaining the security and performance of applications. By adhering to OWASP guidelines, CY2 can ensure effective system operation tracking and mitigate potential security risks.

## Source 2: Expert interview (Marcel)

While Marcel was providing individual feedback for the group, he was also asked about the topic of tracking system operation. {ICAs he noted:

“Most projects usually keep logs with their own systems or leverage external systems specifically designed for it.”



When asked about what kind of logs the applications needs to keep, he replied:  
“The bare minimum of logging is security-related logs and error traces. You want to what’s going on with the system and who does what”

Afterwards, he mentioned that depending on the specific needs of the application, more information can be logged like incoming traffic and usage of specific features. These statistics could then be used by developers to improve the application, which aligns very well with CY2’s use case.

However, Marcel also noted that it’s important to keep logs the “correct way”. This would involve indexing them for increased performance while also deleting any unnecessary or processed logs. A choice also had to be made about where to store the logs – as files or within a database.

## Conclusion:

The system operation of CY2’s application can be monitored by keeping logs. These logs can be stored either in the file system or a database. With these logs System Administrators can keep track of the system and gain insights on its usage, security-related events and potential problems.

# Q4: What is an efficient way to visualize system operation data?[Kiril]

## Source 1: Research on Data Visualization Best Practices

A study conducted by Few (2009) [17] on data visualization best practices highlights the importance of selecting the right type of visualization to effectively communicate data insights. The study emphasizes that data should be presented in a manner that aligns with the user’s cognitive processes, enhancing comprehension and decision-making. Key findings from the research include:

* **Choosing the Right Chart Type:** Different data types and analysis goals require specific chart types. For instance, bar charts are ideal for comparing discrete categories, while line charts are best for showing trends over time.
* **Simplicity and Clarity:** Visualizations should avoid unnecessary complexity. Clear labeling, adequate spacing, and the use of contrasting colors can enhance readability.
* **Interactivity:** Interactive elements, such as tooltips, drill-down capabilities, and dynamic filtering, allow users to explore data in depth and derive more meaningful insights.

## Source 2: User interview

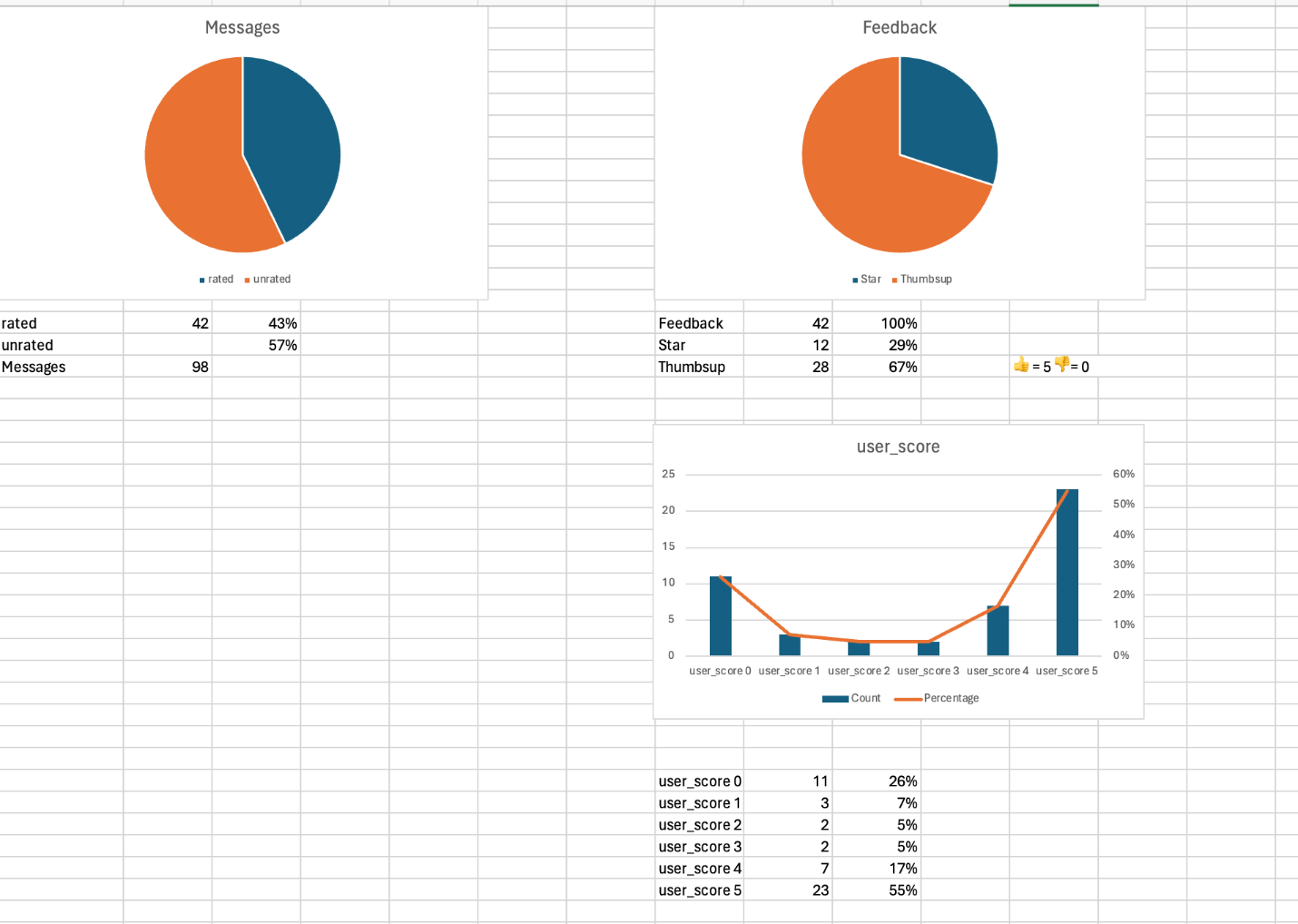
During the bi-weekly meeting with Stefan (product owner) and Koen, an opportunity arose to ask both of them about their ideas for visualizing system data and insights. Since both Koen and Stefan are System Administrators, they naturally had plenty of ideas on how to visualize the data in a manner that's easily readable.

The first thing they noted was that they want to have a general overview of the data. In the case of user feedback (Functionality of the system – the feedback that end users can leave to a chatbot answer; part of the insights which need to be visualized), for example, it would be nice to have a big, paginated table containing the feedback given from users. From there, an administrator can click on a row to see specific information about why the user gave that feedback, and if possible, the chat message which the feedback is for.

Furthermore, for data like API usage and referenced documents they instructed us to use suitable graphs (pie charts, bar charts, etc.) to visualize the general data. From there, once again, the administrator can view more details if he desires:

“It would be nice for the referenced documents graphic to show the name of the documents when hovered over. From there, I can choose a specific one and click on it. Perhaps a new page would open where I see exactly what was referenced in that document and why users like or dislike that section”

They gave an example of how data is visualized in one of their other software solutions, which can be seen below.



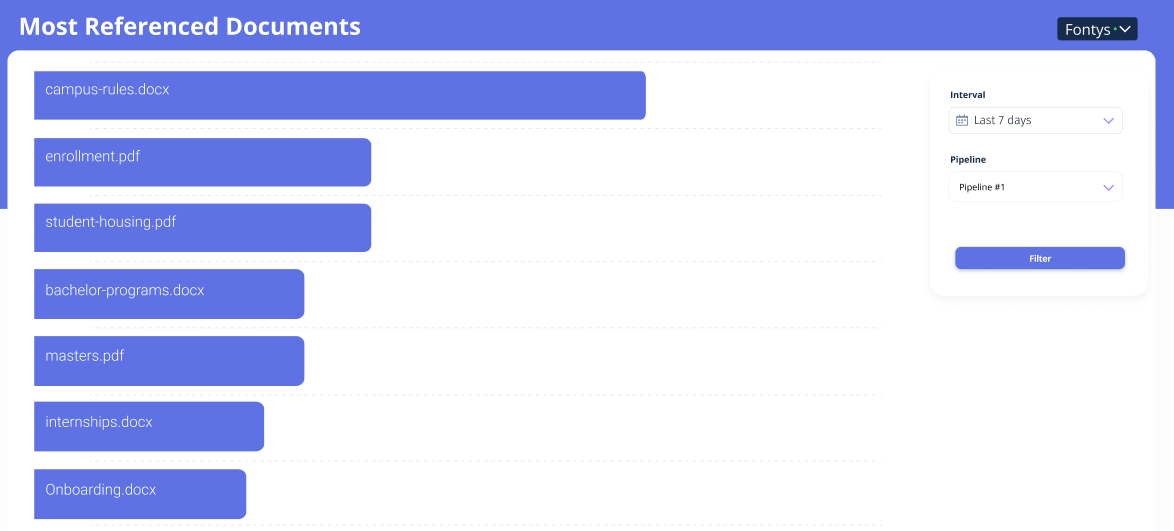
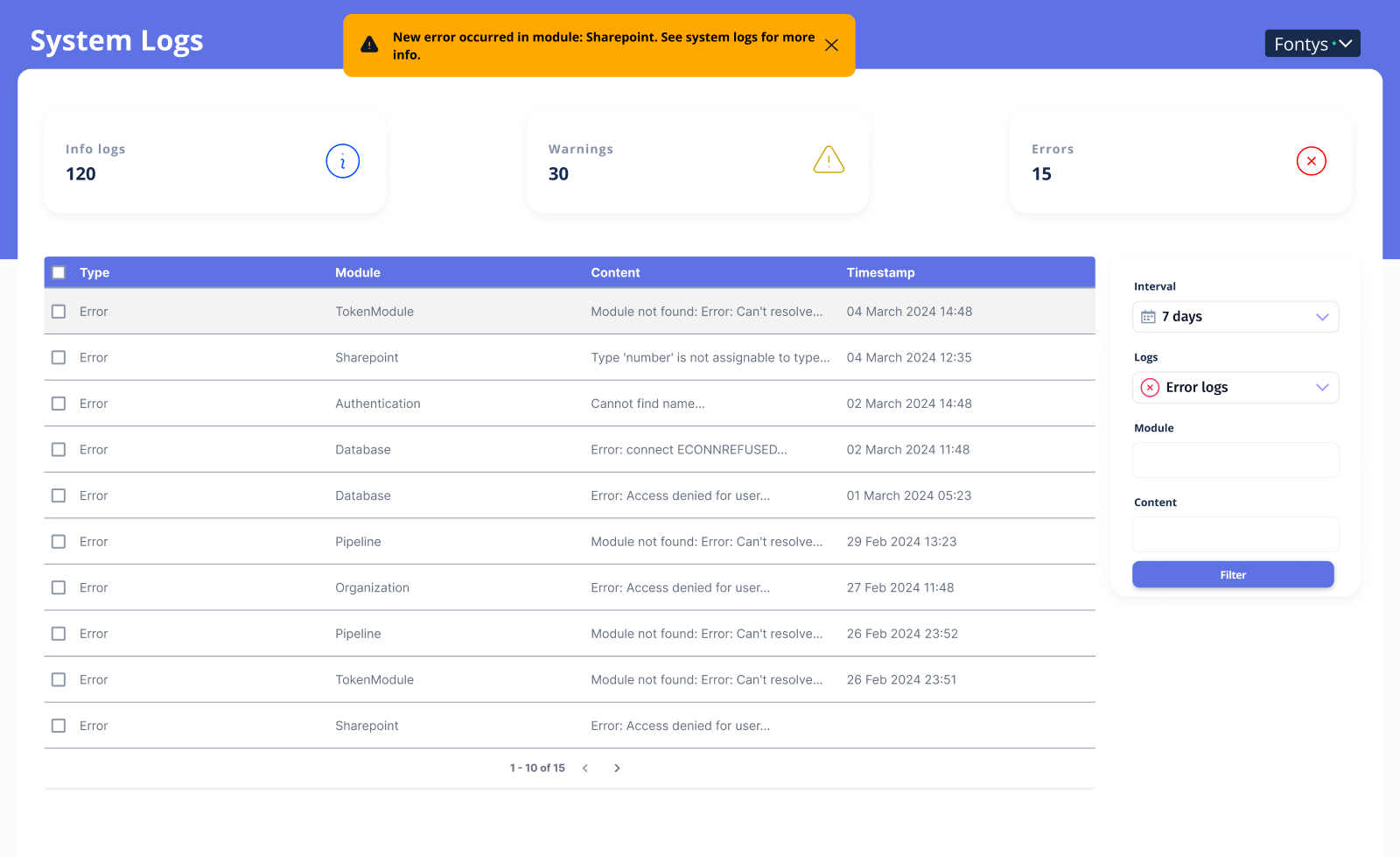
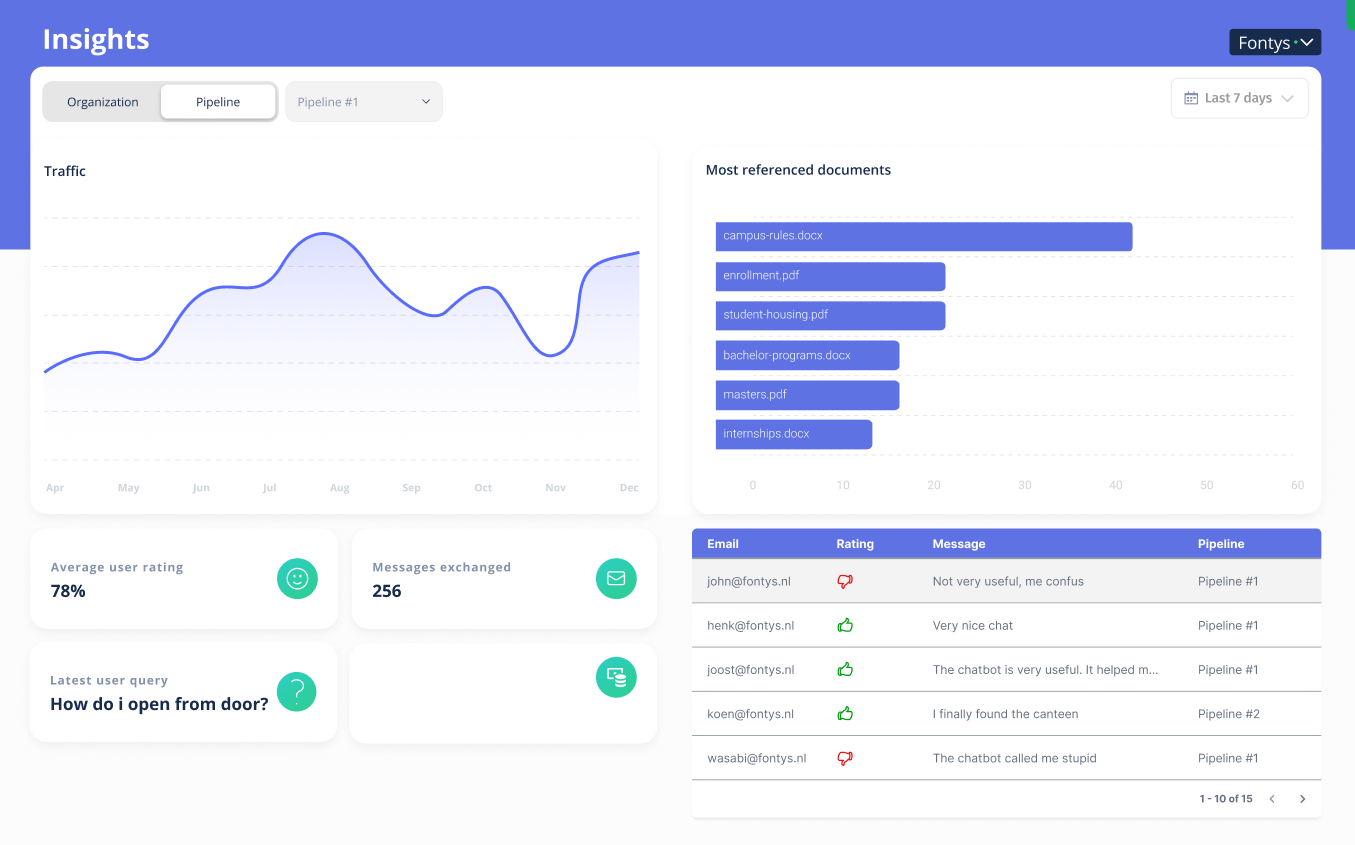
Furthermore, they noted that system logs should be visible only to System Administrators and should have different severities, like “info, warning and error”. The logs should be paginated and easy to search via filters, and easy to navigate and manage.

## Source 3: Prototyping

In order to see if the client’s ideas about data visualization can be fulfilled effectively, a Figma prototype was created by cooperation of the whole team. Such a prototype is very helpful and insightful for the following reasons:

* It gives both developers and the product owner a visual representation of what the system would look like, with the ability to give feedback and tweak any unwanted or unneeded parts. This ensures that the developers and the stakeholders are on the same page about features
* It allows the developers to better understand the technical complexities of the tasks that lie ahead and imagine how it would work under the hood.
* Most notably, it is the best way to visualize a small part of the system’s future data and allows developers to grasp whether it is the correct way to visualize the data and come up with ideas for types of graphs (bar chart, line chart, etc.)

With that being said, the Figma prototypes were drafted and approved by the CY2 stakeholders:



The designs went through a few rounds of feedback before they concluded in the screenshots seen above. In the end, it was decided that for most referenced documents, the best way to represent the data would be through a filterable bar chart.

For the API traffic page, we agreed that the most suitable solution would a filterable line chart, which represent the amount of messages sent for each day of the specified period.

As for the user feedback and system logs, the most sensible solution was to incorporate paginated tables, which made it easy for administrators and developers to navigate through the data.

## Conclusion

The best way to visualize data in the context of CY2’s project is to use established patterns for data visualization like different charts and graphs, complemented by user-centric design principles from research. Incorporating interactive elements and ensuring simplicity and clarity in visual design will facilitate better data comprehension and usability. By integrating feedback from stakeholders and adhering to research-based best practices, CY2 can implement an efficient and effective data visualization system.

# Q5: What is the quality and stability of the system [Yordan + Kiril]

## Source 1: Code review

While the team was browsing the source code of the project in GitHub, some very questionable code was spotted. It was agreed between team members that a code review is needed to assess the system's current quality.

The first questionable code came from the backend’s services. In several scenarios, the source code contained improper use of the dependency injection pattern that goes against all SOLID principles. It can be seen in the code snippet below:

constructor(

private readonly configService: ConfigService,

private readonly prismaService: PrismaService,

) {}

async inferPipeline(

pipelineId: number,

rawToken: string,

question: string,

history: string[],

) {

// validate token

const accessTokenService = new AccessTokenService(this.prismaService);

const hash = createHash('sha256');

hash.update(rawToken);

...

As can be seen, the AccessTokenService is not injected through dependency injection and is just instantiated on the spot, even though it is already injected in the class containing this code. This is considered wasteful, as NestJS provides automatic dependency injection and management through their modules.

In this example, an AccessTokenService is created specifically for this method through the `new` keyword, which creates a new object that needs to be stored in memory. This is detrimental for performance and is an anti-pattern according to the SOLID principles. It also makes testing more difficult, as more effort needs to be put in mocking this extra instance.

There are multiple examples in the backend where this type anti-pattern is performed. The correct way to handle this would be to use NestJS’ dependency injection and declare the accessTokenService in the constructor, after which Nest would automatically inject the class.

constructor(

private readonly configService: ConfigService,

private readonly prismaService: PrismaService,

private readonly accessTokenService: AccessTokenService

) {}

async inferPipeline(

pipelineId: number,

rawToken: string,

question: string,

history: string[],

) {

// validate token

const hash = createHash('sha256');

hash.update(rawToken);

accessTokenService.getToken(rawToken);

...

Apart from the anti-pattern dependency injection used throughout the backend, the rest of the code was of fair quality [18]. There were some cases where smarter queries could be made in a more efficient way, but that was outside of the scope of the project. The idea was to first fulfill the functional requirements and focus on non-functional requirements like performance later.

NestJS is a highly opinionated framework which relies on a predetermined modular architecture and strongly encourages developers to stick to it. However, this is not immediately familiar for new Nest developers, so we came to the conclusion that the previous team that worked on the project was new to NestJS and was not very familiar with its intricacies.

## Source 2: System test

System stability is usually inspected through load testing. For the purposes of load testing the CY2 backend, it was decided to use Artillery to make GET requests for the system logs. The code to create artificial load on the backend created more and more virtual users gradually, where each user was set up to make 100 get requests to the specified endpoint.

config:

  # This is a test server run by team Artillery

  # It's designed to be highly scalable

  target: http://localhost:4000

  phases:

    - duration: 30

      arrivalRate: 10

      rampTo: 30

      name: Spike phase

  plugins:

    ensure: {}

    apdex: {}

    metrics-by-endpoint: {}

  apdex:

    threshold: 100

scenarios:

  - flow:

      - loop:

        - get:

            url: "/api/systemlogs?page=1"

        count: 100

Running the test, the following result could be seen:

A screen shot of a computer

Description automatically generated

Figure 11 - Load testing results

The backend managed to fulfill 200 requests/sec, which is a respectable amount. However, in the context of an enterprise system, it would not be enough to serve responses to all users and the system would crash. This can be simulated by increasing the amount of actions per virtual user to 500. As a result, the backend crashes due to the overwhelming load.

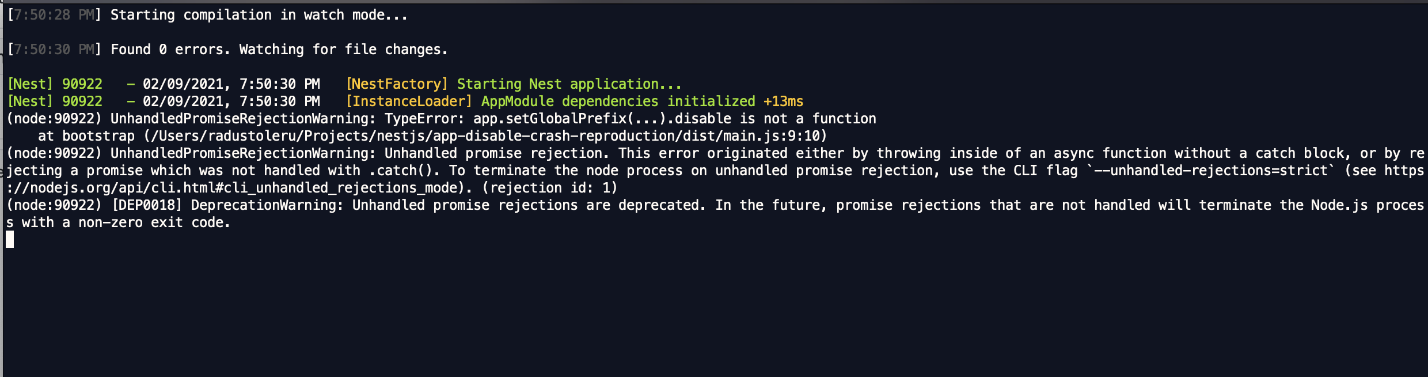


Figure 12 - Backend Crash log

In order to increase the performance, an optimization strategy was deployed to the logs through indexing. By indexing the logs located in the database, they could be found faster, which meant that the application was capable of responding faster, and therefore handling more requests/sec. The solution involved placing an annotation where the log model is declared and specifying an index key. Afterwards, the ORM would take care of the indexing.

@Index('systemLog')

model Log {

  id                                Int                         @id @default(autoincrement())

  type                              log\_type                    @default(error)

  createdAt                         DateTime                    @default(now())

  expiresAt                         DateTime

  message                           String

  module                            String

}

After making this change and performing the load test again, the backend could handle 1478 requests/sec before it started throttling.

A black background with many dots

Description automatically generated

Figure 13 - Load testing Results change

Conclusion:

This sub question had the role of studying the performance of the system by analyzing the quality of the code as well as its performance. In that sense, the quality and stability of the system is a mixed bag. There are some questionable coding decisions taken that do not conform to the SOLID principles. Moreover, there are no optimizations in the backend for handling a large number of concurrent users and stability cannot be guaranteed.

Given that changing the already existing implementation is out of scope for the current assignment, the preferred course of action is to continue the implementation making use of best practices as well as making improvements where the current implementation’s interest and outdated code align. This subchapter provided desirable insights which can aid further development of the application, making the identified issues known.

# Q6: How can the system be improved further [Nikolay + Tudor]

## Source 1: Peer review

Whenever a certain feature is finished being developed, there is always a code + peer review by members of the team. This allows for code quality to stay high and to ensure that the developed features align with the expectations marked in the GitHub issues. For example, the pull request about implementing the user feedback page can be inspected.

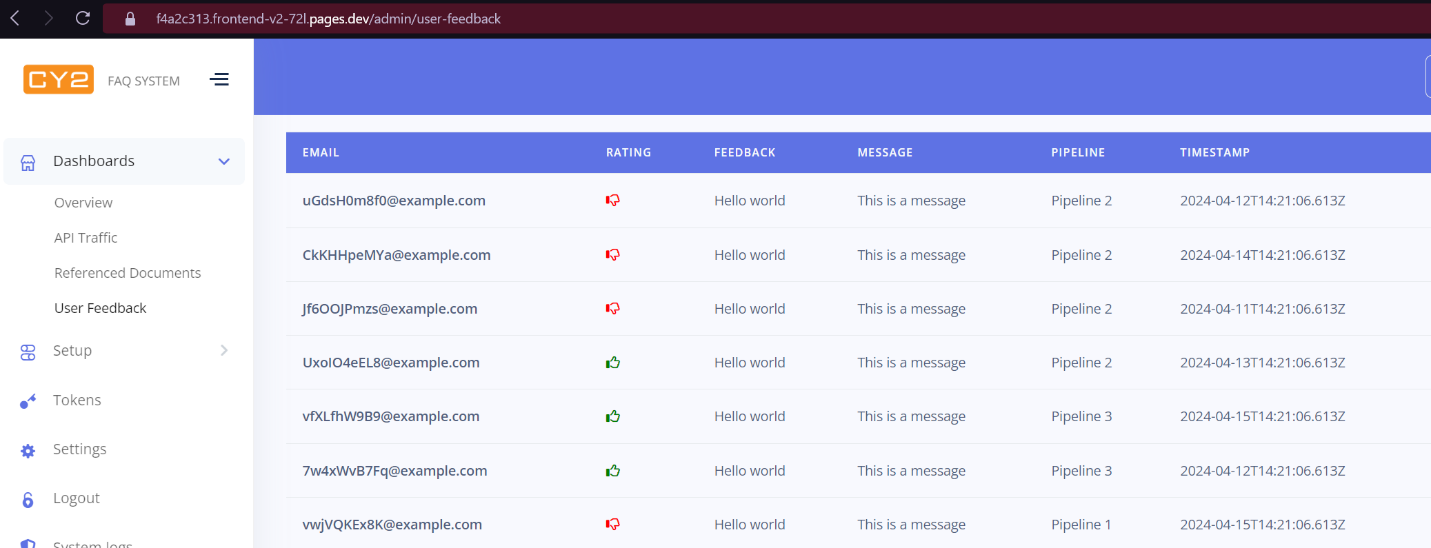


Figure 14 - User Feedback page

Here, it was not possible to click on a specific piece of user feedback to learn more about the details. Also, in terms of user experience, the table had no hover effect, which made it hard for the user to see where their cursor was.

As a result, a peer review was performed between the developer and other team members.

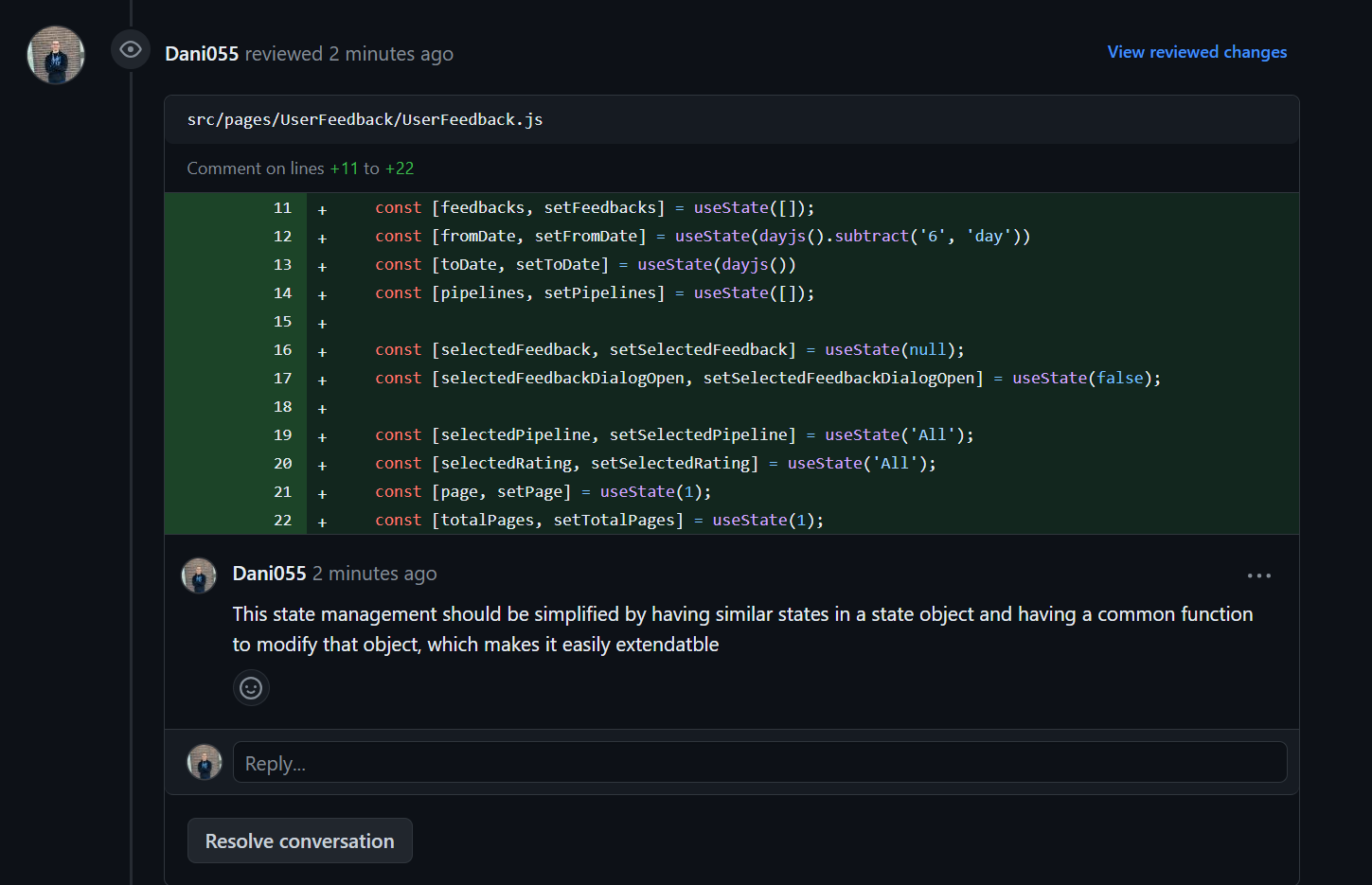


Figure 15 - Code reviews carried out within the team

The review pointed out the possible improvements that can be made to further satisfy the client.

Another point of improvement could also be made regarding the system logs. In their implementation at the time, the table was a bit confusing for the user.

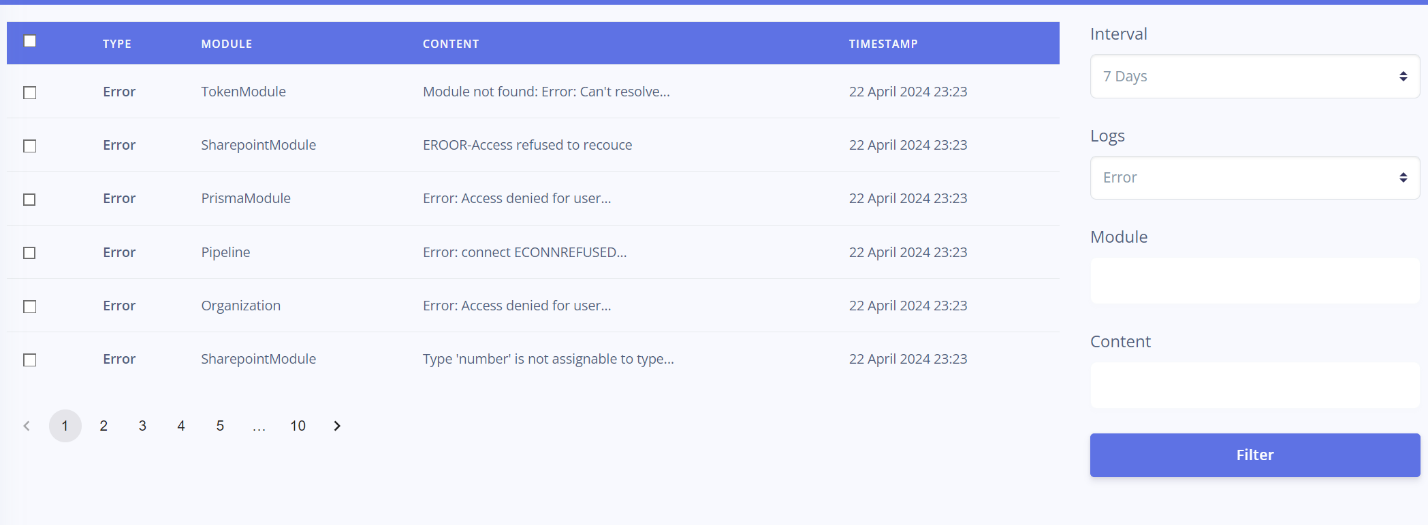


Figure 16 - Error logging page

Team members noted that it is hard for the user to differentiate between the different kinds of logs and it would be nice to include different coloration, as well as icons on the log type.

## Source 2: Product review

During the bi-weekly meetings with the product owner, whenever applicable, there is always a product review in order to validate the team’s progress and make sure the project aligns with the client’s expectations.

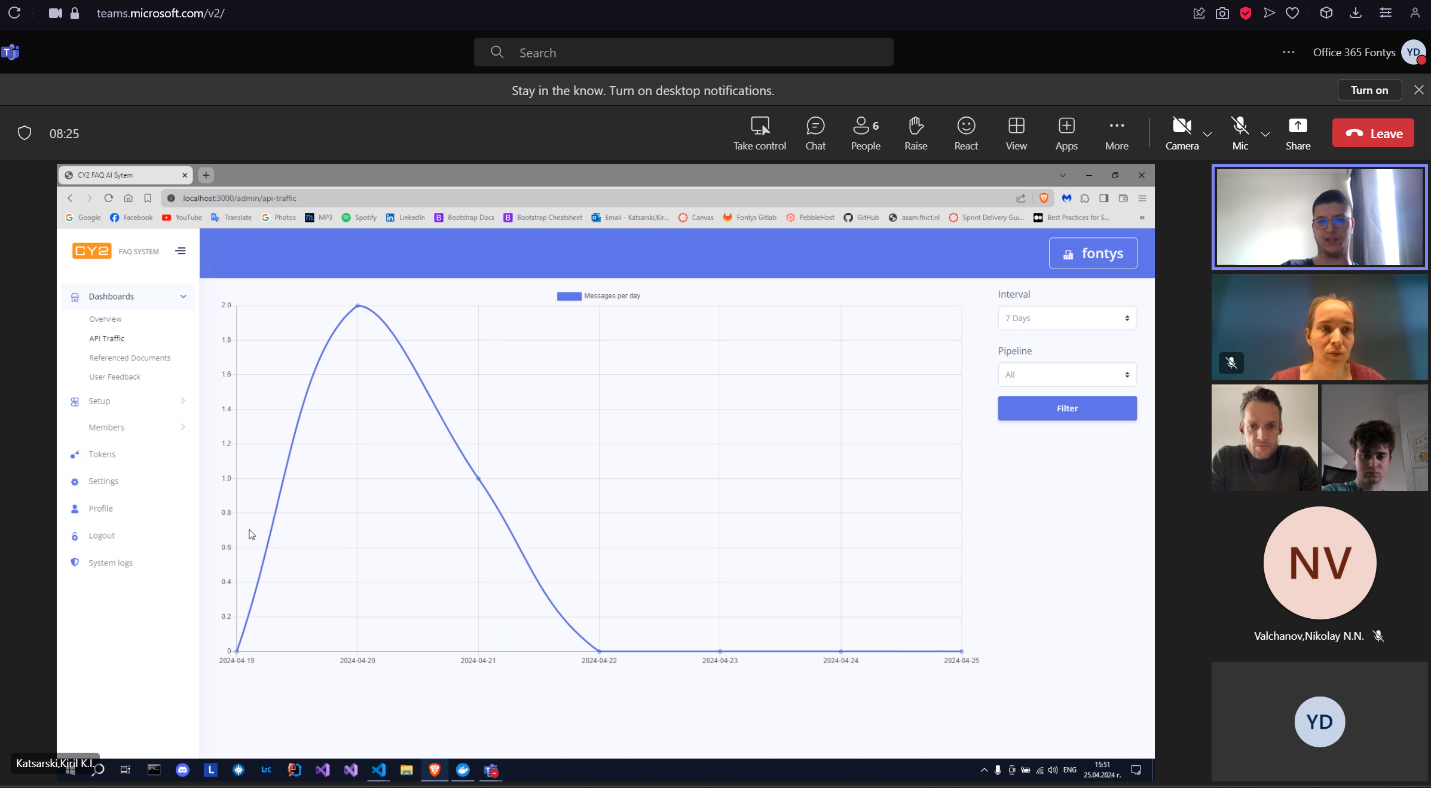


Figure 17 - API Traffic demo

In the last product review, Stefan (product owner) made remarks about the “referenced documents” page. He noted that it would be nice to see the amount of positive and negative feedback in the bar itself, otherwise the data for which document is referenced is not too useful on its own.

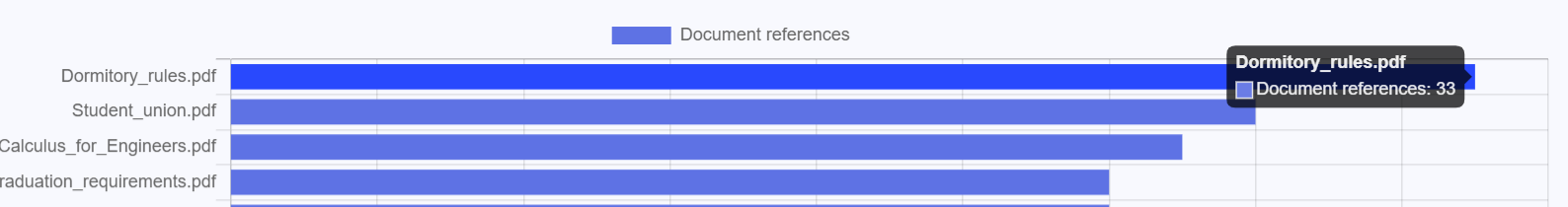


Figure 18 - Referenced Documents Table

From there, a developer could have the ability to click a bar and have an overview of the different feedback given by users, which would be a more useful insight.

Furthermore, Koen instructed us that it would be beneficial to set up review environments for the different pull requests on GitHub. That way it would be easy to conduct future peer + code reviews between teammates, while also allowing for easy demos when there is a product review.

## Conclusion:

At the time of writing this research, the application can be further improved by conducting regular peer review among team members. This would be facilitated by setting up different review environments for pull requests on GitHub. This is of ***utmost importance*** especially when taking into account the previous sub question of the research. This implies that, when code is reviewed, the reviewer will also be responsible with checking outdated code which does not respect fair quality[18] standards, while the reviewee is responsible for fixing any encountered issue. This approach can slowly fix the issues encountered in code quality without having to designate a specific task to rewrites at the cost of time. Is the popularity of the system to gradually increase, this drawback would be negligible. In the case of rapid increase in popularity, specific tasks related to code improvement will have to be undertaken.

Furthermore, the product owner has given specific feedback regarding how to make the “referenced documents” section more useful, by incorporating the user feedback into the bar chart, along with a more detailed view on that specific document’s feedback.

# Conclusion

**How do we allow administrators of CY2 and organization developers to gain more insight about the system operation?**

The research conducted has thoroughly addressed the central question: "How do we allow administrators of CY2 and organization developers to gain more insight about the system operation?"

To enable administrators and developers to gain comprehensive insights into system operations, a multi-faceted approach has been identified as essential. This approach involves the integration of robust system monitoring tools, adherence to best practices for logging and tracking, and the implementation of effective data visualization techniques.

In addition to this, keeping the newly added implementations to a high quality standard as well as fixing identified issues related to relevant implementations should provide a great solution to the project’s main goal as well as set a standard for future development, action which aims to seamlessly address any existing issue.

In summary, by integrating these elements—advanced monitoring tools, rigorous logging practices, and intuitive data visualization—CY2 can provide its administrators and developers with the comprehensive insights needed to maintain optimal system performance and security. This holistic approach not only addresses the immediate requirements but also sets a foundation for continuous improvement and scalability in managing system operations.

# References:

1. *CY2Project*. (n.d.). GitHub. <https://github.com/CY2Project/>
2. *Sentry Docs | Application Performance Monitoring & Error Tracking Software*. (n.d.). <https://docs.sentry.io/>
3. *Logging - OWASP Cheat Sheet Series*. (n.d.). <https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html>
4. *Build software better, together*. (n.d.). GitHub. <https://owasp.org/www-pdf-archive/OWASP_Logging_Guide.pdf>
5. *Artillery.io | Cloud-scale load testing*. (n.d.). artillery.io. <https://www.artillery.io/>
6. Vogel, J. (n.d.). *ICT Research Methods — Methods Pack for research in ICT*. ICT Research Methods. <https://ictresearchmethods.nl/>
7. *Group 2 LLM insights – FHICT Canvas* (n.d.). <https://fhict.instructure.com/courses/13669/assignments/234579/submissions/23164>
8. New Relic, Inc. (n.d.). *Monitor, debug and improve your entire stack*. New Relic. https://newrelic.com/
9. Datadog. (2016, July 14). *Cloud monitoring as a service | DataDog*. Datadog. https://www.datadoghq.com/
10. Prometheus. (n.d.). *Prometheus - Monitoring system & time series database*. https://prometheus.io/
11. Grafana Labs. (n.d.). *Grafana: The open observability platform | Grafana Labs*. https://grafana.com/
12. *Splunk | The key to enterprise resilience*. (n.d.). Splunk. https://www.splunk.com/
13. *NIST*. (2024, May 23). NIST. https://www.nist.gov/
14. *CIS*. (n.d.). CIS. https://www.cisecurity.org/
15. *ISO/IEC 27001:2022*. (n.d.). ISO. https://www.iso.org/standard/27001
16. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. (n.d.). https://owasp.org/
17. Few, S. (2009). *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press
18. SonarSource. (2024, March 19). *code quality developer’s guide*. Sonar. https://www.sonarsource.com/learn/code-quality/