V1.1

Research Report

Real Time Translation

Project for Nidaros

GROUP: IT-2B

Table of Contents

Version control			
Version	**Author**	**Date**	**Changes**
0.1	Iarina Dicu	22-10-2025	First draft created.
0.2	Team	29-10-2025	Added full content for chapters and research cards.
1.0	Team	30-10-2025	Final version for submission.
1.1	Team	17-11-2025	Updated explanations and summary. Added pictures and references.
1.2	Cosmin Buzic	**25-11-2025**	Final review. Fixed missing MPEG abbreviation, fixed spelling mistakes, removed duplicated research question foot note.

Table 1: Version control

Table Of Abbreviation	
Abbreviation	**Meaning**
API	Application Programming Interface
RTMP	Real-Time Messaging Protocol
RTSP	Real-Time Streaming Protocol
HLS	HTTP Live Streaming
STT	Speech-to-Text
FFmpeg	Fast Forward MPEG (multimedia framework)
MPEG	International standard for encoding and compressing video images

PoC	Proof of Concept
WCAG	Web Content Accessibility Guidelines
WebVTT	Web Video Text Tracks

Table 2: Table of abbreviations

1. Introduction

1.1 Background of the project

This project focuses on improving a live streaming platform of Nidaros that distributes television and video content over the internet. The platform allows broadcasters to share live programs with viewers online.

Although the platform offers reliable live streaming, currently does not support real-time subtitles for live and play back broadcasts. This creates some problems for viewers who have impaired hearing as well as for international viewers who do not speak the language used in the broadcast.

The goal of this project is to make the platform more accessible and user-friendly by introducing live subtitles, including automatic translation into multiple languages (English, Ukrainian, Arabic, Farsi, Dutch). This means that viewers will be able to follow live content even if they have hearing difficulties or do not understand the original spoken language.

By adding this feature, the platform can reach a broader and more diverse audience and finally meet the client's goal which is bring the society closer together.

1.2. Background of the company

Nidaros was founded by Gerben Dolsma after being asked to act as an implementation partner of Opalis (now Microsoft Orchestrator). The company was established during the large-scale migration of Postbank and ING, laying a solid foundation in process automation and IT integration (Nidaros, 2025).

Today, Nidaros operates from offices in Hoogeveen and Amsterdam, enabling it to serve a broad region within the Netherlands and Europe. The permanent team of employees is supported by external consultants when needed.

Nidaros specializes in helping organizations integrate and optimize business processes using proven technologies. Its focus lies in automating repetitive tasks, effectively providing “virtual employees” that take over routine work. This allows clients to reduce inefficiency, improve consistency, and free up their workforce for higher-value activities. The company’s competitive strength is built on unburdening customers by delivering scalable and reliable automation solutions.

2. Fishbone Diagram

The Fishbone diagram (also known as an Ishikawa diagram) in Figure 1 was used to determine the probable causes of this project's problem: Non-native speakers and hearing-impaired users can't follow livestream; by organizing contributing factors into four main categories: Material, Method, Manpower, and Machine.

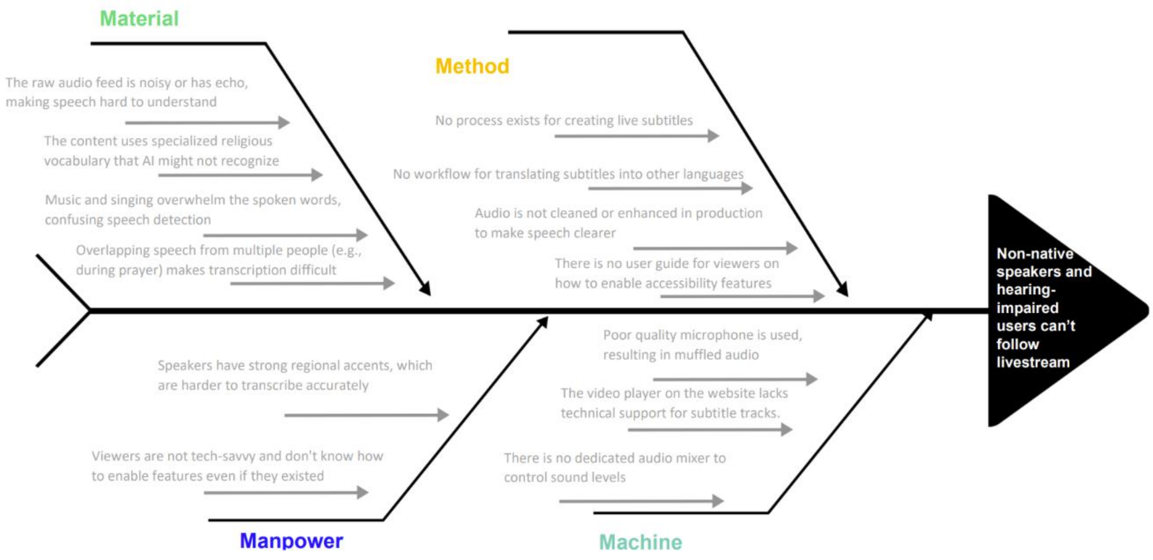

Material
The raw audio feed is noisy or has echo, making speech hard to understand
The content uses specialized religious vocabulary that AI might not recognize
Music and singing overwhelm the spoken words, confusing speech detection
Overlapping speech from multiple people (e.g., during prayer) makes transcription difficult
Method
No process exists for creating live subtitles
No workflow for translating subtitles into other languages
Audio is not cleaned or enhanced in production to make speech clearer
There is no user guide for viewers on how to enable accessibility features
Non-native speakers and hearing-impaired users can't follow livestream
Speakers have strong regional accents, which are harder to transcribe accurately
Viewers are not tech-savvy and don't know how to enable features even if they existed
Manpower
Poor quality microphone is used, resulting in muffled audio
The video player on the website lacks technical support for subtitle tracks.
There is no dedicated audio mixer to control sound levels
Machine

Figure 1: Fishbone Diagram

Below is a breakdown of the causes that have been identified within each of the categories:

The 4Ms Explained

Material: This category addresses the issues with raw content and audio feed.

- The raw audio feed is often noisy or contains an echo, making speech difficult to understand.
- The content itself can be a challenge, as it may include specialized religious vocabulary that an AI transcription service might not recognize.
- Overlapping speech from multiple people (e.g., during prayer) makes transcription difficult.
- Music and singing overwhelm the spoken words, confusing speech detection.

Method: This category highlights the lack of existing processes and workflows.

- This is a primary cause: there is currently no process in place for creating live subtitles.
- Similarly, no workflow exists for translating those subtitles into other languages.
- The audio is not cleaned or enhanced in production to make speech clearer for transcription.
- There is no user guide to instruct viewers on how to find or enable accessibility features.

Manpower: This category focuses on the human factors involved from both the speaker and viewer sides.

- Speakers may have strong regional accents, which are historically more difficult for AI models to transcribe accurately.
- Viewers, particularly in the older demographic, may not be tech-savvy and would not know how to enable or select subtitle features even if they existed.

Machine: This category details the technological and hardware limitations.

- Poor-quality microphones are used, resulting in muffled or distant audio.
- The current video player on the website lacks the technical support or infrastructure required to display subtitle tracks.
- There is no dedicated audio mixer to properly balance sound levels (e.g., lower music volume when speech is present).

Conclusion

The Fishbone diagram makes it clear that the problem is not isolated but a result of interconnected issues in technology, process, and raw content quality.

The most significant and actionable causes fall under the Method and Machine categories. There is a lack of technical “Machine” (a video player that supports subtitles) and the "Method" (a workflow for transcription and translation).

The analysis directly leads to the research question of this project. Since Nidaros cannot easily control the "Material" (audio noise, accents) or "Manpower" (viewer tech skills), the most effective solution must focus on overcoming the "Method" and "Machine" deficits. Therefore, the research is focused on creating this missing system.

This connects to the research question presented in the next chapter:

“How can we implement a real-time transcription and translation solution that enables all viewers, including non-native speakers and hearing-impaired users, to comprehend[1] the live stream content?”

[1] * Comprehend means that they are able now to see the translation of the live stream content in real-time

3. Research Question

The core problem addressed in this project is that non-native speakers and hearing-impaired users cannot follow the church live streams distributed by Nidaros because they do not understand what is being said. This makes it harder for Nidaros to reach a larger audience and makes the platform less accessible. The Fishbone diagram makes this dilemma clearer by demonstrating how factors like missing real-time transcription, lack of translation assistance, technological problems and limited accessibility features all contribute to the problem.

To investigate a suitable solution to this problem, we have formulated the following research question:

How can we implement a real-time transcription and translation solution that enables all viewers, including non-native speakers and hearing-impaired users, to comprehend the live stream content?

4. Research Pattern

Pattern: How do I choose the right technologies for my project?

This research pattern guides the way we approached the technical and organizational decisions required for developing the Real-Time Translation system. The pattern includes several fields that help structure the research process: Library, Field, Workshop, Lab. Although these elements are explained in detail later in their corresponding cards, here we briefly describe how they shaped our choices.

Problem field – Why this pattern?

Our project required integrating multiple systems (Wowza, Whisper/Faster-Whisper, Machine Translation, WebVTT subtitles, WebSocket API, frontend video Player). Because all these must work together in real time under 8 seconds of latency, we needed a pattern that helps evaluate technologies and approaches systematically. This field guided us in identifying the main risks: performance limitations, integration effort, accessibility requirements, and user expectations.

Solution Strategy field – How we approached it

Using this pattern, we selected a mix of research actions that together reduce uncertainty: exploring available technologies, understanding user needs, validating architecture through modelling, and testing ideas early through prototypes and simulations. Each chosen research method (card) supports one aspect of the solution strategy. For example: Literature Study for comparing tools, User Requirements for aligning with real needs, Prototyping for validating assumptions, and System Testing for performance verification.

Research Activities field – Which cards we chose and why

The pattern led us to select seven research cards to collect all information needed for an evidence-based technical decision:

- Literature Study Card: to compare existing STT/translation technologies and choose the ones that fit our environment.
- Explore User Requirements: to understand real user expectations (Dutch viewers, non-Dutch speakers, hearing-impaired users, older users).
- Domain Modelling: to understand system structure, data flow and module interactions.
- Stakeholder Analysis: To identify interests, influence, and expectations from Nidaros and end users.
- Prototyping Card: test assumptions quickly and understand early technical limitations.
- Computer Simulation Card: to test the entire pipeline in a containerized environment.
- System Test Card: validate behavior, latency, and reliability of the integrated system.

Expected Results field – What this pattern gave us

The chosen cards ensured that every aspect of the problem was examined from a different angle: user needs, technical feasibility, architectural clarity, stakeholder expectations, and system performance.

This created a structured foundation for decision-making and helped us avoid technological misalignment, unrealistic assumptions, and integration challenges.

5. Cards

5.1. Literature Study Card

Before building the project, we searched for free and open-source tools that could add subtitles and translations to church livestreams. We used Whisper for converting speech to text, and Argos Translate for translating text into different languages. We also learned about Docker and .NET, since our team didn't know those tools at the start.

Purpose of the Research

- We wanted solutions that don't cost money and work on our servers, so we focused on open-source software.
- We looked for tools that were easy to set up and could handle live speech for different languages.
- We made sure the programs we picked helped people who couldn't hear or understand the stream.

How the Research was Conducted

We conducted a systematic literature review using the following approach:

Database and Source Selection:

We searched GitHub repositories, official documentation, academic benchmarks, and technical forums to find open-source STT and translation tools.

Search Keywords:

"open-source speech-to-text", "real-time transcription", "offline machine translation", "Docker containerization API integration"

Evaluation Criteria:

- Cost: Must be free and open source
- Performance: Low latency suitable for real-time processing (under 8 seconds)
- Accuracy: Low Word Error Rate (WER) for transcription and acceptable translation quality
- Compatibility: Must integrate with .NET backend and Docker containers
- Language Support: Must support Dutch, English, and other required languages

Comparison Method:

We created comparison tables to evaluate different tools based on performance benchmarks, accuracy metrics, resource requirements, and community support.

Results of the Research

Speech-to-Text Technologies

We evaluated several open-source STT options and found the following:

Tool	Word Error Rate (WER)	Real-Time Processing	Language Support	License	Source
OpenAI Whisper (Large v3)	9.0% (Common Voice)	Yes (RTF < 1.0)	57+ languages	MIT	https://github.com/ChocolateMagnate/speech-to-text-benchmarks

Table 3: STT API comparison.

Selection Justification: We selected **OpenAI Whisper** because it achieved the best balance between accuracy and real-time performance across multiple benchmarks. Due to its popularity, it also has great documentation and support which will speed up the development process. Additionally, Whisper supports 99+ languages including Dutch, which is critical for our use case.

Machine Translation Services

We compared open-source translation tools:

Tool	Technology	Offline Support	API Integration	Performance	Source
Argos Translate	OpenNMT, SentencePiece	Yes	REST API, Python library	Moderate (30s for large docs)	https://github.com/argosopentech/argos-translate

Table 4: MT API comparison.

Selection Justification: We selected **Argos Translate** because it is specifically designed for offline, self-hosted translation without reliance on proprietary services like Google Translate or DeepL. While performance can be slower for larger documents, it meets our requirement for free, open-source translation that preserves user privacy.

5.2. Explore User Requirements Card

Purpose of the Research

The purpose of the Explore User Requirements research was to understand exactly what the target audience needs from the Real-Time Translation system on the Nidaros MIJNIPTV platform. As this system has been designed to make live church streams more accessible to non-Dutch speakers and the hard of hearing, it was important to gather user expectations and translate them into concrete functional and non-functional requirements. This research ensured that the final design reflected real-world usage, and accessibility needs directly rather than assumptions made by the development team.

How the Research was conducted

To develop the system, we conducted a research process based on primary and secondary data analysis.

Identification of Target Personas: Target user types were identified at the outset, including Dutch viewers, non-Dutch speakers, hearing-impaired viewers, older people and returning viewers. These were documented in the requirements analysis to understand their various motivations and access constraints.

Requirements gathering via scenario-based interviews and observation: We discussed with stakeholders and example end-users how people currently interact with the MIJNIPTV interface, and the ways in which it prevents them from fully enjoying live streams. Some of the usability barriers discovered included difficulty in enabling subtitles, missing translations, and delayed captions.

Translation to user stories: These findings were then translated into user stories in the Requirements Catalogue, such as: As a non-Dutch viewer, I want to be able to select English subtitles so that I can follow the live stream. Each story highlights who the user is, what they want and why. These became the starting point for Jira tasks and the project's functional requirements.

Prioritization and validation: This was done in collaboration with the client and technical leaders, and requirements were prioritized.

Result of the Research

- The outcome of the research was a clear, structured catalogue of requirements that related user needs to system functionalities.
- Key insights included:
- Accessibility: Subtitles should be readable and consistent and adhere to all WCAG guidelines.
- Performance: Subtitles should appear no later than eight seconds after live speech begins.
- Usability: Provide users with an intuitive Toggle subtitles/select language option within JW Player.
- Personalization: The system should remember subtitle preferences across sessions.
- Reliability: The system should handle translation errors in a user-friendly way with helpful notifications rather than failing entirely.

What we Learned

These findings have directly influenced our design document and informed architectural decisions such as multi-language subtitle support, font customization and session-based preference storage.

Reliability: Translation errors should be handled with friendly notifications instead of system failures.

These findings have directly influenced the design document and informed architectural choices such as multi-language subtitle support, customizable fonts, and session-based preference storage.

5.3. Domain Modelling Card

Purpose of the Research

The domain modelling research focused on identifying and representing the main entities and relationships within the Real-Time Translation system for Nidaros.

Its aim was to define how different components – such as modules for livestream, speech recognition, translation, and subtitle packaging modules – interact with one another. The model visually captures the logical structure of the system using Domain objects such as stream, audio segment, transcription, translation, subtitle, and user.

This card connects the research done in the architecture and design document by assuring that all technical modules share a common understanding of data flow and relationships.

How the Research was conducted

We extracted the entities and relationships directly from the requirements of the project using the domain-driven design approach, the Use Case diagram, and the Activity diagram.

We analyzed the workflow of the platform, from the moment audio is streamed from Wowza, through the noise reduction module, speech-to-text module and machine translation, subtitle packager, and finally to the client viewer.

The process involved:

- Review the Use Case and Activity diagrams from the design document to identify actions, actors, and data passed between modules.
- Mapping these interactions into entities and relationships by using UML notation.
- Collaboration with developers and architecture to align the model to the actual implementation structure.
- Validating the model by checking consistency, for example, that the subtitle entity links to the translation and stream entities appropriately according to the architecture diagram.

Result of the Research

The final domain model provides a clear and scalable structure for the Real-Time Translation system:

- A given stream can be associated with one or more audio segments.
- Every audio segment generates one transcription.
- Each transcription can have multiple translations (for English, Dutch etc.)
- A subtitle is a translation combined with a timestamp to synchronize it with the video.
- The user entity represents the viewer who gets the stream video with translated subtitles.

What We Learned

The model provided a mutual conceptual understanding by the developers and researchers, ensuring alignment between software design and project goals. It also helped refine the architecture diagram and improved data consistency across modules.

In conclusion, the domain model connected research and implementation by laying the groundwork for the system's data flow, making it clear who is responsible for what within modules and making it easier to add new translation languages or accessibility features in the future.

5.4. Stakeholder Analysis

Card description

As part of our research approach, we used a Stakeholder Analysis to understand who is involved in or would be affected, what their interests and expectations are, and how their influence could redirect the project. This helped us to be ensured that the needs of different stakeholder groups were considered early in the project.

Purpose of the Research

The main goal of this research method was to identify all relevant stakeholders so we can understand their roles, influence, and their level of interest in the project. By doing this, we would be informed about:

- whose needs should be prioritised,
- how we have to make sure we meet their needs,
- who has the influence to challenge the project,
- and what expectations must be aligned for the project to succeed.

How the Research Was Conducted

We followed a structured process to analyse the stakeholders:

1. **Identifying all the stakeholders;**

   We started by listing all individuals and groups connected to the project. We divide the stake holders between direct and indirect, as well as internal and external stakeholders. This allowed us to go beyond the obvious stakeholders and include groups that may not be directly involved but could affect the project later (e.g., GDPR, competitors, Nidaros employees).

2. **Categorising and Prioritising:**

   We used the Mendelow Matrix to classify stakeholders based on their level of influence and interest. This helped us determine how to engage with each group.
   For example, the CTO and the client and the developer team were placed in the "High Power, High Interest" category, meaning they require close collaboration and frequent updates and their needs must be considered as the most important.

3. **Analysing Stakeholder Needs and Expectations:**

   We mapped out how each stakeholder relates to the project, the kind of impact they have, and what they expect from the final product. This ensured that we kept user needs, technical feasibility, and compliance requirements in mind.

4. **Communication Plan:**

   Based on the matrix table, we developed a communication plan to maintain support from key stakeholders throughout the project. Each category received a communication strategy (e.g., weekly meetings for key figures, milestone updates for influencers) for example we keep track of the people who how low influence, because their level interest can change during the time.

Results of the Analysis

Through this Stakeholder analysis, we gained several important information:

- Different stakeholders have different priorities some focus on feasibility and technical quality, while others care about accessibility, user experience, or regulatory compliance.
- The end users have the most influence in defining project success, as the final product is being developed for them.
- Frequent communication with key stakeholders is essential to keep the project aligned with expectations and avoid major changes later.

5.5. Prototyping Card

In terms of prototyping, we have done an initial throw-away prototype, that we will simply call “prototype” and the final product of this period, a proof of concept that we will call “PoC”.

Prototype:

The scope of our prototype was to pinpoint the difficulties that we might encounter while working on the PoC and make better architecture decisions before developing the PoC.

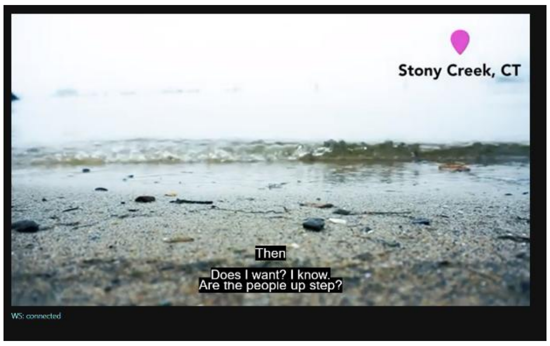


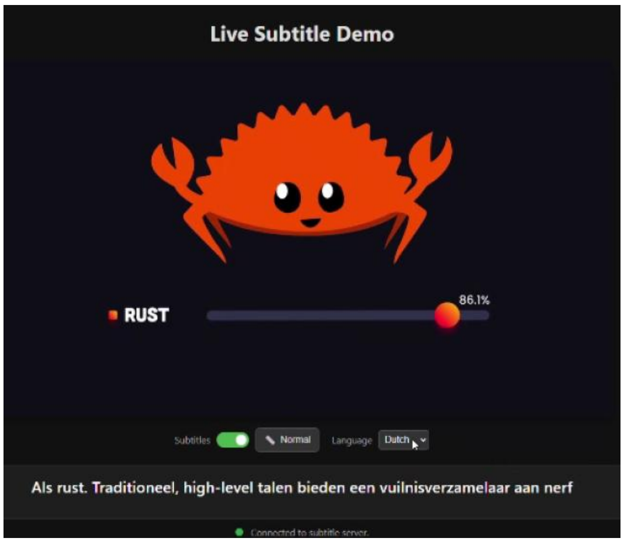
Figure 2: Prototype creating English captions

We have developed it in a programming language that is very unsuitable for our Use Case, Python, being very slow, and our client’s company is not even supporting it. The reason why we used it though is that it is very easy and fast to develop a simple application using it. A website player would play a video, whose sound was captured by our app. Our app would segment that audio, transcribe it using AI, and send the transcriptions back to the website to display them. We have learnt through it how difficult it can be to achieve good segmentation, that doesn’t miss any audio, the different types of connections for our Use Case (RTSP and HLS), the most important lesson was that HLS is the way to go, but it is a very complicated process.

PoC:

The scope of our PoC was to prove that this project is doable as a professional product, see any places of improvement for the minimum valuable product of the second period and become familiar with the technologies that we will use, Wowza, Docker, .net.

We have developed it using the clients supported programming language and framework (C#, .net) and made it scalable, by containerizing it. We have also used their software, Wowza for serving the livestream. A container would receive the audio from Wowza, create timestamps for each audio segment, create an AI based transcription, create AI based Machine Translation, and then send the captions, along with their timestamps to the client. The client receives the video/audio stream directly from Wowza, and the subtitles from our program. The subtitles are synchronized with the video and displayed.



Als rust. Traditioneel, high-level talen bieden een vuilnisverzamelaar aan nerf

Figure 3: PoC translating an English livestream into Dutch

From this PoC we have learnt that this can clearly become a professional product, and we can start working on a minimum valuable product in the second period. In terms of improvement, we will change the architecture of the system for more robustness and long-term support. Our app will take the whole stream by HLS, embed the subtitles into it, and send it to the website. This way, the website doesn't have to do any synchronization between the subtitles and the stream, as they are a single package, the dataflow is a single continuous flow, instead of diverging and converging, and we don't have to rely on RTSP, a protocol that is becoming less supported every day. As a bonus, a lot of our code can be reused for the minimum valuable product, the transcription module, the Wowza module, and the translation module.

5.6. Computer Simulation Card

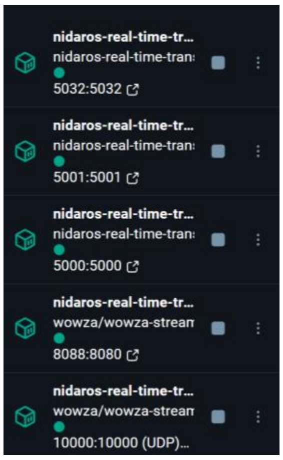
Purpose of the Research

The goal of this research was to test how our Real-Time Translation system behaves when all services run together inside Docker.

We wanted to check if the livestream from OBS through Wowza could reach the backend and send subtitles to the viewer. Running the setup this way helped us notice possible technical problems before the final version is deployed.

How the Research was conducted

We started all containers using Docker Compose to make a test setup similar to a real deployment. Wowza received the RTMP stream from OBS, which included both video and microphone input. Whisper Service transcribed the captured audio into text. The API Service (.NET) used FFmpeg inside the container to process small audio segments, send them to Whisper, and forward the transcribed text to connected clients using SignalR.



```
nidaros-real-time-tr...
nidaros-real-time-tran:
5032:5032

nidaros-real-time-tr...
nidaros-real-time-tran:
5001:5001

nidaros-real-time-tr...
nidaros-real-time-tran:
5000:5000

nidaros-real-time-tr...
wowza/wowza-stream
8088:8080

nidaros-real-time-tr...
wowza/wowza-stream
10000:10000 (UDP)...
```

Figure 4: Docker containers used in the simulation.

As shown in Figure 4, all required Docker containers were running simultaneously, allowing us to test the full pipeline in a realistic environment. During testing we streamed from OBS to Wowza and watched how the system reacted. Subtitles appeared correctly when we spoke English, but there was some delay between the sound and the text. The delay changed depending on the computer that was used. On stronger laptops (like gaming models) the subtitles appeared faster, while slower ones started to fall behind after a while. Sometimes Docker needed to be restarted or the environment settings adjusted to make the services connect properly again.

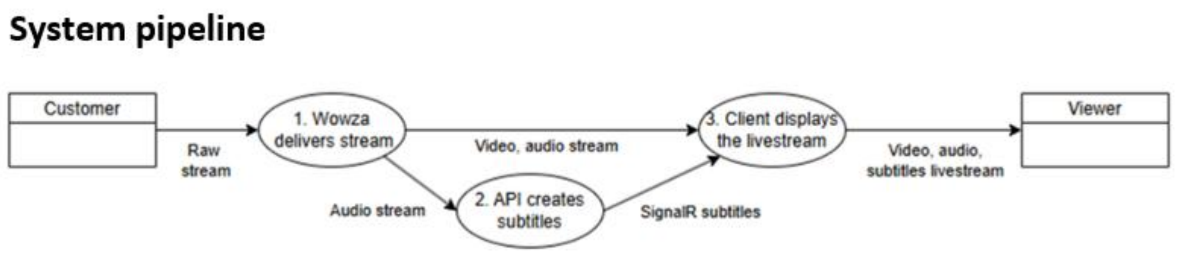


Figure 5: System pipeline during the simulation.

As shown in Figure 5, the system pipeline shows how the stream starts at OBS (which acts as the Customer in the diagram), moves through Wowza, then the transcription and translation steps, and finally reaches the viewer.

Results of the simulation

The test confirmed that the whole data flow from the microphone through Wowza, Whisper, and the backend, to the output interface works correctly. Subtitles appeared live, showing that the system design is working as intended. We also found two main issues: the growing delay between speech and subtitles, and performance differences caused by hardware speed. These results gave us a clear idea of what needs improvement, such as reducing delays and using system resources more efficiently.

What we learned

From this simulation, we learned how Docker Compose helps connect several containers and how small network settings can cause connection problems. We also noticed that the speed of the computer makes a big difference.

Faster CPUs handle the sound and create subtitles quicker, while slower ones start to build delay over time.

We found that Whisper works only for English input in this setup.

Doing this kind of test helped us understand how each part depends on the others and what still needs to be improved before release.

The simulation showed that our proof of concept already works with real video and microphone input.

There are still points to fix, like delay and multi-language support, but the system now runs as a full chain and gives us a clear view of what to focus on next to make the platform more accessible for all users.

5.7. System Test Card

Card Description

The system test card describes a process taken to ensure that a system works to meet the expected outcome during the realization phase of a project. The purpose of this process is to find problems within the system before it goes live. This is done by testing all subsystems not individually, but as a complete unit against the original requirements. The card also demonstrates that, this needs a specific "bug-killing mindset", which is different from a developer's mindset of just showing that the system is working.

The processes include making a detailed test plan with specific test cases (for both "happy flow" and error handling) and then comparing the actual results to the expected outcomes.

How Research was conducted

These are the steps I took during my research:

1. I first went through the main components of the card: the "Why" (preventing production errors), the "How" (creating test plans), and the "Ingredients" (test scenarios, bug-killing mindset).

2. I used these concepts directly to our most recent Proof of Concept (POC), which consists of multiples components which include:

- a user interface that displays a stream and shows subtitles of the stream.
- a tool that captures a live stream audio in chunks and sends it over to a STT model. (FFmpeg)
- a locally hosted speech to text model from openAI that takes an incoming audio stream and transcribes it to text.
- and the logic that stitches all these services together.

Identifying test scenarios

Based on this, I researched and brainstormed test case scenarios, and I focused on the "happy flow" and, more importantly, the "specific cases" for error handling, as the card suggests. This involved thinking about what could go wrong with a live translation system specifically.

Results of the Research

Happy flow:

- Test case 1: The stream audio plays simple words ("Hello, how are you?").
- Expected outcome: The system accurately transcribes the sentence within 8 seconds. ("Hello, how are you?").

Error handling:

- Test case 2: The stream has a lot of background noise.
- Expected outcome: The system will either filter the noise and give a correct translation, or it displays a user-friendly error message like, "Inaudible sound".

Specific Case:

- Test case 3: User selects the same output language as the stream language.

- Expected outcome: The output displays the language (passing the text through without translation).

Error handling:

- Test case 4: The translation API fails due to, for example, a network error or a translation system crash.
- Expected outcome: The system does not crash. A user-friendly error message ("Translation service unavailable") is displayed.

Specific Case:

- Test Case 5: Speech is in German language ("Guten Tag") while the API language has been set to translate from Dutch to any other language.
- Expected outcome: The system recognizes the mismatch and either provides a nonsensical translation (documenting this "bad" result) or flags an error.

Specific Case:

- Test Case 6: Speech is very quick without pauses in the stream.
- Expected Outcome: The system will capture and translate the full sentence, and testing the limits of the speech-to-text service will be recorded.

6. Summary

The project successfully answers the question "How can we implement a real-time transcription and translation solution that enables all viewers, including non-native speakers and hearing-impaired users, to comprehend the live stream content?" by using a research pattern and a set of cards to select the technologies and define the requirements and validate the system performance.

- **Selecting technologies (Literature Study):** Choosing high-performance, open-source tools: OpenAI Whisper for better accuracy of multilingual speech to text and Argos Translate for self-hosted translation, fulfilling the requirement for a free, scalable system.
- Defining Requirements (Defining Requirements and Stakeholder Analysis): we mapped all the people who are interested and can influence the project and take their needs and expectations into consideration. For example, Subtitles will pop up within 8 seconds after speech to make users feel close to real time conversation which is directly affect to the end users.
- **Validating system performance (Prototyping, Simulation, & System Test):** The Domain Modelling card defined the data flow in our system, and the Proof of Concept (PoC), tested via Computer Simulation and System Testing, confirmed that the entire integrated pipeline from Wowza audio capture to subtitle display works smoothly. This ensures the implementation method is functional, enabling non-native speakers and impaired hearing users to finally understand the live stream content.

Overall, the research shows that using Whisper, Argos Translate and WebVTT subtitles together with Wowza and JW Player is an effective way to implement real-time transcription and translation.

7. Resources

Argos Open Technologies. (n.d.). *Argos Translate* [GitHub repository]. GitHub. https://github.com/argosopentech/argos-translate

ChocolateMagnate. (n.d.). *Speech-to-text benchmarks* [GitHub repository]. GitHub. https://github.com/ChocolateMagnate/speech-to-text-benchmarks

OpenAI. (2023). *Whisper: Speech recognition model* [Computer software]. https://github.com/openai/whisper