

# DIPLOMARBEIT

Gesamtprojekt

## Implementierung und Transkription von Web Videocalls

**Entwicklung einer skalierbaren Backend-Architektur für Videomeeting-Anwendungen: Untersuchung der Leistungsfähigkeit und Effizienz von AWS Lambda und Aurora Serverless in der serverlosen Infrastruktur. (DB)**

Bastian Seidl 5CHIF Betreuer: Andreas Zöttl

**Optimierung der Benutzeroberfläche einer Videomeeting-Anwendung: Eine Untersuchung der Effizienz und Nutzerfreundlichkeit von Next.js, React und Tailwind CSS für skalierbare Webanwendungen.**

Philip Schrenk 5CHIF Betreuer: Andreas Zöttl

**Automatisierte Bereitstellung und Skalierung serverloser Anwendungen: Best Practices für CI/CD-Pipelines in AWS unter Verwendung von GitLab CI für Videomeeting-Lösungen.**

Louis Muhr 5CHIF Betreuer: Andreas Zöttl

**KI-gestützte Transkription und Analyse von Videomeetings: Untersuchung der Anwendung von OpenAI Whisper und GPT zur Verbesserung von Meeting-Aufzeichnungen, Zusammenfassungen und Handlungsempfehlungen.**

Maximilian Schwarz 5CHIF Betreuer: Andreas Zöttl

ausgeführt im Schuljahr 2024/25

---

Abgabevermerk:

Datum: 01.04.2025

übernommen von:

 SPENGERGASSE	<b>HTBLVA Wien V, Spengergasse</b> Aufbaulehrgang für Informatik	<b>Reife- und Diplomprüfung</b>
---	---	-------------------------------------

## EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen Hilfsmittel als die angegebenen benutzt habe. Die Stellen, die anderen Werken (gilt ebenso für Werke aus elektronischen Datenbanken oder aus dem Internet) wörtlich oder sinngemäß entnommen sind, habe ich unter Angabe der Quelle und Einhaltung der Regeln wissenschaftlichen Zitierens kenntlich gemacht. Diese Versicherung umfasst auch in der Arbeit verwendete bildliche Darstellungen, Tabellen, Skizzen und Zeichnungen. Für die Erstellung der Arbeit habe ich auch folgende Hilfsmittel generativer KI-Tools **xAi Grok 3, DeepL Write**

zu folgendem Zweck verwendet:

### Auslegung und Formulierung von Sätzen

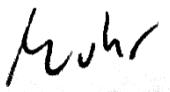
Die verwendeten Hilfsmittel wurden vollständig und wahrheitsgetreu inkl. Produktversion und Prompt ausgewiesen.

Wien, am 01.04.2025

Verfasser / Verfasserinnen:

Bastian Seidl

Philip Schrenk

  
Louis Muhr

Maximilian Schwarz

 <b>SPENGERGASSE</b>	<b>HTBLVA Wien V, Spengergasse</b> Aufbaulehrgang für Informatik	<b>Reife- und</b> <b>Diplomprüfung</b>
--	---	---

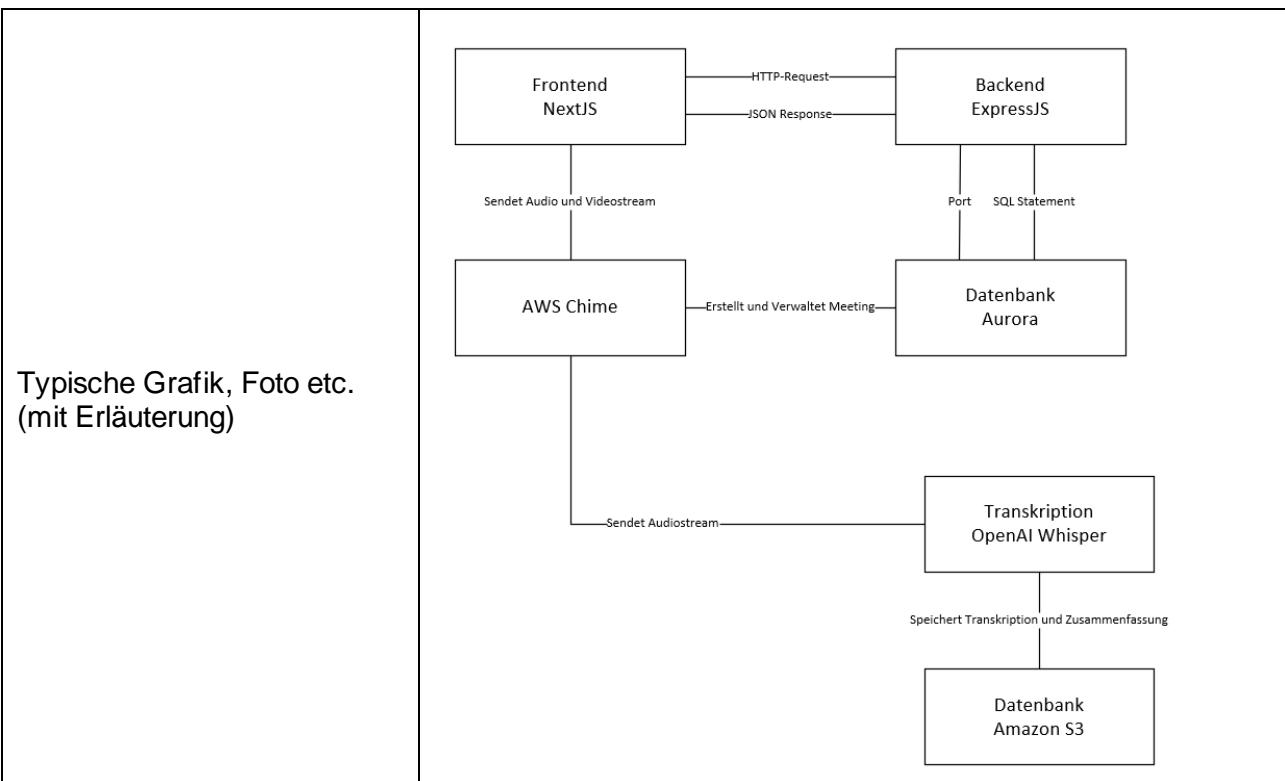
## **DIPLOMARBEIT DOKUMENTATION**

Namen der Verfasser/innen	Bastian Seidl Philip Schrenk Louis Muhr Maximilian Schwarz
Jahrgang Schuljahr	5CHIF 2024/25
Thema der Diplomarbeit	Implementierung und Transkription von Web Videocalls
Kooperationspartner	Naos GmbH

Aufgabenstellung	Entwicklung einer gebrandeten 1:1-Videomeeting-Anwendung mit AWS Chime, Next.js/React und Tailwind. Die Anwendung bietet feste Meeting-URLs für Hosts, Warteräume für Gäste, Authentifizierung über SuperTokens, Videokonferenzfunktionen und automatische Aufzeichnung mit Transkription und Zusammenfassung durch OpenAI Whisper und GPT. Das Backend basiert auf AWS-Lambda und Aurora Serverless.
------------------	---

Realisierung	Das Projekt wurde wie folgt realisiert: Zum einen haben wir eine Webapplikation mit Next.js/React und Tailwind im Frontend erstellt. Und gleichzeitig ein Backend basierend auf AWS-Lambda und Aurora Serverless.
--------------	---

Ergebnisse	Vollständige Entwicklung einer Videomeeting-Plattform, welche dem Kunden ermöglicht, seine Meetings zu planen, aufzuzeichnen und gemütlich durchzuführen. Zusätzlich wurden einige Features eingebaut wie z.B. die Authentifizierung mithilfe von SuperTokens, die Verteilung von Userrechten innerhalb eines Calls und feste Meeting-URLs.
------------	---



Teilnahme an Wettbewerben, Auszeichnungen	Keine
---	-------

Möglichkeiten der Einsichtnahme in die Arbeit	
---	--

Approbation (Datum / Unterschrift)	Prüfer/Prüferin	Direktor/Direktorin Abteilungsvorstand/Abteilungsvorständin
---------------------------------------	-----------------	--

 <b>SPENGERGASSE</b>	<b>HTBLVA Wien V, Spengergasse</b> Aufbaulehrgang für Informatik	<b>Reife- und</b> <b>Diplomprüfung</b>
--	---	---

## DIPLOMA THESIS

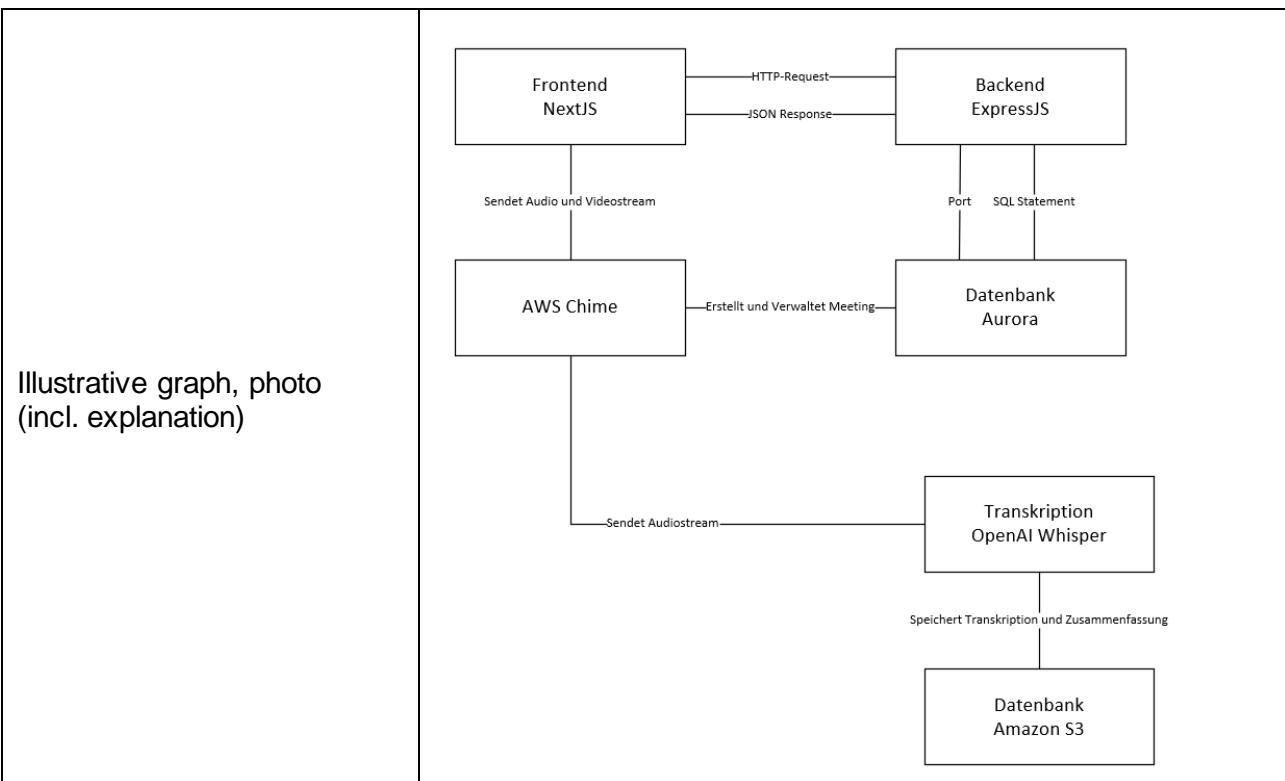
### Documentation

Author(s)	Bastian Seidl Philip Schrenk Louis Muhr Maximilian Schwarz
Form Academic year	5CHIF 2024/25
Topic	Implementation and Transcription of a Web-Videocall
Co-operation partners	Naos GmbH

Assignment of tasks	Development of a branded 1:1 video meeting application using AWS Chime, Next.js/React, and Tailwind. The application provides fixed meeting URLs for hosts, waiting rooms for guests, authentication via SuperTokens, video conferencing features, and automatic recording with transcription and summarization using OpenAI Whisper and GPT. The backend is based on AWS Lambda and Aurora Serverless.
---------------------	---

Realisation	The project was implemented as follows: On one hand, we created a web application with Next.js/React and Tailwind for the frontend. At the same time, we developed a backend based on AWS Lambda and Aurora Serverless.
-------------	---

Results	Complete development of a video meeting platform that enables the customer to schedule, record, and comfortably conduct their meetings. Additionally, several features were integrated, such as authentication using SuperTokens, the assignment of user permissions within a call, and fixed meeting URLs.
---------	---



Participation in competitions	
Awards	None

Accessibility of diploma thesis	
---------------------------------	--

Approval (date / signature)	Examiner	Head of College / Department
--------------------------------	----------	------------------------------



# Inhaltsverzeichnis

Themenstellung von Bastian Seidl .....	2
1. Was ist "serverless"? .....	2
2. Cloudanbieter: eine Marktanalyse .....	8
3. Benutzte Features in der Videomeeting Applikation .....	10
4. Kommunikation mit der Cloud Infrastruktur .....	13
5. Kostenschätzung für den laufenden Betrieb .....	16
6. Erweiterung der Applikation .....	17
Fazit .....	23
Literaturverzeichnis .....	24
Themenstellung von Philip Schrenk .....	25
1. Einleitung und Begriffsdefinition .....	25
2. Next.js: Moderne Lösung zu Fullstack-Projekten .....	26
3. TailwindCSS: Ersatz zu herkömmlichen CSS .....	29
4. AWS Chime und Next.js als Lösung für Videomeet-Anwendungen .....	31
5. Gestaltung und Optimierung der Benutzeroberfläche .....	34
6. Erweiterung der Applikation .....	36
Fazit .....	42
Literaturverzeichnis .....	42
Themenstellung von Louis Muhr .....	44
1. Einleitung .....	44
2. Grundlagen .....	46
3. Best Practices für CI/CD-Pipelines in AWS mit GitLab CI .....	50
4. Anwendungsfall: Videomeeting-Lösungen .....	56
Fazit .....	61
Literaturverzeichnis .....	61
Themenstellung von Maximilian Schwarz .....	63
1. Einleitung .....	63
2. Grundlagen .....	64
3. Stand der Technik .....	67
4. Implementierung .....	75
Literaturverzeichnis .....	82
Anhang .....	84



# Themenstellung von Bastian Seidl

Entwicklung einer skalierbaren Backend-Architektur für Videomeeting-Anwendungen: Untersuchung der Leistungsfähigkeit und Effizienz von AWS Lambda und Aurora Serverless in der serverlosen Infrastruktur

---

## 1. Was ist "serverless"?

### 1.1 Begriffserklärung

Der Begriff „Serverless“ hat in den letzten Jahren enorm an Bedeutung gewonnen, wird jedoch oft missverstanden. Entgegen der wörtlichen Interpretation bedeutet „serverless“ nicht, dass keine Server im Spiel sind – vielmehr übernimmt der Cloud-Anbieter die gesamte Verwaltung, Wartung und Skalierung der zugrunde liegenden Infrastruktur. Diese Abstraktion ermöglicht es Entwicklern, sich vollständig auf die Entwicklung von Anwendungen zu konzentrieren, ohne sich mit Serverkonfigurationen, Betriebssystem-Updates oder Hardwareproblemen auseinanderzusetzen zu müssen. Der Ansatz wurde maßgeblich durch die Einführung von AWS Lambda im Jahr 2014 populär, einem Dienst, der als Pionier der serverlosen Architektur gilt. Lambda erlaubt es Entwicklern, kleine, ereignisgesteuerte Funktionen – sogenannte Lambda-Funktionen – zu schreiben, die automatisch auf Ereignisse wie HTTP-Anfragen, Datenbankänderungen, Datei-Uploads oder zeitgesteuerte Trigger reagieren. [9]

Für Videokonferenzanwendungen ist dieses Konzept besonders wertvoll, da die Nutzerzahlen stark schwanken können: Ein kleines Team-Meeting mit fünf Teilnehmern erfordert deutlich weniger Ressourcen als eine internationale Konferenz mit mehreren Hundert Teilnehmern oder gar ein globales Webinar mit Tausenden von Zuschauern. Serverless-Architekturen bieten hier eine entscheidende Stärke: Sie skalieren automatisch und dynamisch. Bei einem plötzlichen Anstieg der Nachfrage – etwa wenn ein großes Event beginnt – werden zusätzliche Ressourcen in Echtzeit bereitgestellt. Sinkt die Auslastung, etwa nach Ende eines Meetings, werden diese Ressourcen ebenso schnell wieder freigegeben, ohne dass manuelle Eingriffe erforderlich sind. Im Gegensatz dazu erfordern traditionelle Architekturen die Bereitstellung fester Serverkapazitäten, die oft überdimensioniert oder ungenutzt bleiben, was sowohl Kosten als auch Verwaltungsaufwand erhöht. Besonders für Start-ups und kleinere Unternehmen, die keine großen IT-Abteilungen unterhalten, bietet Serverless eine Möglichkeit, sich auf ihre Kerngeschäfte – wie die Entwicklung innovativer Anwendungen – zu konzentrieren, anstatt Zeit und Budget in die Infrastruktur zu investieren. [9]

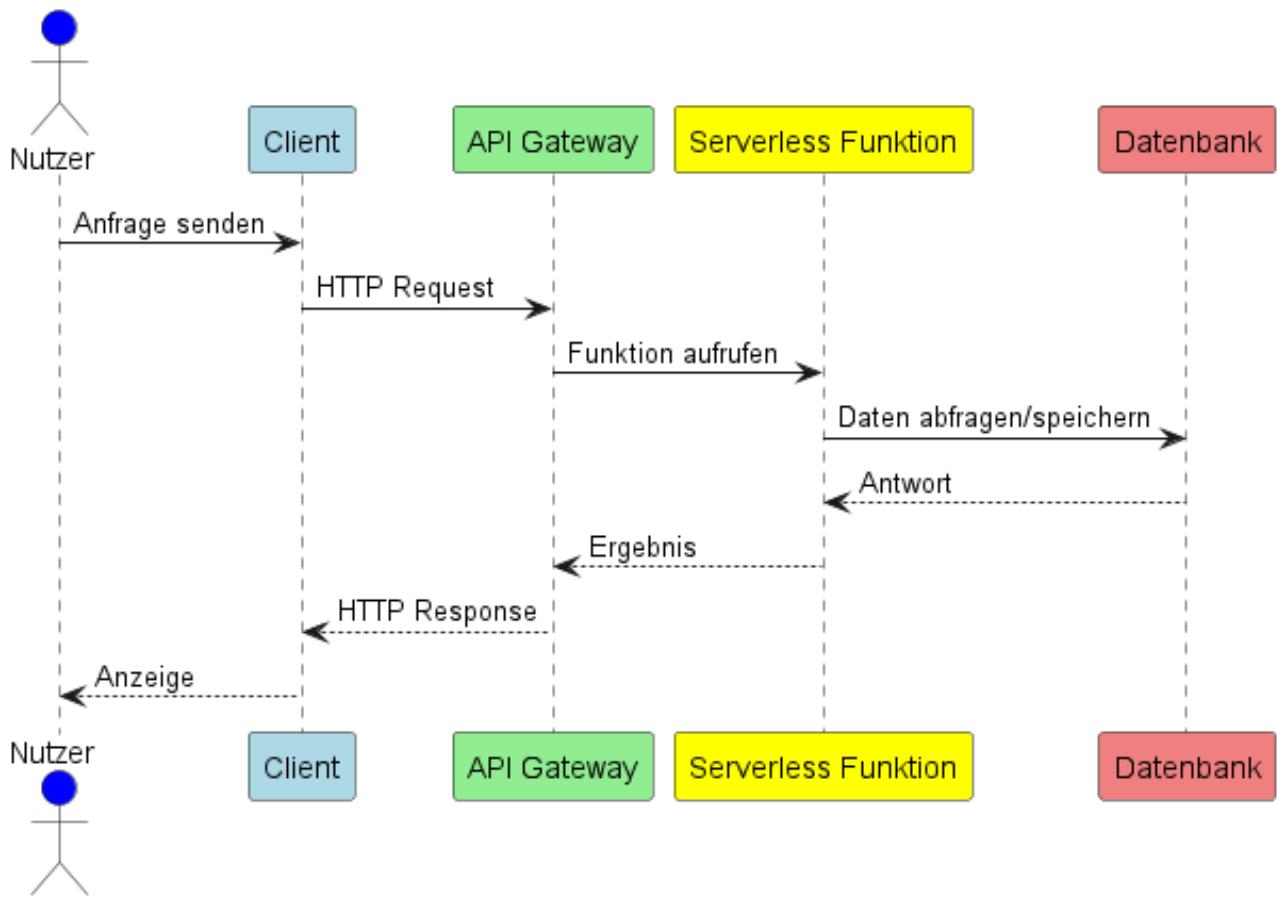


Abbildung 1. Serverless Architektur

## 1.2 Technische Grundlagen und Funktionsweise

Die technische Basis einer serverlosen Architektur ist ein ereignisgesteuertes Programmiermodell, das sich grundlegend von klassischen monolithischen Ansätzen unterscheidet. Ein Ereignis – etwa der Start eines Videomeetings, das Senden einer Chatnachricht oder das Hinzufügen eines neuen Teilnehmers – löst eine Funktion aus, die in einer isolierten, temporären Umgebung ausgeführt wird. Bei AWS Lambda wird diese Umgebung als „Container“ bezeichnet, der innerhalb von Millisekunden gestartet wird, den bereitgestellten Code ausführt und anschließend wieder beendet wird. Dieser Ablauf erfolgt vollständig automatisiert: Entwickler müssen weder die Container selbst verwalten noch deren Lebenszyklus überwachen. Ein zentraler Vorteil dieses Modells ist der „Pay-per-Use“-Ansatz: Kosten entstehen nur für die tatsächliche Ausführungszeit der Funktionen, gemessen in Millisekunden, und nicht für Leerlaufzeiten wie bei ständig laufenden Servern. Für eine Videokonferenzanwendung könnte dies bedeuten, dass die Initialisierung eines Meetings mit zehn Teilnehmern nur wenige Cent kostet, während ein großer Event mit Tausenden Teilnehmern entsprechend mehr Ressourcen beansprucht – jedoch stets proportional zur Nutzung. [9]

Ein weiteres Schlüsselement serverloser Architekturen ist die Datenbanktechnologie. Hier kommt Aurora Serverless ins Spiel, eine serverlose Variante der relationalen Datenbank Amazon Aurora, die ihre Kapazität automatisch an die aktuelle Last anpasst. Im Gegensatz zu traditionellen Datenbanken, bei denen Entwickler die Anzahl der Instanzen oder die Servergröße im Voraus



festlegen müssen, skaliert Aurora Serverless dynamisch zwischen minimalen und maximalen Kapazitätsgrenzen. Für eine Videokonferenzanwendung könnte dies folgendes Szenario bedeuten: Bei einem kleinen Meeting mit wenigen Teilnehmern bleibt die Datenbank auf einem niedrigen Kapazitätsniveau, um Metadaten wie Teilnehmerlisten, Zeitstempel oder Chatverläufe zu speichern. Tritt jedoch ein plötzlicher Anstieg der Nutzerzahlen auf – etwa durch ein virales Event oder eine große Firmenkonferenz – erhöht Aurora Serverless die Kapazität nahtlos, ohne dass Entwickler eingreifen müssen. Diese Flexibilität macht Aurora Serverless zu einem idealen Partner für AWS Lambda in serverlosen Architekturen und unterscheidet sie deutlich von statischen Datenbanklösungen, die bei Lastspitzen oft an ihre Grenzen stoßen. [10]

### 1.3 Vorteile und Herausforderungen

#### *Liste der wichtigsten Vorteile von Serverless*

- **Keine Serveradministration:**

Entwickler können sich vollständig auf die Geschäftslogik ihrer Anwendung konzentrieren, während der Cloud-Anbieter – in diesem Fall AWS – die gesamte Hardware, Betriebssysteme, Sicherheitsupdates und Serverwartung übernimmt. Dies reduziert den administrativen Overhead erheblich und beschleunigt den Entwicklungsprozess.

- **Automatische Skalierung:**

Die Infrastruktur passt sich in Echtzeit an die aktuelle Nachfrage an. Bei einem unerwarteten Anstieg der Teilnehmerzahlen – etwa durch ein spontanes Großevent – werden zusätzliche Ressourcen ohne Verzögerung bereitgestellt, während bei geringer Nutzung automatisch heruntergefahren wird, was Ressourcenverschwendungen verhindert.

- **Kosteneffizienz:**

Im Vergleich zu dedizierten Servern, die rund um die Uhr laufen und auch im Leerlauf Kosten verursachen, fallen bei Serverless nur Kosten für die tatsächliche Nutzung an. Dies ist besonders vorteilhaft für Anwendungen mit stark schwankender Auslastung, wie Videokonferenzlösungen.

#### *Herausforderungen von Serverless*

- **Kaltstart-Latenz:**

Beim ersten Aufruf einer Lambda-Funktion oder nach längerer Inaktivität muss ein neuer Container initialisiert werden, was zu Verzögerungen von einigen Millisekunden bis hin zu Sekunden führen kann. Für latenzsensitive Anwendungen wie Echtzeit-Videostreaming kann dies ein kritischer Nachteil sein, der durch Techniken wie „Provisioned Concurrency“ abgemildert werden muss.

- **Komplexe Debugging-Prozesse:**

Da serverlose Architekturen stark verteilt sind und viele kleine Funktionen miteinander interagieren, kann das Debugging schwieriger sein als bei monolithischen Systemen. Logs sind über verschiedene Dienste verteilt, und Fehlerursachen sind nicht immer sofort offensichtlich.

- **Vendor Lock-in:**

Die enge Bindung an einen spezifischen Anbieter wie AWS bedeutet, dass eine spätere Migration zu einem anderen Anbieter – etwa aufgrund von Kosten oder neuen Anforderungen –



SPENGERGASSE

**HTBLVA Wien V, Spengergasse**  
**Höhere Lehranstalt für Informatik**

**Reife- und  
Diplomprüfung**

technisch und finanziell aufwendig sein kann. Funktionen, Datenbanken und Integrationen sind oft auf die proprietären Dienste des Anbieters zugeschnitten.



## 1.4 Einsatz in der Praxis

In dieser Diplomarbeit wird der serverlose Ansatz als Grundlage für eine skalierbare Videomeeting-Anwendung untersucht, die den Anforderungen moderner Kommunikation gerecht wird. Ein praxisnahes Szenario könnte wie folgt aussehen: Ein Nutzer startet ein neues Meeting über eine Webanwendung. Dieser Vorgang löst eine Lambda-Funktion aus, die den Videostream initialisiert, die Teilnehmerdaten in Aurora Serverless speichert und die Verbindung zwischen den Teilnehmern über ein Echtzeitprotokoll wie WebRTC aufbaut. Während des Meetings könnten weitere Lambda-Funktionen aktiviert werden, etwa um Teilnehmer hinzuzufügen. Dieser Ansatz reduziert nicht nur die Entwicklungszeit, sondern optimiert auch die Ressourcennutzung erheblich: Ressourcen werden nur dann bereitgestellt, wenn sie tatsächlich benötigt werden, und nicht pauschal für alle potenziellen Nutzer vorgehalten. Dies macht Serverless besonders attraktiv für Anwendungen, die von Natur aus dynamisch und unvorhersehbar sind – ein Merkmal, das Videokonferenzlösungen perfekt beschreibt.



## 1.5 Beispiel einer Lambda-Funktion

Das folgende Beispiel zeigt einen Teil der `createMeeting.js`. Dieser Ausschnitt beschreibt eine Lambda-Funktion, die ein neues Meeting in einer Aurora Serverless-Datenbank erstellt. Die Funktion wird aufgerufen, wenn ein Benutzer ein Meeting über eine API-Anfrage startet, nimmt die Meeting-Daten entgegen, validiert sie und speichert sie in einer PostgreSQL-Datenbank. Der Rückgabewert enthält die eindeutige ID des erstellten Meetings, die an den Frontend-Client zurückgegeben wird.

*createMeeting.js - Beispiel einer Lambda-Funktion die ein Meeting erstellt*

```
// Create Meeting
const { Client } = require("pg");

const client = new Client({
  user: 'postgres',
  host: 'database-1.cwq5q1jz8m4c.us-east-1.rds.amazonaws.com',
  database: 'ioe-meeting',
  password: 'password',
  port: 5432,
});

exports.handler = async (event) => {
  try {
    const body = JSON.parse(event.body);
    if (!body.title || !body.start_time || !body.end_time) return { statusCode: 400, body: "Missing fields" };

    await client.connect();
    const { rows } = await client.query("INSERT INTO meetings (title, start_time, end_time) VALUES ($1, $2, $3) RETURNING id", [body.title, body.start_time, body.end_time]);
    await client.end();

    return { statusCode: 201, body: JSON.stringify({ meeting_id: rows[0].id }) };
  } catch (error) {
    return { statusCode: 500, body: error.message };
  }
};
```



## 2. Cloudanbieter: eine Marktanalyse

### 2.1 Überblick über den Markt

Die Wahl des Cloud-Anbieters hat großen Einfluss auf die Architektur, die Kosten und die Leistung einer serverlosen Videokonferenzanwendung. Laut Synergy Research Group (Stand Q3 2024) hat Amazon Web Services (AWS) mit rund 32% den größten Marktanteil, gefolgt von Microsoft Azure (22%) und Google Cloud Platform (GCP, 11%). Diese Analyse vergleicht die drei führenden Anbieter hinsichtlich ihrer serverlosen Angebote. [1]

Cloud-Dienste sind für moderne Videokonferenzanwendungen unverzichtbar, da sie die Bewältigung dynamischer Lasten ermöglichen – ein entscheidender Unterschied zu traditionellen On-Premise-Lösungen, die auf statische Kapazitäten ausgelegt sind. Während ein lokaler Server bei einem Anstieg von 10 auf 10.000 Nutzern innerhalb weniger Minuten an seine Grenzen stoßen würde, skalieren Cloud-basierte, serverlose Lösungen automatisch und ohne manuellen Eingriff. AWS führte diesen Trend 2014 mit Lambda ein, gefolgt von Microsoft mit Azure Functions und Google mit Cloud Functions. Für eine Videokonferenzanwendung bedeutet dies, dass selbst unvorhersehbare Szenarien – wie ein globales Webinar oder eine spontane Unternehmenskonferenz – effizient und ohne Qualitätsverlust abgewickelt werden können. [2] [3] [4]

### 2.2 Amazon Web Services (AWS)

AWS ist seit seiner Markteinführung im Jahr 2006 der unangefochtene Marktführer und bietet ein umfangreiches Portfolio an Diensten, das speziell für serverlose Architekturen optimiert ist. Für Videokonferenzanwendungen stellt AWS eine Vielzahl an Tools bereit: Das AWS Chime SDK ermöglicht Video- und Audiostreaming in Echtzeit mit niedriger Latenz, Lambda übernimmt die serverlose Verarbeitung von Ereignissen, Amazon S3 bietet kostengünstigen Speicher für Aufzeichnungen und Aurora Serverless sorgt für eine skalierbare Datenbanklösung. Mit über 100 Availability Zones weltweit gewährleistet AWS eine minimale Latenz – ein entscheidender Faktor für die Qualität von Videokommunikation, insbesondere bei internationalen Meetings. Dienste wie API Gateway und Cognito erleichtern zudem die Entwicklung sicherer Anwendungen mit Benutzerauthentifizierung und API-Management.

Ein praktisches Beispiel: Ein Meeting-Ereignisse wie das Hinzufügen eines Teilnehmers zu verwalten, während Chime den Stream bereitstellt und S3 die Aufzeichnung speichert. Ein Nachteil von AWS ist jedoch die Komplexität: Die Vielzahl an Diensten und Konfigurationsmöglichkeiten ist überwältigend. Zudem können die Kosten bei unzureichender Optimierung schnell steigen, etwa durch übermäßige Lambda-Aufrufe oder ungenutzte Ressourcen. Dennoch bleibt AWS aufgrund seiner Reife, Dokumentation und Flexibilität die erste Wahl für viele Entwickler. [2] [5]

## 2.3 Microsoft Azure

Microsoft Azure hat sich durch die Integration mit Tools wie Teams etabliert. Azure Functions ist eine serverlose Alternative zu Lambda, Azure SQL Database bietet skalierbaren Speicher und Azure Communication Services unterstützt Video- und Audiofunktionen. Die Hybrid-Cloud-Fähigkeit von Azure ist ein Vorteil für Unternehmen mit On-Premise-Systemen. Ein Beispiel: Ein Unternehmen mit 10.000 Mitarbeitern könnte Azure nutzen, um Videomeetings in Teams zu integrieren, mit Blob Storage für die Aufzeichnungen. Das Wachstum von Azure von über 70 % bis 2024 zeigt seine Stärke, auch wenn die Integration mit Nicht-Microsoft-Tools begrenzt ist. [1] [6]

## 2.4 Google Cloud Platform (GCP)

GCP liegt mit 11% Marktanteil hinter AWS und Azure, punktet aber mit KI und Netzwerkleistung. Google Cloud Functions ist die serverlose Lösung, Cloud Spanner bietet globale Datenbankskalierung und WebRTC unterstützt Videomeetings. Die Netzwerkinfrastruktur von Google sorgt für niedrige Latenzzeiten. Ein Start-up-Unternehmen könnte GCP für ein KI-gestütztes Meeting-Tool mit automatischer Transkription verwenden. GCP ist kostengünstiger für einfache Workloads, aber das kleinere Portfolio und die geringere Community-Unterstützung sind Nachteile. [1] [7]

## 2.5 Vergleich und Entscheidung für AWS

Table 1. Vergleichstabelle der Anbieter

Anbieter	Serverless Compute	Serverless DB	Stärken	Schwächen
AWS	Lambda	Aurora Serverless	Flexibilität, Integration	Komplexität
Azure	Azure Functions	Azure SQL	Microsoft-Ökosystem	Weniger Drittanbieter
GCP	Cloud Functions	Cloud Spanner	Einfachheit	Kosten, Community

AWS wurde für diese Arbeit aus mehreren Gründen ausgewählt: Es bietet die beste serverlose Plattform, eine umfassende Dokumentation und spezifische Tools wie das Chime SDK, die direkt auf Videokonferenzanwendungen zugeschnitten sind. Zudem ermöglicht die Kombination aus Lambda und Aurora Serverless eine nahtlose Skalierung, die mit den dynamischen Anforderungen der Anwendung harmoniert. Während Azure und GCP ihre eigenen Stärken haben, bietet AWS die beste Balance zwischen Funktionalität, Erfahrung und spezialisierten Diensten für dieses Projekt.

### 3. Benutzte Features in der Videomeeting Applikation

#### 3.1 Überblick über die verwendeten AWS-Features

Die Videomeeting-Anwendung nutzt eine Kombination aus AWS Lambda, Aurora Serverless, AWS Chime SDK und Amazon S3, um eine skalierbare, effiziente und benutzerfreundliche Plattform zu schaffen. Diese Dienste decken die Kernanforderungen ab: Echtzeitverarbeitung von Ereignissen, persistente Speicherung von Metadaten, Videostreaming in hoher Qualität und sichere Speicherung von Aufzeichnungen und Transkriptionen. Jeder Dienst wird im Detail untersucht, mit technischen Implementierungen, praktischen Anwendungsszenarien und Optimierungsstrategien, um Leistung und Kosteneffizienz zu maximieren. Die serverlose Architektur ermöglicht eine dynamische Anpassung der Ressourcen an die Anzahl der Nutzer - von kleinen Teambesprechungen mit fünf Teilnehmern bis hin zu großen Konferenzen mit Hunderten von Teilnehmern - ohne dass eine manuelle Bereitstellung erforderlich ist. [8]

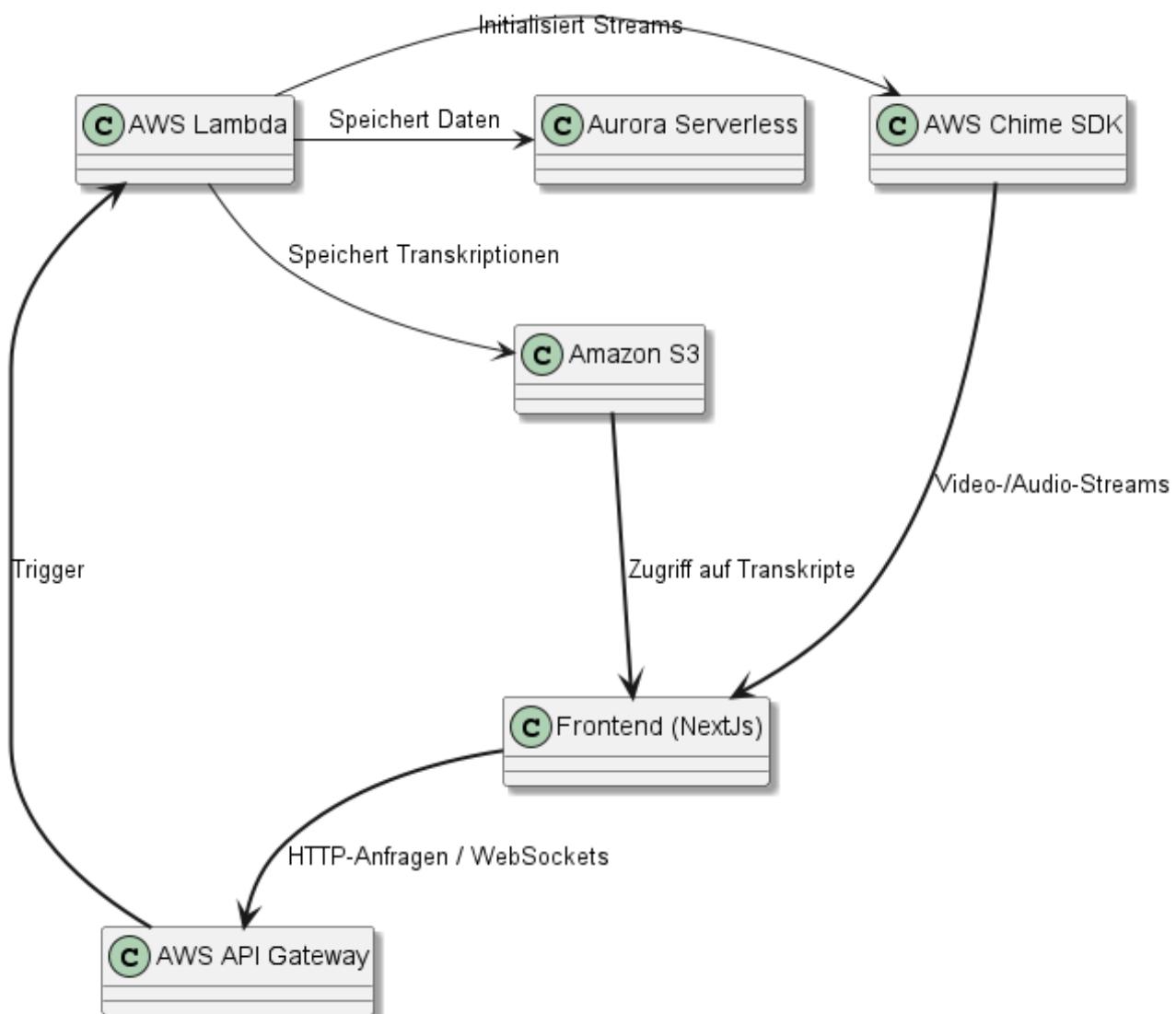


Abbildung 2. Benutze AWS-Dienste

## 3.2 AWS Lambda

AWS Lambda bildet das Rückgrat der serverlosen Architektur und übernimmt die Ereignisverarbeitung in Echtzeit. Es dient als zentrale Steuerungseinheit für die gesamte Anwendungslogik, z.B. für den Start eines Meetings, die Verwaltung von Teilnehmeraktionen (wie das Ein- und Ausschalten von Mikrofon oder Kamera) oder die Koordination von Backend-Prozessen. Lambda-Funktionen werden durch verschiedene Ereignisquellen ausgelöst, wie z.B. HTTP-Anfragen über das API-Gateway, WebSocket-Nachrichten oder Änderungen in der Datenbank. Die automatische Skalierung von Lambda ist ein entscheidender Vorteil: Bei zehn parallelen Meetings werden zehn Instanzen gestartet, bei 1.000 Meetings skaliert der Dienst nahtlos auf die entsprechende Anzahl, ohne dass Entwickler manuell Ressourcen bereitstellen müssen. [9]

## 3.3 Aurora Serverless

Aurora Serverless ist die bevorzugte Datenbanklösung für die Anwendung und bietet eine flexible, serverlose relationale Datenbank, die sich automatisch an die Last anpasst. Sie skaliert dynamisch zwischen 1 und 256 Aurora Capacity Units (ACUs), wobei jede ACU etwa 2 GB RAM und entsprechende CPU-Leistung bereitstellt. Bei einem Meeting mit 100 Teilnehmern speichert Aurora in Echtzeit Metadaten wie Teilnehmerlisten, Startzeiten und Meeting-IDs und kann bei Bedarf nahtlos auf 1.000 Teilnehmer hochskalieren, ohne dass Server manuell provisioniert werden müssen. Diese Flexibilität reduziert den administrativen Aufwand erheblich und sorgt für eine hohe Verfügbarkeit. [10]

Ein Beispielschema könnte wie folgt aussehen:

Table 2. Meeting-Tabelle in Aurora Serverless

Spalte	Typ	Beschreibung
meeting_id	UUID	Eindeutige ID des Meetings
aws_meeting_id	VARCHAR	ID des Meetings in AWS Chime
title	VARCHAR	Titel des Meetings
start_time	TIMESTAMP	Startzeit des Meetings
end_time	TIMESTAMP	Endzeit des Meetings

Table 3. User-Tabelle in Aurora Serverless

Spalte	Typ	Beschreibung
user_id	UUID	Eindeutige ID des Benutzers
aws_user_id	VARCHAR	ID des Benutzers in AWS Chime
name	VARCHAR	Name des Benutzers
meeting_id	UUID	ID des Meetings, an dem der Benutzer teilnimmt

Spalte	Typ	Beschreibung
isHost	bool	Rolle des Benutzers (Host oder Teilnehmer)

Aurora Serverless unterstützt zudem Funktionen wie automatische Backups, Point-in-Time-Recovery und Multi-AZ-Deployment für erhöhte Ausfallsicherheit. Die Kosten richten sich nach der genutzten Kapazität (ca. 0,12 USD pro ACU-Stunde), wobei die Datenbank bei Inaktivität auf null ACUs heruntergefahren werden kann, um Kosten zu sparen.

### 3.4 AWS Chime SDK

Das AWS Chime SDK ist der Kern der Echtzeitkommunikation und ermöglicht hochwertiges Video- und Audio-Streaming mit minimaler Latenz. Es unterstützt bis zu 250 Teilnehmer pro Meeting und bietet Funktionen wie Screen-Sharing, Chat, Teilnehmerverwaltung (z. B. Stummschalten oder Entfernen von Teilnehmern) und virtuelle Wartezimmer. Die Integration mit Lambda erlaubt eine dynamische Steuerung der Streams: Wenn ein neuer Teilnehmer hinzugefügt wird, erstellt eine Lambda-Funktion die entsprechenden Chime-Ressourcen und liefert dem Frontend die notwendigen WebSocket-URLs. Die Latenz liegt in optimalen Netzwerken bei unter 200 ms, was eine flüssige Kommunikation gewährleistet.

Ein Beispiel für die Verwendung des SDK in Node.js könnte so aussehen:

#### Verwendung der Chime SDK

```
const AWS = require('aws-sdk');
const chime = new AWS.ChimeSDKMeetings({ region: 'eu-central-1' });

async function startMeeting(meetingId) {
    const meetingResponse = await chime.createMeeting({
        ExternalMeetingId: meetingId
    }).promise();
    const attendeeResponse = await chime.createAttendee({
        MeetingId: meetingResponse.Meeting.MeetingId,
        ExternalUserId: 'user-456'
    }).promise();
    return { meeting: meetingResponse, attendee: attendeeResponse };
}
```

Dieser Code initialisiert ein Meeting, fügt Teilnehmer hinzu und gibt die notwendigen Informationen an das Frontend zurück, um den Stream aufzubauen. Das SDK ist besonders für die einfache Integration mit anderen AWS-Diensten und für die Instandhaltung gegenüber schwankenden Netzwerkbedingungen optimiert.



### 3.5 Amazon S3 für Transkriptionen

Amazon S3 wird verwendet, um Transkriptionen von Meetings zu speichern. Nach Abschluss eines Meetings kann eine Lambda-Funktion die Transkription an GPT, wo die Zusammenfassung entsteht, senden und das resultierende Textdokument wird in einem S3-Bucket gespeichert. Dies ermöglicht eine dauerhafte Speicherung und einen einfachen Zugriff auf Sitzungsprotokolle. Ein typisches Szenario wäre: Ein Meeting wird transkribiert, die Aufzeichnung wird an GPT gesendet, und die fertige Zusammenfassung wieder in S3 gespeichert.

S3 bietet zudem Versionierung, Lebenszyklusrichtlinien (z. B. zum Archivieren älterer Dateien in S3 Glacier) und Zugriffskontrollen via IAM, um die Sicherheit und Kosteneffizienz zu maximieren.

## 4. Kommunikation mit der Cloud Infrastruktur

### 4.1 Architektur und Datenfluss

Die Kommunikation zwischen dem Frontend (z.B. einer NextJS-Anwendung) und dem Backend erfolgt über zwei Hauptmechanismen: RESTful APIs und WebSockets. Das AWS API Gateway dient als Schnittstelle, die Anfragen an Lambda-Funktionen weiterleitet. Ein typischer Datenfluss sieht folgendermaßen aus:

1. Frontend sendet POST /meetings an API Gateway mit { "meeting\_id": "MEETING\_ID", "user-name": "TEST-USER" }
2. API Gateway triggert Lambda, das Chime initialisiert und Aurora aktualisiert.
3. Chime liefert einen WebSocket-Link, über den das Frontend den Stream aufbaut.

Ein Response würde dannn so aussehen:

*Response von API Gateway*

```
{  
  "Meeting": {  
    "MeetingId": "MEETING_ID",  
    "MediaPlacement": {  
      "AudioFallbackUrl": "wss://audio_url",  
      "AudioHostUrl": "wss://audio_url",  
      "ScreenDataUrl": "wss://screen_url",  
      "ScreenSharingUrl": "wss://screen_url",  
      "ScreenViewingUrl": "wss://screen_url",  
      "SignalingUrl": "wss://signaling_url",  
      "TurnControlUrl": "wss://turn_url"  
    }  
  },  
  "Attendee": {  
    "AttendeeId": "ATTENDEE_ID",  
    "ExternalUserId": "TEST-USER",  
  },  
}
```



```
"Capabilities": {  
    "Audio": "string",  
    "Content": "string",  
    "Video": "string"  
},  
"ExternalUserId": "string",  
"JoinToken": "string"  
}  
}
```



## 4.2 Technische Details

- RESTful APIs: Für asynchrone Operationen wie das Erstellen eines Meetings (HTTP POST /meetings).
- WebSockets: Für synchrone Kommunikation wie Video, Audio oder dem Watoriaum.
- Event-Driven Design: Lambda-Funktionen reagieren auf Ereignisse (z.B. „Benutzer hat das Meeting betreten“) und aktualisieren die Datenbank entsprechend.

## 4.3 Optimierung und Fehlerbehandlung

Um die Latenz zu minimieren, werden die Lambda-Funktionen in einer Virtual Private Cloud (VPC) in der Nähe der Aurora-Datenbank ausgeführt. Fehler wie Timeouts oder Datenbanküberlastung werden durch Retry-Mechanismen und Dead-Letter-Queues (DLQs) in AWS SQS behandelt. Dies gewährleistet eine hohe Verfügbarkeit und Zuverlässigkeit der Anwendung.

## 5. Kostenschätzung für den laufenden Betrieb

### 5.1 Grundlagen der Kostenberechnung

Die serverlose Architektur folgt einem Pay-per-Use-Modell, bei dem die Kosten direkt mit der tatsächlichen Nutzung zusammenhängen. Für eine realistische Schätzung wird von einer Anwendung mit 1.000 aktiven Benutzern pro Monat ausgegangen, die durchschnittlich 10 Meetings pro Tag mit jeweils 10 Teilnehmern (insgesamt 100 Teilnehmer pro Tag) abhalten.

### 5.2 Detaillierte Kostenaufstellung

- **AWS Lambda:**

- 1 Million Anfragen (z. B. Meeting-Starts, Chat-Nachrichten) à 0,20 USD/Million.
- 3 Millionen GB-Sekunden Rechenzeit (0,0000167 USD/GB-Sekunde) ≈ 50 USD/Monat.

- **Aurora Serverless:**

- Durchschnittlich 10 ACUs à 0,12 USD/Stunde × 720 Stunden/Monat ≈ 90 USD/Monat.

- **API Gateway:**

- 500.000 REST-Anfragen à 3,50 USD/Million + 100.000 WebSocket-Verbindungen à 1 USD/Million ≈ 2 USD/Monat.

- **S3 (für Transkriptionen):**

- 100 GB Speicher à 0,023 USD/GB + 10.000 PUT-Anfragen à 0,005 USD/1.000 ≈ 3 USD/Monat.

**Gesamtkosten:** ca. 145 USD/Monat. [\[11\]](#)

### 5.3 Skalierungsszenarien

- **10.000 Nutzer:** Kosten steigen auf ca. 1.400 USD/Monat (10-fache Anfragen/Daten).
- **100 Nutzer:** Kosten sinken auf ca. 20 USD/Monat (minimale Last).

### 5.4 Optimierungsmöglichkeiten

- **Caching:** AWS ElastiCache kann häufig abgerufene Daten (z. B. Meeting-Metadaten) zwischen-speichern, um Datenbankzugriffe zu reduzieren.
- **Batch-Verarbeitung:** Mehrere Ereignisse (z. B. Teilnehmeraktionen) können gebündelt in einer Lambda-Funktion verarbeitet werden, um Aufrufe zu minimieren.
- **Lifecycle Policies:** Alte S3-Dateien können in S3 Glacier verschoben werden, um Speicherkosten zu senken.



## 6. Erweiterung der Applikation

### 6.1 Einbindung von Websockets für einen Wateraum

Die Erweiterung einer Videokonferenzanwendung um einen „Wateraum“ ist eine anspruchsvolle Aufgabe, die eine gut durchdachte Backend-Logik braucht. Der „Wateraum“ ist ein virtueller Wartebereich, in dem Teilnehmer bleiben, bis der Gastgeber sie ins eigentliche Meeting lässt. Das ist praktisch, um Meetings sicherer und organisierter zu machen – etwa in Firmen oder Schulen, wo nicht jeder einfach reinkommen soll. Dafür muss der Server ständig und schnell mit den Nutzern kommunizieren, und zwar in Echtzeit. Wir setzen dafür socket.io ein, eine Technologie, die das super hinbekommt. In diesem Text schauen wir uns an, wie der Wateraum auf der Server-Seite aufgebaut wird, wie er gesteuert wird und wie er mit den Nutzern (Clients) abgestimmt bleibt. Außerdem gehen wir auf die Details ein, die das Ganze funktionieren lassen.

### 6.2 Einbindung von WebSockets für einen Wateraum mit Socket.io

Aus Sicht des Backends ist der „Wateraum“ ein zentraler Baustein. Er sorgt dafür, dass Teilnehmer verwaltet werden, dass der Gastgeber entscheidet, wer reinkommt, und dass alle sofort Bescheid wissen, wenn sich etwas ändert. Socket.io ist dafür ideal, weil es WebSockets nutzt – eine Technik, die Daten blitzschnell hin und her schickt. Dazu kommen praktische Funktionen wie Räume, mit denen man Nutzer in Gruppen einteilen kann, und Ereignisse, die bestimmte Aktionen auslösen. Wir verwenden Node.js als Basis, weil es viele Verbindungen gleichzeitig verarbeiten kann, ohne dass der Server hängen bleibt. Das ist perfekt für eine App, die in Echtzeit läuft, wie eine Videokonferenz.

#### Backend-Ziele:

- **Teilnehmerverwaltung:** Der Server muss wissen, wer gerade im Wateraum wartet und wer schon im Meeting ist. Das bedeutet, er braucht eine Übersicht über alle Nutzer und ihren Status.
- **Zugriffskontrolle:** Wenn jemand ins Meeting will, muss der Server die Anfrage prüfen und an den Gastgeber schicken. Der entscheidet dann, ob die Person reindarf oder nicht.
- **Echtzeit-Updates:** Sobald etwas passiert – zum Beispiel ein neuer Teilnehmer im Wateraum oder jemand wird reingelassen – müssen alle sofort informiert werden, damit niemand den Überblick verliert.
- **Skalierbarkeit:** Der Server soll auch dann stabil bleiben, wenn hunderte oder tausende Leute gleichzeitig mitmachen. Das ist wichtig für große Meetings oder Events.



## 6.3 Grundlegende Backend-Implementierung

Jetzt zeige ich, wie der Server den Watern Raum mit socket.io steuert. Dafür nehmen wir drei wichtige Situationen: Der Gastgeber tritt ein, ein Teilnehmer fragt an, und der Gastgeber lässt jemanden zu. Dazu gibt's Code-Beispiele, die zeigen, wie das technisch läuft. Wir nutzen auch AWS Chime, den AWS-Dienst für Videokonferenzen, um die Meetings zu verwalten.

Hier ist ein Teil Backend-Logik mit socket.io:

**1. Host tritt dem Meeting bei:** Der Gastgeber ist derjenige, der das Meeting startet und die Kontrolle hat. Ohne ihn läuft nichts.

- **Was passiert:**

- Der Gastgeber meldet sich mit seiner ID (userId) und der Meeting-ID (meetingId) beim Server.
- Der Server sucht in der Datenbank nach dem Namen des Gastgebers, damit alle wissen, wer da ist.
- Eine Nachricht geht an alle, die schon im Meeting sind, damit jeder Bescheid weiß, dass ein neuer Teilnehmer da ist.
- Der Gastgeber wird einem speziellen Raum nur für Gastgeber hinzugefügt (meeting- \${meetingId}). Von dort aus kriegt er später Anfragen von Teilnehmern.

### 1. - socketHelper.ts - Host tritt dem Meeting bei

```
socket.on('joinMeeting', async ({ userId, meetingId }: { userId: string, meetingId: string }) => {
    var attendeeName = await Attendee.findOne({ where: { uuid: userId },
    attributes: ['name'] }); ①
    io.to(`meeting-${meetingId}`).emit('userJoined', attendeeName); ②
    io.to(`meeting-attendee-${meetingId}`).emit('userJoined', attendeeName); ②
    socket.join(`meeting-${meetingId}`); ③
});
```

① Sucht den Namen des Hosts in der Datenbank.

② Sendet ein „userJoined“-Event an alle Teilnehmer im Meeting-Raum.

③ Fügt den Host dem Host-Raum hinzu.

**2. Teilnehmer fragt um Beitritt an:** Ein neuer Teilnehmer will ins Meeting, darf aber nicht einfach rein – er landet erst im Watern Raum.

- **Was passiert:**

- Der Teilnehmer schickt eine Anfrage mit seiner ID (requested\_id) und der Meeting-ID.
- Der Server überprüft die Anfrage in der Datenbank, um sicherzugehen, dass sie echt ist und zur richtigen Besprechung gehört.



- Der Gastgeber kriegt eine Nachricht, dass ein Teilnehmer im Wateraum wartet.
- Der Teilnehmer wartet jetzt im Wateraum, bis der Gastgeber entscheidet.

### 2. - socketHelper.ts - Teilnehmer fragt um Beitritt an

```
socket.on('joinRequest', async ({ requested_id, meetingId }: { requested_id: string; meetingId: string }) => {
    var requested = await Requested.findOne({ where: { uuid: requested_id, meeting_id: meetingId } });
    io.to(`meeting-${meetingId}`).emit('joinRequest', { requested_id, meetingId, name: requested.name });
});
```

① Sucht den Anfragenden in der Datenbank.

② Sendet ein „joinRequest“-Event an alle Hosts im Meeting-Raum.

### 3. Host genehmigt Beitritt: Der Gastgeber sieht die Anfrage und entscheidet, ob der Teilnehmer reindarf oder nicht.

#### • Was passiert:

- Der Gastgeber schickt eine Bestätigung mit der ID des Teilnehmers und der Meeting-ID.
- Der Server erstellt einen neuen Teilnehmer bei AWS Chime, damit er an der Videokonferenz teilnehmen kann.
- Die Infos über den Teilnehmer (Name, ID usw.) werden in der Datenbank gespeichert.
- Die Anfrage wird aus der Warteliste gelöscht, weil der Teilnehmer jetzt drin ist.
- Der Teilnehmer kriegt die Meeting-Daten (z. B. Zugangslink und ID), damit er beitreten kann.
- Alle im Meeting kriegen eine Nachricht, dass ein neuer Teilnehmer da ist.
- Der Teilnehmer wird einem Raum für normale Teilnehmer hinzugefügt (meeting-attendee- \${meetingId}).

Das ist der Moment, wo der Teilnehmer vom Warten ins Mitmachen wechselt. Der Server sorgt dafür, dass alles glatt läuft und jeder Bescheid weiß.

### 3. - socketHelper.ts - Host genehmigt Beitritt

```
socket.on('acceptRequest', async ({ requested_id, meetingId }: { requested_id: string; meetingId: string }) => {
    const requested = await Requested.findOne({ where: { uuid: requested_id, meeting_id: meetingId } });
    var uuid = v4();
    const awsAttendee = await chime.createAttendee({
        MeetingId: meetingId,
        ExternalUserId: uuid
```

```
}).promise(); ①
var attendee = await Attendee.create({
  uuid: uuid,
  name: requested.name,
  attendee_id: awsAttendee.Attendee.AttendeeId,
  meeting_id: requested.meeting_id,
  isHost: false
}); ②
var socket_id = requested.socket_id;
await requested.destroy();
const awsMeeting = await chime.getMeeting({ MeetingId: meetingId }).promise();
io.to(socket_id).emit('acceptedRequest', { Meeting: awsMeeting.Meeting,
Attendee: awsAttendee.Attendee }); ③
io.to('meeting-${meetingId)').emit('userJoined', attendee.name); ④
io.to('meeting-attendee-${meetingId}').emit('userJoined', attendee.name); ④
socket.join('meeting-attendee-${meetingId}'); ⑤
});
```

- ① Erstellt einen neuen AWS-Chime-Teilnehmer.
- ② Erstellt einen neuen Teilnehmer in der Datenbank.
- ③ Sendet ein „acceptedRequest“-Event an den anfragenden Teilnehmer.
- ④ Sendet ein „userJoined“-Event an alle Teilnehmer im Meeting-Raum.
- ⑤ Fügt den Teilnehmer dem Teilnehmer-Raum hinzu.

Hier ist auch ein Diagramm, das den Datenfluss zeigt:

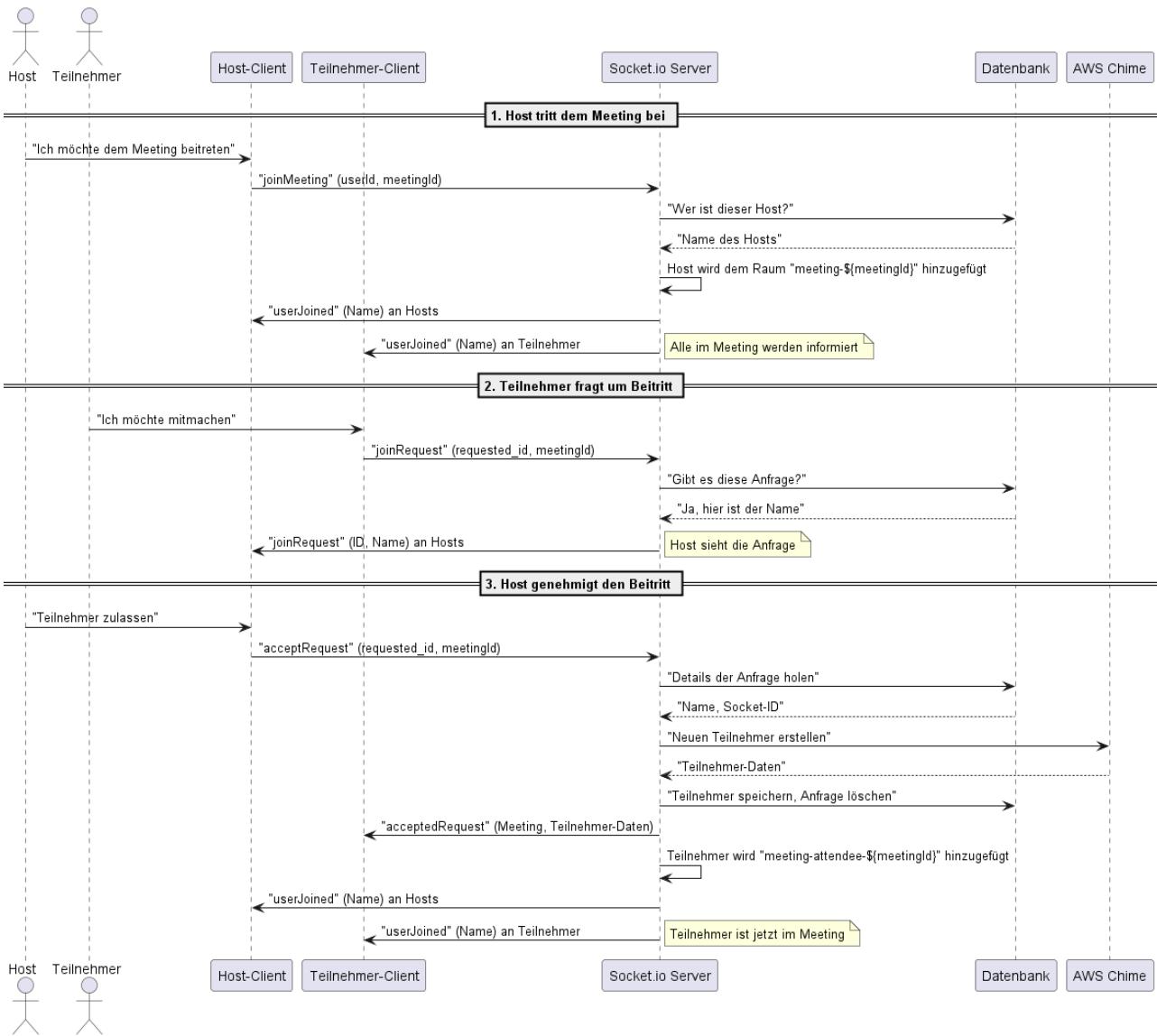


Abbildung 3. Diagramm für die Teilnehmerverwaltung

Diese Beispiele zeigen, wie Socket.io verwendet werden kann, um Echtzeitkommunikation in einer Videokonferenzanwendung zu ermöglichen. Die Implementierung erfordert eine sorgfältige Planung und Abstimmung zwischen Frontend und Backend, um eine reibungslose Benutzererfahrung zu gewährleisten.



## 6.4 Backend-Architekturüberlegungen

Damit der Watern Raum funktioniert, hat der Server eine klare Struktur. Hier sind die wichtigsten Teile:

- **Räume mit Socket.io:**

- meeting-\${meetingId}: Ein Raum nur für Gastgeber. Hier kommen Anfragen rein, und sie können entscheiden, wer rein darf.
- meeting-attendee-\${meetingId}: Ein Raum für normale Teilnehmer, die schon zugelassen sind. Sie kriegen Updates über neue Leute oder Änderungen.
- Der Watern Raum selbst ist kein echter Raum, sondern eine Liste in der Datenbank (Requested). Wer hier drin ist, wartet noch und hat keinen direkten Zugang.

- **Datenbank:**

- Attendee: Hier stehen alle, die im Meeting sind – Gastgeber und Teilnehmer. Zu jedem gibt's Infos wie Name, ID und ob er Gastgeber ist.
- Requested: Das ist die Warteliste. Hier landen Teilnehmer, die fragen, ob sie rein dürfen. Sobald der Gastgeber entscheidet, verschwinden sie aus dieser Liste.

- **Skalierung:**

- Bei großen Meetings mit hunderten Nutzern muss der Server wachsen können. Dafür kann man mehrere Server einsetzen und sie mit einem Load Balancer verbinden. Socket.io nutzt dann Redis, um Nachrichten zwischen den Servern zu teilen, damit alles synchron bleibt.
- Die Datenbank muss schnell bleiben, auch bei vielen Anfragen. Dafür kann man z. B. Indizes setzen, damit Suchen nicht lange dauern.

## 6.5 Warum das Ganze?

Der Watern Raum macht Meetings sicherer und übersichtlicher. Ohne ihn könnte jeder einfach reinplatzen, was vor allem bei wichtigen Besprechungen relevant ist. Der Gastgeber entscheidet, wer dabei ist, und alle kriegen live mit, was passiert. Das ist nicht nur praktisch, sondern auch professionell.



## Fazit

Aufgrund der Verwendung der serverlosen Infrastruktur von AWS konnte das Projekt erfolgreich umgesetzt werden. Die Kombination aus AWS Lambda, Aurora Serverless, AWS Chime SDK und Amazon S3 ermöglicht eine skalierbare, effiziente und benutzerfreundliche Videokonferenzanwendung. Die serverlose Architektur erlaubt eine dynamische Anpassung der Ressourcen an die Anzahl der Nutzer und sorgt für eine hohe Verfügbarkeit und Zuverlässigkeit der Anwendung. Die Implementierung von socket.io ermöglicht eine Echtzeitkommunikation zwischen Frontend und Backend und erweitert die Anwendung um einen virtuellen Wartebereich (Wateriaum), der die Sicherheit und Organisation von Meetings verbessert.

Die Kosten für den laufenden Betrieb der Anwendung wurden auf ca. 145 USD/Monat geschätzt, wobei verschiedene Optimierungsmöglichkeiten wie Caching, Batch-Verarbeitung und Lifecycle Policies zur Verfügung stehen, um die Kosten zu senken. Der Preis steigt mit der Anzahl der Nutzer, aber die serverlose Architektur erlaubt eine flexible Skalierung, um mit steigenden Anforderungen Schritt zu halten.

Insgesamt bietet die serverlose Architektur von AWS eine leistungsstarke und kosteneffiziente Plattform für die Entwicklung von Videokonferenzanwendungen. Durch die Kombination von serverlosen Diensten, Echtzeitkommunikation und Datenbanklösungen können Entwickler hochwertige Anwendungen erstellen, die den Anforderungen moderner Videokonferenzen gerecht werden.



## Literaturverzeichnis

- [1] Vgl. Synergy Research Group. *Cloud Market Share Report Q3 2024*. URL: [www.srgresearch.com/](http://www.srgresearch.com/) (abgerufen am 22.03.2025)
- [2] Vgl. Amazon Web Services. *AWS Lambda Introduction Whitepaper*, 2014. URL: [aws.amazon.com/whitepapers/lambda-introduction-2014](http://aws.amazon.com/whitepapers/lambda-introduction-2014) (abgerufen am 22.03.2025)
- [3] Vgl. Microsoft Azure. *Azure Functions Overview*. URL: [azure.microsoft.com/en-us/services/functions/overview](http://azure.microsoft.com/en-us/services/functions/overview) (abgerufen am 22.03.2025)
- [4] Vgl. Google Cloud. *Cloud Functions Documentation*. URL: [cloud.google.com/functions/docs](http://cloud.google.com/functions/docs) (abgerufen am 22.03.2025)
- [5] Vgl. Amazon Web Services. *Chime SDK Technical Documentation*. URL: [docs.aws.amazon.com/chime-sdk/latest/dg](http://docs.aws.amazon.com/chime-sdk/latest/dg) (abgerufen am 22.03.2025)
- [6] Vgl. Microsoft Azure. *Azure Communication Services Whitepaper*. URL: [azure.microsoft.com/en-us/resources/communication-services-whitepaper](http://azure.microsoft.com/en-us/resources/communication-services-whitepaper) (abgerufen am 22.03.2025)
- [7] Vgl. Google Cloud. *\_Cloud Spanner Documentation*. URL: [cloud.google.com/spanner/docs](http://cloud.google.com/spanner/docs) (abgerufen am 22.03.2025)
- [8] Vgl. Amazon Web Services. *Serverless Architecture Overview*, 2025. URL: [aws.amazon.com/whitepapers/serverless-architecture-2025](http://aws.amazon.com/whitepapers/serverless-architecture-2025) (abgerufen am 22.03.2025)
- [9] Vgl. Amazon Web Services. *AWS Lambda Documentation*. URL: [docs.aws.amazon.com/lambda](http://docs.aws.amazon.com/lambda) (abgerufen am 22.03.2025)
- [10] Vgl. Amazon Web Services. *Aurora Serverless Documentation*. URL: [docs.aws.amazon.com/aurora-serverless](http://docs.aws.amazon.com/aurora-serverless) (abgerufen am 22.03.2025)
- [11] Vgl. Amazon Web Services. *WS Pricing Calculator*. URL: [calculator.aws/](http://calculator.aws/) (abgerufen am 22.03.2025)



# Themenstellung von Philip Schrenk

Optimierung der Benutzeroberfläche einer Videomeeting-Anwendung: Eine Untersuchung der Effizienz und Nutzerfreundlichkeit von Next.js, React und Tailwind CSS für skalierbare Webanwendungen.

---

## 1. Einleitung und Begriffsdefinition

### 1.1 Einleitung

Die globale Pandemie des *SARS-CoV-19-Virus* im Jahr 2019 hat die Arbeitswelt nachhaltig verändert. Meetings, die vormals auf herkömmliche Weise in Person stattfanden, konnten plötzlich nicht mehr durchgeführt werden. Dies führte zu einer signifikant höheren Beliebtheit von Videomeetings-Anbietern wie Zoom, Teams und Google Meet. Die zunehmende Nutzung, auch durch unseren Partner *naos GmbH*, dieser Dienste hat die Notwendigkeit einer effizienten und benutzerfreundlichen Lösung für Videomeetings-Webanwendungen deutlich gemacht. Heutzutage sollten Webanwendungen so entwickelt werden, dass sie skalierbar und ausbaubar sind, und dabei helfen verschiedene JavaScript-Frameworks. Diese Diplomarbeit befasst sich mit der Skalierbarkeit und Nutzerfreundlichkeit solcher Frameworks am Beispiel von Next.js, React und Tailwind CSS.

### 1.2 Begriffsdefinition

Diese Diplomarbeit untersucht die oben genannten Frameworks im Hinblick auf Effizienz und Nutzerfreundlichkeit aus der Sicht des Entwicklers. Nun folgt eine kurze Definition der Begriffe, die in dieser Arbeit verwendet werden:

#### Effizienz

Effizienz bezieht sich hier darauf, wie einfach es für den Entwickler ist, die Funktionalitäten des Frameworks zu nutzen, wie schnell die Anwendung entwickelt werden kann und wie einfach sie in einen Workflow integriert werden kann.

#### Nutzerfreundlichkeit

Nutzerfreundlichkeit bedeutet, wie einfach es für einen Entwickler ist, Informationen über das Framework zu erhalten, und wie gut es dokumentiert ist.

#### Skalierbarkeit

Skalierbarkeit bezieht sich auf die Fähigkeit des Frameworks, mit wachsenden Anforderungen umzugehen, wie einfach es ist, neue Funktionen hinzuzufügen und wie gut es mit anderen Technologien integriert werden kann.

---



## 2. Next.js: Moderne Lösung zu Fullstack-Projekten

### 2.1 Next.js als Fullstack Framework

Was ursprünglich ein React-Framework war, das sich auf *Server-Side Rendering* (SSR) und *Static Site Generation* (SSG) spezialisiert hatte, ist nun ein vollständig optimiertes *Fullstack-Framework*, das von den größten Unternehmen verwendet wird. Während es für kleinere Projekte möglicherweise zu komplex ist, überzeugt es in größeren Anwendungen, wie oben definiert, durch Effizienz, Skalierbarkeit und Nutzerfreundlichkeit. [1]

Next.js basiert grundsätzlich auf React[1], was bedeutet, dass jeder in React geschriebene Code auch auf Next.js anwendbar ist. Allerdings ist Next.js nicht nur eine Optimierung von React, sondern so umfassend verändert, dass es als *eigenes Framework* betrachtet werden kann. Während React sehr dynamisch ist, bietet Next.js die kritische Struktur, die meiner persönlichen Meinung nach oft in React-Projekten fehlt. Next.js bietet eine Reihe von Vorteilen, darunter ein *Datei-basiertes Routing-System*, integrierte API- und Middleware-Routes und vieles mehr. Ein wesentlicher Vorteil von Next.js ist seine deutlich bessere Leistung, weshalb es nicht auf Webpack, sondern auf modernere Lösungen wie Turbopack[2] und Speedy Web Compiler basiert. Im Folgenden werden einige bemerkenswerte Features von Next.js aufgeführt, die so in React nicht umsetzbar sind:

#### Auflistung an nennenswerten Next.js Features

- **Einfache Integration von externen Bibliotheken:**

Next.js lässt sich problemlos mit gängigen Bibliotheken integrieren. Jede Bibliothek, die für React geschrieben ist, lässt sich ebenfalls für Next.js verwenden. Dies umfasst auch Komponentenbibliotheken wie die von *AWS Chime*, ohne deren Einsatz die Erstellung einer Anwendung wie der in diesem Diplomprojekt nicht möglich wäre.

- **Unterstützung für moderne JavaScript-Features:**

Next.js unterstützt moderne JavaScript-Features wie *ES6+*, *TypeScript* und *dynamische Importe*, was die Entwicklung effizienter und zukunftssicher macht. [3]

- **Übersichtliches Routing:**

Next.js stellt neuartige Lösungen für Component-Routes und API-Routes bereit. Mithilfe eines *Datei-Routing-Systems* können alle Routen der Webanwendung einfach als separate Datei im Verzeichnis */pages* oder */app* untergeordnet werden. Mit der Erstellung neuer Verzeichnisse können Seiten vernestet werden. Dies ermöglicht eine äußerst einfache Versionskontrolle durch Git und Übersicht für Entwicklerinnen und Entwickler. [4]

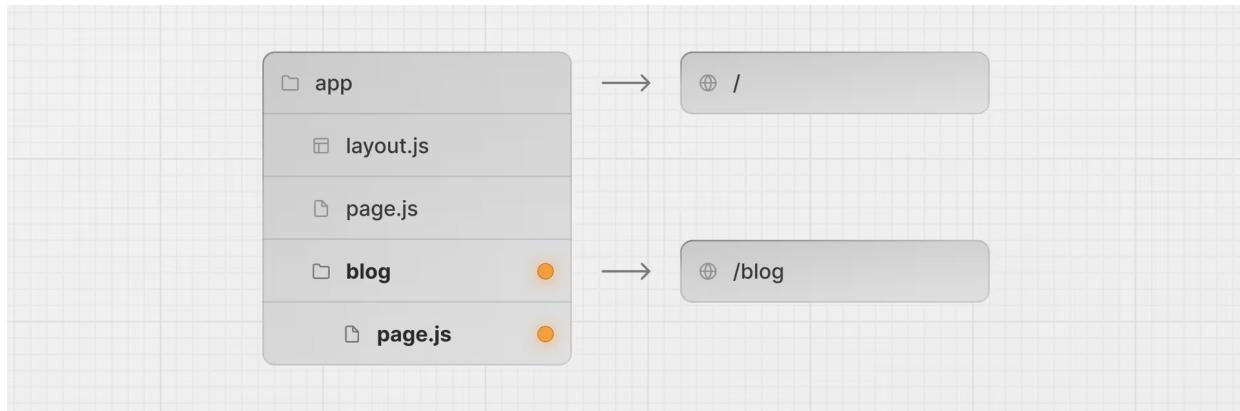


Abbildung 4. Visualisierung des Datei-Routing-Systems

- **Turbopack:**

Ein entscheidender Vorteil von Next.js ist die Nutzung von Turbopack, einem modernen Build-Tool, das Webpack ablöst. Turbopack wurde entwickelt, um die Build-Zeiten erheblich zu verkürzen, insbesondere bei großen Projekten mit vielen Dateien und Abhängigkeiten. Im Vergleich zu Webpack, das in diesem Projekt in einer früheren Phase verwendet wurde, reduzierte Turbopack die Zeit für einen vollständigen Build von etwa 15 Sekunden auf unter 5 Sekunden. Dies erhöht die Effizienz im Entwicklungsprozess, da Änderungen schneller sichtbar werden. [2]

- **SEO-freundliches Laden:**

SEO (Search Engine Optimized) ist bei modernen Frontend-Frameworks oft ein heißes Thema, weshalb die Seite klassisch als Barebone zurückgegeben wird und der Rest der Seite dann quasi mit JavaScript-Code "aufgebaut" wird. Ist dies nicht der Fall, kann es passieren, dass die Seite sehr lange zum Laden braucht und dem Nutzer in dieser Zeit kein Bild angezeigt werden kann. Das ist der Grund, warum man beim Besuch mancher Webseiten oft für einen kurzen Moment nur Weiß sieht. Next.js verhindert dies, indem es SEO bleibt, aber das Laden großer Komponenten oder *asynchroner Komponenten*, die von Daten aus dem Backend abhängen, dem Server überlässt. Diese werden dann normalerweise durch sogenannte *Spinner* oder *Skeletons* ersetzt, wie sie von Anwendungen wie TikTok verwendet werden, können aber auch so primitiv wie einfacher Text sein. Die während dieser Zeit angezeigte Seite wird als *Instant-Loading-State* bezeichnet. [5]

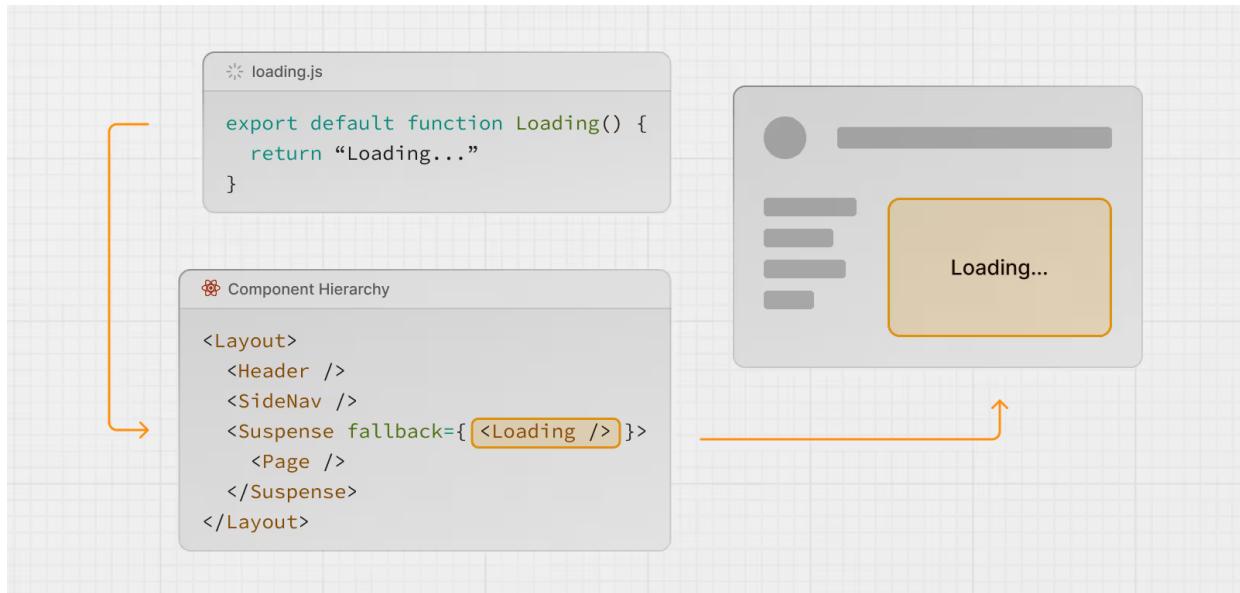


Abbildung 5. Visualisierung des Systems von Instant-Loading-States

- **Automatische Bildoptimierung:**

Next.js bietet eine integrierte Bildoptimierung über die Komponente `next/image`, die Bilder automatisch in moderne Formate wie WebP konvertiert, an die Bildschirmgröße des Nutzers anpasst und Lazy Loading unterstützt. Dies reduziert die Dateigröße und beschleunigt das Laden der Anwendung, was besonders in einer Videomeet-Anwendung nützlich ist, um Bandbreite zu sparen und die Performance zu verbessern. In diesem Projekt wurde die `next/image`-Komponente für statische Assets wie das Hintergrundbild des Warteraums verwendet, um die Ladezeiten der UI zu minimieren. [15]

Beispiel für die Nutzung von `next/image`

```
import Image from 'next/image'

export default function Page() {
  return (
    <Image
      src="/profile.png"
      width={500}
      height={500}
      alt="Picture of the author"
    />
  )
}
```

## 2.2 Next.js im Vergleich zu React

Um die Unterschiede zwischen Next.js und React eingehend zu beleuchten, ist es wichtig, ihre Kernfunktionen sowie ihre Einsatzmöglichkeiten im Hinblick auf die in der individuellen Themenstellung [definierten Kriterien](#) zu analysieren.

- **Effizienz:**

React bietet eine flexible Basis für dynamische UIs, benötigt aber externe Bibliotheken wie *React Router* oder *Axios* für Routing und Datenfetching, was den Entwicklungsaufwand erhöht. Next.js integriert diese Funktionen direkt, z. B. durch *dateibasiertes Routing* und *getServerSideProps*, was die Entwicklung schneller und effizienter macht.

- **Nutzerfreundlichkeit:**

React profitiert von einer großen Community, einer gut gepflegten Dokumentation und vielen Ressourcen. Next.js, das von Vercel unterstützt wird, kann praktisch auf der React-Community aufbauen, sodass alle Bibliotheken für React auch für Next.js zur Verfügung stehen und enthält eine klare, praxisorientierte Dokumentation, die mir persönlich bei der Entwicklung des Projekts sehr geholfen hat.

- **Skalierbarkeit:**

React ist aufgrund seiner Flexibilität skalierbar, erfordert aber eine sorgfältige Architekturplanung, z. B. durch Integration mit *Redux* oder *React Native* für mobile Anwendungen. Next.js erleichtert die Skalierung durch strukturierte Lösungen wie *API-Routen* und *Middleware*, die sich nahtlos mit Technologien wie *TypeScript* kombinieren lassen.

---

## 3. TailwindCSS: Ersatz zu herkömmlichen CSS

Tailwind CSS ist ein *Utility-First-CSS-Framework*, das Entwicklern ermöglicht, Benutzeroberflächen direkt im *JSX/TSX-Code* über vordefinierte Klassen zu gestalten. In diesem Projekt wurde es gewählt, um die UI-Entwicklung zu beschleunigen und eine einheitliche Designsprache für die Webvideocalls zu gewährleisten.[\[6\]](#)

### 3.1 Vorteile von TailwindCSS

- **Effizienz:**

Statt umfangreiche CSS-Dateien zu pflegen, ermöglicht TailwindCSS die direkte Anwendung im Code. TailwindCSS erstellt für jedes CSS-Attribut eine eigene CSS-Klasse, die dann in Next.js (und React) in einem konsekutiven String unter *className* angewendet werden kann. Dies ermöglicht es, in Projekten wie diesem, die nur wenig eigenes CSS benötigen, effizienter und unkomplizierter zu arbeiten als mit herkömmlichem CSS. [\[6\]](#)

- **Nutzerfreundlichkeit:**

Klare Klassennamen (wie z. B. *text-lg*), die übersichtliche Dokumentation und die *integrierte Extension* für Visual Studio Code ermöglichen auch Entwicklern, die mit dem Framework eher unerfahren sind, ein besseres Arbeiten. Außerdem kann TailwindCSS in neueren Versionen als



Standardoption für die Installation mit Next.js ausgewählt werden. [6]

- **Skalierbarkeit:**

Durch Anpassung der Datei `tailwind.config.js` können verschiedene Variablen, wie z. B. projektspezifische Farben und Schriftarten, einfach geändert werden. [6]

### 3.2 Integration mit Next.js und AWS Chime

Da AWS *Chime* einen Großteil des CSS von Haus aus mitbringt, ist es nicht notwendig, ein komplexes Design für die Seite zu entwerfen. Hier hilft TailwindCSS durch die kleine und performante Implementierung des CSS enorm.

Das folgende Beispiel zeigt einen Teil der `MeetingView.tsx`. Dieser Ausschnitt beschreibt den Einsatz des Frameworks in diesem Diplomprojekt sehr gut. Viele Sektionen sind mithilfe von drei Punkten aus dem Code ausgeblendet, um den Einsatz von TailwindCSS hervorzuheben.

*MeetingView.tsx - Beispiel für den Einsatz von TailwindCSS*

```
const MeetingView: React.FC<Props> = ({ ... }) => {
  // ...
  return (
    <div className="min-h-screen flex-col flex relative"> ①
      <CustomVideoTileGrid className="w-full min-h-screen" ... /> ①
      <div
        className="w-full flex justify-center bottom-0 absolute z-20 space-x-2"> ②
          <FloatingButton ... />
          {/* ... */}
        </div>
      </div>
    );
}
```

① Es ist zu beachten, dass hier `min-h-screen` zweimal verwendet wird. Der Grund dafür war ein Fehler von *AWS Chime*, bei dem die Video-Komponente nicht die gesamte Breite und Höhe der Webseite einnahm, weshalb es hier manuell überschrieben werden musste.

② Hier wird mit `absolute` und dem darüberliegenden `relative` das Positioning des "FloatingButton" auf absolut gesetzt. Dies dient dazu, dass die Position des Buttons immer gleich bleibt.

### 3.3 Vergleich zu herkömmlichen CSS

- **Responsive Design:**

Während konventionelles CSS verlangt, jede gewünschte Klasse mit einem angepassten `@media (min-width: ...px)` zu umschließen, benötigt TailwindCSS lediglich ein spezifisches Präfix vor der gewünschten Klasse (z. B. `md:text-base`, `sm:text-sm`). Dies erleichtert nicht nur die Übersicht, sondern auch die Konsistenz, sodass keine Zahlen manuell übertragen werden müssen. [6]



- **Dateigröße und Performance:**

Ein weiterer Unterschied besteht in der Dateigröße des generierten CSS. Tailwind CSS generiert während des Build-Prozesses eine optimierte `build.css`-Datei, die nur die tatsächlich verwendeten Klassen enthält. Werden z. B. nur drei Klassen verwendet, so hat die Datei auch nur die Größe von drei CSS-Klassen, im Gegensatz zu herkömmlichem CSS, bei dem oft unverwendete Klassen und Styles gespeichert werden. Dies ermöglicht TailwindCSS eine durchschnittliche Größe von unter 10kB. [6]

- **Anpassung an Branding:**

Ein oft übersehener Vorteil von Tailwind CSS ist die einfache Anpassung an das projektspezifische Branding. In diesem Projekt wurde die Datei `tailwind.config.js` verwendet, um eine benutzerdefinierte Farbpalette zu definieren, die den Vorgaben der naos GmbH entspricht. So wurde z. B. die Primärfarbe auf ein bestimmtes Blau (#005380) gesetzt, das als Hauptfarbe für Buttons und Titel verwendet wird.

*tailwind.config.js - Beispiel für Branding-Anpassung*

```
module.exports = {  
  theme: {  
    extend: {  
      colors: {  
        primary: '#005380',  
        secondary: '#00a0d1',  
      },  
    },  
  },  
};
```

- **Darkmode-Klassen:**

Ein sehr beliebtes Feature von Tailwind ist der Darkmode. `dark`-Klassen werden von Tailwind verwendet, um die Anwendung automatisch an die Systemeinstellungen des Benutzers anzupassen.

## 4. AWS Chime und Next.js als Lösung für Videomeet-Anwendungen

*AWS Chime* ist ein von Amazon entwickelter Dienst, der die Entwicklung eines Web-Videocalls durch vorgefertigte React-Komponenten, wie im Folgenden näher erläutert wird, aber auch durch *Serverless Backend-Infrastruktur*, wie in der individuellen Themenstellung von Bastian Seidl näher erläutert wird, vereinfacht.[7] Durch Verwendung von *WebRTC* (Web Real-Time Communication) ermöglicht *AWS Chime* die direkte Kommunikation zwischen den Teilnehmern, ohne dass ein Server als Vermittler fungieren muss. Dies ermöglicht eine schnellere und effizientere Kommunikation, da die Daten direkt zwischen den Teilnehmern ausgetauscht werden.[12]



Da Next.js, wie oben erwähnt, die Verwendung von React-Bibliotheken ermöglicht, kann es auch als Framework für *AWS Chime* verwendet werden. *AWS Chime* bietet nämlich eine komplette Open-Source-Komponentenbibliothek, die in React geschrieben ist und nützliche Hooks, Funktionen und vorgefertigte Komponenten bereitstellt. Diese Bibliothek ermöglicht es, ohne praktische Erfahrung eine Videomeet-Anwendung mit wenig Boilerplate-Code zu schreiben. [8]

Gleichzeitig erlaubt das Routing-System die strukturierte Definition von Meeting-spezifischen Endpunkten wie (bei diesem Diplomprojekt:) `/rooms/[id]` ohne zusätzliche Bibliotheken. Im Vergleich zu einer reinen *React*-Implementierung reduziert dies den Boilerplate-Code und erhöht die Effizienz, wie in der Begriffsdefinition beschrieben.

## 4.1 Terminologie von AWS Chime

In dieser Diplomarbeit werden einige englische Begriffe verwendet, die für Nichtprogrammierer etwas verwirrend sein können. Hier eine kurze Übersicht:

- **Host:**

Der/die Host ist grundsätzlich derjenige/diejenige, der/die das Meeting starten kann. Darüber hinaus hat er/sie die Aufgabe, die Rechte der Attendees zu verwalten, ihre Beiträtsanträge abzulehnen oder anzunehmen. Hosts werden durch individuelle Token definiert, die von einem bereits vom Kunden erstellten System an die Webanwendung übergeben werden.

- **Attendee:**

Jeder, der nicht über das oben genannte Token verfügt, ist ein/eine Attendee. Ein Attendee muss von mindestens einem Host autorisiert werden, um an der Sitzung teilnehmen zu können, andernfalls ist die Anwendung nicht benutzbar.

Normalerweise gibt es ein 1:n-Verhältnis zwischen Host und Attendee, da die Anwendung so konzipiert ist, dass die Tutoren mit ihren Kunden sprechen, aber ein n:n-Verhältnis verändert die Funktionsweise der Anwendung nicht.

## 4.2 Implementierung von AWS Chime

Bei der Recherche und Umsetzung waren die Dokumentationen[8][9] von *AWS Chime* sehr hilfreich. Die Integration von *AWS Chime* in Next.js beginnt mit der Installation der Bibliothek über NPM. Der *MeetingProvider* umhüllt den Programmeinstiegspunkt (`app.tsx`), um den *\_MeetingManager* global verfügbar zu machen. Dies ermöglicht den Zugriff auf Meeting-Funktionen wie das Starten oder Beenden eines Meetings von jeder Komponente aus.

`_app.tsx - Der "Applikationseinstiegspunkt"`

```
import { AppProps } from "next/app";
import { MeetingProvider } from "amazon-chime-sdk-component-library-react";

function MyApp({ Component, pageProps }: AppProps) {
  return (
    <MeetingProvider> ①
```



```
<Component /> ②
</MeetingProvider>
);
}

export default MyApp;
```

① *MeetingProvider* von AWS Chime.

② Der Rest der Applikation.

Der *MeetingProvider* dient als Kontextgeber, der den *MeetingManager* in der gesamten Next.js-Anwendung verfügbar macht. Dadurch können Komponenten wie Videoanzeigen oder Steuerungen effizient auf Meeting-Ereignisse reagieren, ohne dass manuell ein *State-Management* aufgebaut werden muss. [9]

*index.tsx - Benutzung innerhalb der Komponenten*

```
import { useMeetingManager } from "amazon-chime-sdk-component-library-react";
//...
const meetingManager = useMeetingManager();
```

Während Next.js und AWS Chime bereits eine solide Basis für Videocalls bieten, erfordern moderne Anwendungen oft zusätzliche Funktionen wie Wartebereiche oder benutzerdefinierte Designs. Diese werden in den nächsten Kapiteln näher erläutert.

### 4.3 Performance-Optimierung mit AWS Chime

Ein wichtiger Aspekt bei der Nutzung von AWS Chime ist die Möglichkeit, die Performance der Video- und Audioübertragung zu optimieren. Die WebRTC-Technologie ermöglicht es, die Bitrate dynamisch an die Netzwerkbedingungen der Teilnehmer anzupassen. In diesem Projekt wurde die Standardkonfiguration von AWS Chime verwendet, die eine maximale Videoauflösung von 720p bei 30 FPS vorsieht. Für Nutzer mit schwacher Internetverbindung wurde die Bitrate automatisch reduziert, um Verzögerungen zu minimieren.

Dieser Ansatz erhöht die Benutzerfreundlichkeit, da Teilnehmer unabhängig von ihrer Bandbreite an Meetings teilnehmen können, ohne dass die Qualität für alle reduziert werden muss. Zukünftige Optimierungen könnten eine manuelle Auswahl der Videoqualität ermöglichen, um den Nutzern mehr Kontrolle zu geben.



## 5. Gestaltung und Optimierung der Benutzeroberfläche

Die Welt der Videomeet-Anwendungen ist breit gefächert und bietet viele Möglichkeiten, die Benutzeroberfläche zu gestalten. Mit einem immer steigenden Markt, welcher einen Gesamtumsatz von 10,03 Milliarden USD in 2023 erzielt hatte, ist die Gestaltung der Benutzerfreundlichkeit oberste Priorität[13].

### 5.1 Optimierung der Benutzeroberfläche im Alleingang

Die Optimierung einer Benutzeroberfläche für Szenarien, in denen ein Nutzer alleine in einer Videokonferenz agiert, stellt spezifische Anforderungen an Übersichtlichkeit und Effizienz. In solchen Fällen liegt der Fokus darauf, unnötige Elemente zu reduzieren und die Darstellung auf den einzelnen Nutzer zuzuschneiden. Die Kombination von Next.js, Tailwind CSS und AWS Chime bietet hierfür geeignete Tools, um eine intuitive und performante Lösung zu schaffen.

Mit Next.js lässt sich die Benutzeroberfläche dynamisch an den Zustand „allein im Meeting“ anpassen. Durch die Verwendung von serverseitigem Rendering (SSR) oder statischer Seitengenerierung (SSG) kann der Zustand der Sitzung bereits vor dem Rendern ermittelt werden. Dies ermöglicht es, die Darstellung effizient zu steuern, etwa indem nur die Kamera des einzelnen Nutzers gerendert wird. Durch Backend- und Chime-Abfragen wie [RosterAttendee\[16\]](#) können die Teilnehmer geladen und angezeigt werden. Ist dies nicht der Fall, wird eine reduzierte UI-Komponente geladen, die auf den Allein-Nutzer optimiert ist.

Tailwind CSS unterstützt diese Optimierung durch seine Utility-Klassen, die eine schnelle Anpassung des Layouts erlauben. Die responsive Gestaltung bleibt erhalten, da Breakpoints wie `sm:` `md:` `lg:` die Größe an verschiedene Bildschirme anpassen. Zudem können Klassen wie `border-2 border-gray-300 rounded-lg` hinzugefügt werden, um die Kamera visuell abzugrenzen und die Wahrnehmung zu erleichtern.

AWS Chime trägt zur Optimierung bei, indem vorgefertigte Komponenten wie [VideoTile](#) genutzt werden, die sich automatisch an die Anzahl der Teilnehmer anpassen. Die [amazon-chime-sdk-component-library-react\[9\]](#) bietet Hooks wie `useActiveSpeakersState`, die den Status eines Meetings abfragen können. Bei einem einzelnen Nutzer wird die Videodarstellung entsprechend skaliert, ohne dass zusätzlicher Code geschrieben werden muss. Dies reduziert die Komplexität und sorgt für eine flüssige Performance, da die SDK interne Optimierungen wie Bandbreitenanpassung und Ressourcenmanagement übernimmt.

Die Implementierung dieser Arbeit sieht wie folgt aus:

### 5.2 Gestaltung der Anwendung

In diesem Projekt wurde die Benutzeroberfläche so gestaltet, dass sie einfach und intuitiv zu bedienen ist, wobei die Verwendung von *Tailwind CSS* für das Styling und die Wiederverwendung von *AWS Chime* Komponenten als ganzes vieles dazu beigetragen hat.



Während einer extensiven Recherche über existierende Designs, wurde festgestellt, dass viele Videomeet-Anwendungen oft überladen und unübersichtlich sind, weshalb sich mit dem unternehmen darauf geeinigt wurde, die Anwendung so einfach wie möglich zu gestalten, um die Benutzerfreundlichkeit zu erhöhen. Als Richtwert wurde hier mehrmals Google Meet vorgeschlagen, welches durch seine Einfachheit und Übersichtlichkeit überzeugte, und letztendlich als Vorbild für das Design der Anwendung diente.[\[14\]](#)

Die Anwendung wurde in zwei Hauptbereiche eingeteilt:

1. **Der Teilnehmerbereich**, welcher sich automatisch anpasst, je nachdem wie viele Teilnehmer sich im Meeting befinden. Sollten nur zwei Personen sich im Anruf befinden, rutscht die eigene Kamera in die rechts-untere Ecke, um die andere Person nicht zu verdecken. Sollte sich nur eine aktuell im Meeting befinden, ist dies ebenfalls der Fall.

Diese Anpassung der Kamerapositionen ist essentiell, um die Benutzerfreundlichkeit zu erhöhen, da es so nicht zu ungewollten Verdeckungen kommt. Dies wurde durch die Verwendung von *AWS Chime* Komponenten erreicht, welche diese Funktionalität bereits mitbringen. Ebenfalls beinhaltet die Kameraanzeige jeder einzelnen Person deren Namen, um mögliche Verwirrungen aus dem Weg zu schaffen.

2. **Die Steuerungsleiste**, welche sich unten in der Mitte der Applikation befindet, dient als Schnellzugriff auf wichtige Funktionen wie das Stummschalten des Mikrofons oder das Beenden des Meetings. Die Position des Feldes ist statisch, durch die *TailwindCSS* "absolute"-Klasse, festgelegt, um sicherzustellen, dass es immer die gleiche Position behält.

Die Leiste dient vor allem dazu, oft verwendete Funktionen schnell und einfach zu erreichen, ohne mehrere Klicks verschwenden zu müssen. Ebenfalls sind die Buttons mit Benutzer-vertrauten Icons ausgestattet. Dies erhöht die Benutzerfreundlichkeit, da so nicht lange nach den Funktionen gesucht werden muss.

Die nachstehende Abbildung zeigt einen Ausschnitt aus unserer Applikation, in der ein Benutzer, genannt "Guest", alleine in der Sitzung ist. Zu sehen sind beide Bereiche.

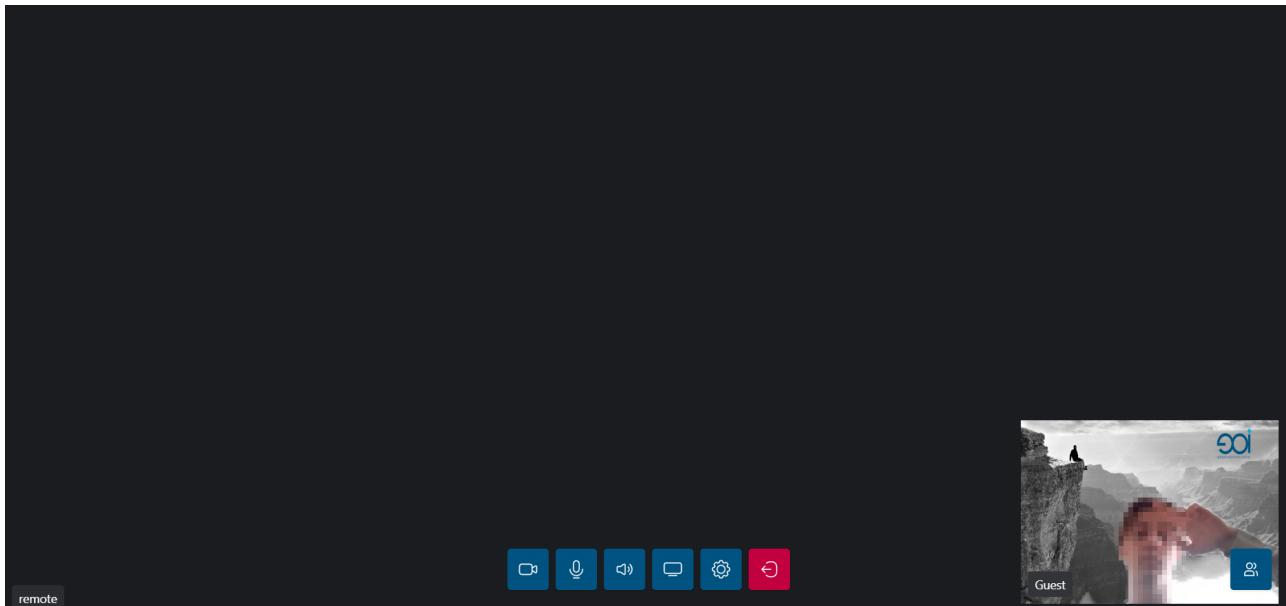


Abbildung 6. Screenshot der Applikation

## 6. Erweiterung der Applikation

### 6.1 Einbindung eines Warteraums

#### 6.1.1 Technische Grundlagen

Die Integration eines Warteraums war eine Anforderung dieses Diplomprojektes. Die ersten Ideen für die Umsetzung waren klar: Wir brauchten eine Art Socket-Verbindung zwischen Client und Server, die nicht nur den Nachrichtenaustausch ermöglicht, sondern auch speichert, wer gerade was gesendet hat. Wie sich letztendlich herausstellte, war dies nicht ausschließlich mit *AWS Chime* möglich, sondern nur mit Hilfe der Integration mit dem Package *SocketIO*.

*SocketIO* ist sowohl ein Backend- als auch ein Frontend-Package, welches das Versenden und Erstellen von Socket-Nachrichten wesentlich vereinfacht[10]. In Absprache mit dem Backend haben wurde sich für diesen Ablauf der Anfragen entschieden, der in Form eines Sequenzdiagrammes in *PlantUML* dargestellt wurde:

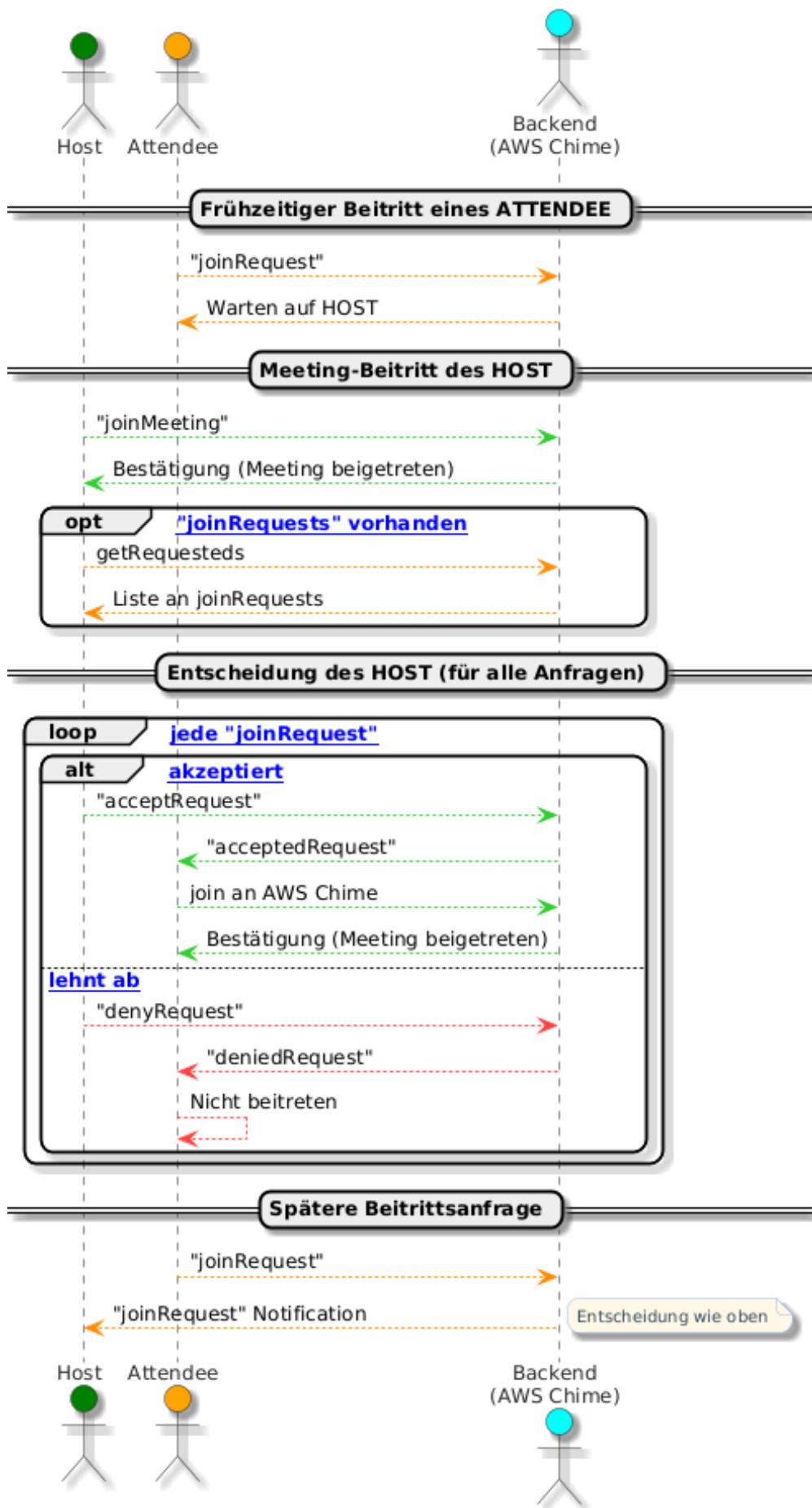


Abbildung 7. Visualisierung des Warterraum-Prozesses inkl. Host



Die Rollen des **Host** und **Attendee**, wie in der Therminologie von AWS Chime beschrieben, werden hier nochmals ersichtlich gemacht, daher der Attendee, wie bereits erwähnt, auf die Berechtigung des Host warten muss um Beitreten zu können.

### 6.1.2 Theoretische Implementierung im Frontend

Der nächste Schritt war die Implementierung des Sockets im Frontend. Der ursprüngliche Ansatz führte zu unnötigen Neuladungen der Seite, was mit Single-Page-Application-Prinzipien kollidierte. Jedoch wurde dieser dann durch einen besseren ersetzt, welcher mehrere React-Contexte und -Hooks, welche primär zur Speicherung von Beitrittsanfragen und zur Kommunikation mit dem Backend dienen, implementiert.

*Was passiert im Frontend wenn ein Attendee eine Anfrage sendet?*

1. Sobald ein/e Benutzer/in die Seite betritt, werden die Warteraumkomponenten geladen. Noch werden keine Komponenten geladen, abgesehen vom React Context Provider **JoinRequestsProvider**, welcher immer aktiv ist und sich nur updated, sobald eine neue Request vom Backend kommt. Daher zur Zeit noch keine konkrete Verbindung zum Backend besteht, ist dieser leer.
2. Unmittelbar nachdem, ein/e Benutzer/in den Knopf "Meeting Starten" oder "Beitrittsanfrage senden" betätigt hat, wird eine Verbindung zu unserem Socket mit Hilfe von **useSocket** und **SocketHelpers** und dem Socket von AWS Chime hergestellt. Damit wird die Socket-Message "joinMeeting", oder "joinRequest" für Attendees, zum Beitreten geschickt. Welche im letzten Fall, auch nun allen Hosts angezeigt wird.
3. Beim Laden der Seite wird die Komponente MeetingView, und damit auch **SocketEventProvider**, geladen und dessen useEffect aufgerufen. Das useEffect, in **SocketEventProvider**, erstellt Listener für diverse eingehende Socket-Nachrichten, unter anderem "joinRequest". Dieser ist nur nützlich für Hosts, daher auch nur Hosts "joinRequests" empfangen können.
4. Im Falle eines Host-Clients, wartet das System danach auf Beitrittsanfragen. Wenn eine Beitrittsanfrage kommt, wird sie von **SocketHelpers** abgefangen. Da **SocketHelpers** als Hook selbst keine *Client-Component Features* ausführen kann, wird der Listener, samt Callback, in **SocketEventProvider** ausgeführt.
5. Der Callback im **SocketEventProvider** ruft danach **JoinRequestsProvider** auf, um die Liste der Beitrittsanfragen zu aktualisieren. Jede Komponente, die nun diese Liste verwendet, wird automatisch mit den Änderungen aktualisiert, da es sich hier, wie schon erwähnt, um einen React Context Provider handelt.
6. Es ist nun Aufgabe der Frontend-Komponenten, diese Benachrichtigung, samt den Optionen zum Annehmen und Ablehnen, anzuzeigen. Nach der Annahme oder Ablehnung der Anfrage wird die **useSocket** Methode *emit()* direkt vom **JoinRequestsProvider** aufgerufen.

Folgendes Sequenzdiagramm, welches ebenfalls mit PlantUML geschrieben wurde, soll diese Abläufe darstellen:

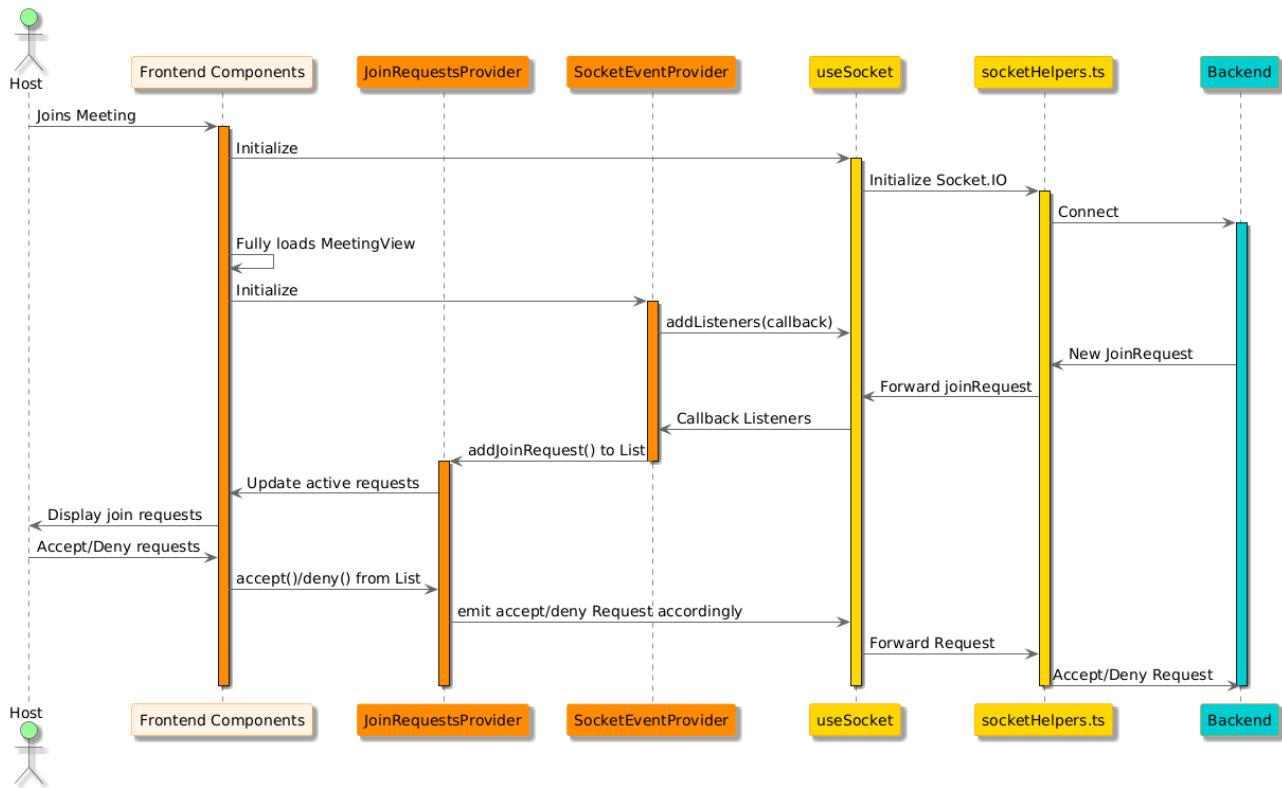


Abbildung 8. Visualisierung der Frontend-Komponenten für das Socket und deren Verbindungen

### 6.1.3 Edge-Cases und Warteschlange

Doch man darf nicht immer davon ausgehen, dass ein Attendee, immer nach einem Host dem Meeting beitreten will. Nach dem oben beschriebenen System, würde ein Attendee, welcher zu früh eine Anfrage schickt, einfach ignoriert werden, da kein Host da ist, um die Anfrage zu empfangen. Um diesen Edge-Case entgegenzukommen, haben wir zur Sicherheit eine Warteschlange eingebaut, die alle Anfragen speichert, bis ein Host da ist. Dieser Ablauf ist ebenfalls in Abbildung 7 dargestellt.

*Was passiert wenn ein Attendee vor einem Host eine Anfrage schickt?*

1. Wenn ein Host das Meeting startet, und der Komponente **SocketEventProvider** initialisiert ist, wird eine Anfrage an das Backend geschickt, welche alle aktuell wartenden Attendees als Liste zurück gibt.
2. Diese Liste wird dann in **JoinRequestsProvider** gespeichert und an alle Komponenten, die diese Liste verwenden, weitergegeben.
3. Das Frontend zeigt dann die Anfragen an, und der Host kann sie annehmen oder ablehnen, genau wie oben.

### 6.1.4 Umsetzung

Möglicherweise sind die oben genannten Vorgänge schwer vorzustellen, da hier ein wichtiger Einstiegspunkt fehlt. Dieser Einstiegspunkt ist die Funktion **joinMeeting**, welche die Verbindung zum Backend herstellt und die Anfrage sendet. Diese Funktion ist in der Datei **uuid.tsx** zu finden, da diese Datei die Route **/room/HOSTID** repräsentiert, in der sich der Warteraum mit dem Button zum



Beitreten befindet. Der relevante Code ist hier aufgeführt:

[uuid.tsx] - Initialisierung des Sockets nach Beitreten (nur kontextuell wichtiger Code)

```
const meetingManager = useMeetingManager();
const { initSocket, emit, addListener } = useSocket();
const joinMeeting = async () => {
    const data = await fetchMeeting( ①
        uuid as string,
        username as string,
        token as string | undefined
    );

    initSocket(); ②

    if (isAwaitType(data)) { ③
        emit(AttendeeSocketEmitType.JOINREQUEST, { ④
            requested_id: data.requestedId,
            meetingId: data.meetingId,
        });
        console.log("AWAITING APPROVAL");
        //...
    } else {
        // ...
        const meetingSessionConfiguration = new MeetingSessionConfiguration(
            data.Meeting,
            data.Attendee
        );
        await meetingManager.join(meetingSessionConfiguration, { ⑤
            deviceLabels: DeviceLabels.AudioAndVideo,
        });
        await meetingManager.start();
    }
}
```

① Holt die Meetingdaten von AWS Chime. Essentiell wichtig zum beitreten der Anwendung

② Baut die Verbindung zu unserem Socket auf.

③ Prüft, ob die Antwort des Servers vom Typ **AwaitType** ist. Wenn ja, muss auf eine Berechtigung gewartet werden.

④ Schickt die Socket-Nachricht "joinRequest" an das Backend

⑤ Ist die Antwort kein **AwaitType**, so wird direkt dem Meeting beigetreten.



## 6.2 Einbindung von Hintergrundeffekten

Viele Videomeet-Anwendungen bieten mittlerweile die Möglichkeit, den Hintergrund zu verschwemmen, um so, mehr oder weniger, die Privatsphäre der Benutzer zu schützen. Aufgrund dessen wurde dies ein weiteres Feature, welches im Frontend umgesetzt wurde.

Dank *AWS Chime*, welches ebenfalls durch gründliche Dokumentation und eine Vielzahl von Beispielen unterstützt wird, war die Implementierung dieses Features relativ einfach. Die Implementierung erfolgte konkret in der Komponente `MeetingView`, welche die Hauptkomponente für die Anzeige des Videomeetings ist. Diese Komponente wurde dann lediglich mit der Komponente `BackgroundBlurProvider` umhüllt, welche die Funktionalität des Hintergrundeffekts bereitstellt. Mit einem einfachen Button-klick, kann dann der Hook `useBackgroundBlur` verwendet werden, um den Effekt zu aktivieren oder deaktivieren. Folgendes Code-Beispiel stammt aus der Dokumentation von *AWS Chime*: [11]

*Implementierung des Hintergrundeffekts*

```
const meetingManager = useMeetingManager();
const { selectedDevice } = useVideoInputs();
const { isBackgroundBlurSupported, createBackgroundBlurDevice } =
  useBackgroundBlur();
const [isVideoTransformCheckBoxOn, setisVideoTransformCheckBoxOn] =
  useState(false);

async function toggleBackgroundBlur() {
  try {
    let current = selectedDevice;
    if (isVideoTransformCheckBoxOn) {
      current = await createBackgroundBlurDevice(selectedDevice); ①
    } else {
      if (isVideoTransformDevice(selectedDevice)) {
        current = await selectedDevice.intrinsicDevice(); ②
      }
    }
    await meetingManager.startVideoInputDevice(current);
  } catch (error) {
    console.error('Failed to toggle Background Blur');
  }
}
```

① Erstellt ein neues Kamera-"Device", welches den Hintergrund verschwemmt.

② Setzt das Kamera-"Device" auf die normale Kamera zurück, um den Effekt zu deaktivieren.

## Fazit

Durch die Verwendung von **Next.js** als Full-Stack-Framework konnten wesentliche Vorteile wie serverseitiges Rendering und statische Seitengenerierung genutzt werden, die für die Performance und SEO-Optimierung entscheidend sind. **React** bildet die Grundlage für eine komponentenbasierte Architektur, die Modularität und Wiederverwendbarkeit fördert, was insbesondere bei der Gestaltung dynamischer Benutzeroberflächen von Vorteil ist. **Tailwind CSS** ermöglicht durch seinen Utility-First-Ansatz ein schnelles und konsistentes Design, das speziell an die Anforderungen einer Videokonferenzanwendung angepasst werden kann. Die Integration der vorgefertigten Komponenten von **AWS Chime** mit **Tailwind CSS** erleichterte zudem die Erstellung einer intuitiven und ansprechenden Benutzeroberfläche.

Aus Effizienzgründen ermöglicht die Integration dieser Technologien die Reduzierung des Boilerplate Codes und die Nutzung der eingebauten Optimierungen, so dass sich die Entwickler auf die Kernfunktionalität konzentrieren können. Die Benutzerfreundlichkeit wird durch umfassende Dokumentation, aktive Communities und intuitive APIs unterstützt, die den Einstieg erleichtern und schnelle Entwicklungszyklen ermöglichen.

Zusammenfassend kann gesagt werden, dass **Next.js**, **AWS Chime** und **Tailwind CSS** eine solide Basis für die Entwicklung von skalierbaren und benutzerfreundlichen Videokonferenzanwendungen bieten. Sie erfüllen die Anforderungen der modernen Webentwicklung und bieten eine ausgewogene Mischung aus Performance, Flexibilität und Benutzerfreundlichkeit.[\[17\]](#)

## Literaturverzeichnis

- [1] Vgl. Vercel *Next.js by Vercel - The React Framework*. URL: [nextjs.org](https://nextjs.org) (abgerufen am 12.03.2025)
- [2] Vgl. Vercel. *API Reference: Turbopack*. URL: [nextjs.org/docs/app/api-reference/turbopack](https://nextjs.org/docs/app/api-reference/turbopack) (abgerufen am 27.02.2025)
- [3] Vgl. Vercel. *Next.js by Vercel - The React Framework*. URL: [nextjs.org/](https://nextjs.org/) (abgerufen am 27.02.2025)
- [4] Vgl. Vercel. *Routing Fundamentals*. URL: [nextjs.org/docs/app/building-your-application/routing](https://nextjs.org/docs/app/building-your-application/routing) (abgerufen am 27.02.2025)
- [5] Vgl. Vercel. *Loading UI and Streaming*. URL: [nextjs.org/docs/app/building-your-application/routing/loading-ui-and-streaming](https://nextjs.org/docs/app/building-your-application/routing/loading-ui-and-streaming) (abgerufen am 27.02.2025)
- [6] Vgl. Tailwind Labs. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: [tailwindcss.com](https://tailwindcss.com) (abgerufen am 05.02.2025)
- [7] Vgl. Amazon Web Services. *Videokonferenzen und Online-Meetings - Amazon Chime*. URL: [aws.amazon.com/de/chime/](https://aws.amazon.com/de/chime/) (abgerufen am 12.03.2025)
- [8] Vgl. Amazon Web Services. *aws/amazon-chime-sdk-component-library-react*. URL: [github.com/aws/amazon-chime-sdk-component-library-react](https://github.com/aws/amazon-chime-sdk-component-library-react) (abgerufen am 12.03.2025)
- [9] Vgl. Amazon Web Services. *Introduction - Page - amazon-chime-sdk-component-library-react*.



URL: [aws.github.io/amazon-chime-sdk-component-library-react/](https://aws.github.io/amazon-chime-sdk-component-library-react/) (abgerufen am 02.03.2025)

- [10] Vgl. Automattic. *Introduction | Socket.IO*. URL: [socket.io/docs/v4/](https://socket.io/docs/v4/) (abgerufen am 12.03.2025)
- [11] Amazon Web Services. *SDK Hooks / useBackgroundBlur*. URL: "https://aws.github.io/amazon-chime-sdk-component-library-react/?path=/docs/sdk-hooks-usebackgroundblur—page" (abgerufen am 19.03.2025)
- [12] Vgl. Amazon Web Services. *How Amazon Chime SDK meetings use WebRTC media*. URL: [docs.aws.amazon.com/chime-sdk/latest/dg/webrtc-media.html](https://docs.aws.amazon.com/chime-sdk/latest/dg/webrtc-media.html) (abgerufen am 20.03.2025)
- [13] Grand View Research. *Video Conferencing Market Size & Trends*. URL: [www.grandviewresearch.com/industry-analysis/video-conferencing-market](https://www.grandviewresearch.com/industry-analysis/video-conferencing-market) (abgerufen am 20.03.2025)
- [14] Vgl. Google. *Videoanrufe- und konferenzen für alle*. URL: [meet.google.com/landing?pli=1](https://meet.google.com/landing?pli=1) (abgerufen am 20.03.2025)
- [15] Vercel. *Image*. URL: [nextjs.org/docs/pages/api-reference/components/image](https://nextjs.org/docs/pages/api-reference/components/image) (abgerufen am 22.03.2025)
- [16] Amazon Web Services. *Using the meeting roster*. URL: [docs.aws.amazon.com/chime/latest/ug/attendees-panel.html](https://docs.aws.amazon.com/chime/latest/ug/attendees-panel.html) (abgerufen am 22.03.2025)
- [17] xAi Grok 3, X. Eingabe: *Schreibe mir ein Fazit zu dieser Arbeit, was auf die Themenstellung "... aufgreift*. URL: [grok.com/?referrer=website](https://grok.com/?referrer=website) (abgerufen am 24.03.2025)



# Themenstellung von Louis Muhr

Automatisierte Bereitstellung und Skalierung serverloser Anwendungen: Best Practices für CI/CD-Pipelines in AWS unter Verwendung von GitLab CI für Videomeeting-Lösungen.

## 1. Einleitung

Die fortschreitende Digitalisierung seit den 2000ern, die Corona-Pandemie und der wachsende Bedarf an automatisierten und skalierbaren Anwendungen haben den Fokus sowohl auf Video-meeting-Anwendungen, aber auch auf sonstigen serverlosen, automatisierten & skalierbaren Softwareprogrammen gelenkt. Diese Arbeit widet sich dem Thema der automatisierbaren Bereit-stellung und Skalierung von serverlosen Anwendungen und bespricht dabei auch Best Practices für CI/CD-Pipelines in AWS unter Verwendung von GitLab CI.

Das Ziel von dieser Arbeit ist es, die Grundlagen von Serverless Computing und CI/CD mit der praktischen Umsetzung zu verbinden. Ebenfalls soll aufgezeigt werden, wie Best Practices diese Entwicklung erleichtern und verbessern können.

### 1.1 Wichtige Begriffe

#### Pipeline

So nennt man eine automatisierte Abfolge von Schritten, diese steuert den Entwicklungspro-zess einer Software von der Code-Integration bis hin zur Bereitstellung. Die Pipeline ist norma-lerweise in drei Stufen eingeteilt:

1. **Build** - In diesem Schritt wird der Code kompiliert und in eine ausführbare Form gebracht
2. **Test** - Um Qualität und fehlerfreiheit zu garantieren werden automatische Tests ausge-führt
3. **Deploy** - Der fertiggetestete Code wird in eine Produktionsumgebung ausgerollt

#### Job

Im Sinne dieser Arbeit ist mit dem Begriff "Job" immer eine einzelne Aufgabe innerhalb einer CI/CD Pipeline gemeint. Er besteht aus einem Namen, es wird die Phase definiert in welcher er sich befindet und es werden Befehle erteilt, welche bei der Ausführung in einer CLI-Umge-bung durchgeführt werden.

#### Best Practices

Ist ein Begriff, welcher verwendet wird um funktionierte Methoden bzw. Vorgehensweisen zu bezeichnen. Diese haben sich meist als effizient, zuverlässig und effektiv erwiesen und sollen vor allem dabei helfen bei der Fertigstellung von Projekten.

## Command Line (CLI)

Im deutschen auch Befehlzeile, ist ein textbasiertes Interface, dies hilft dabei ein System (wie z.B. Windows) ohne grafische Benutzeroberfläche zu steuern. So sieht das in Windows aus:

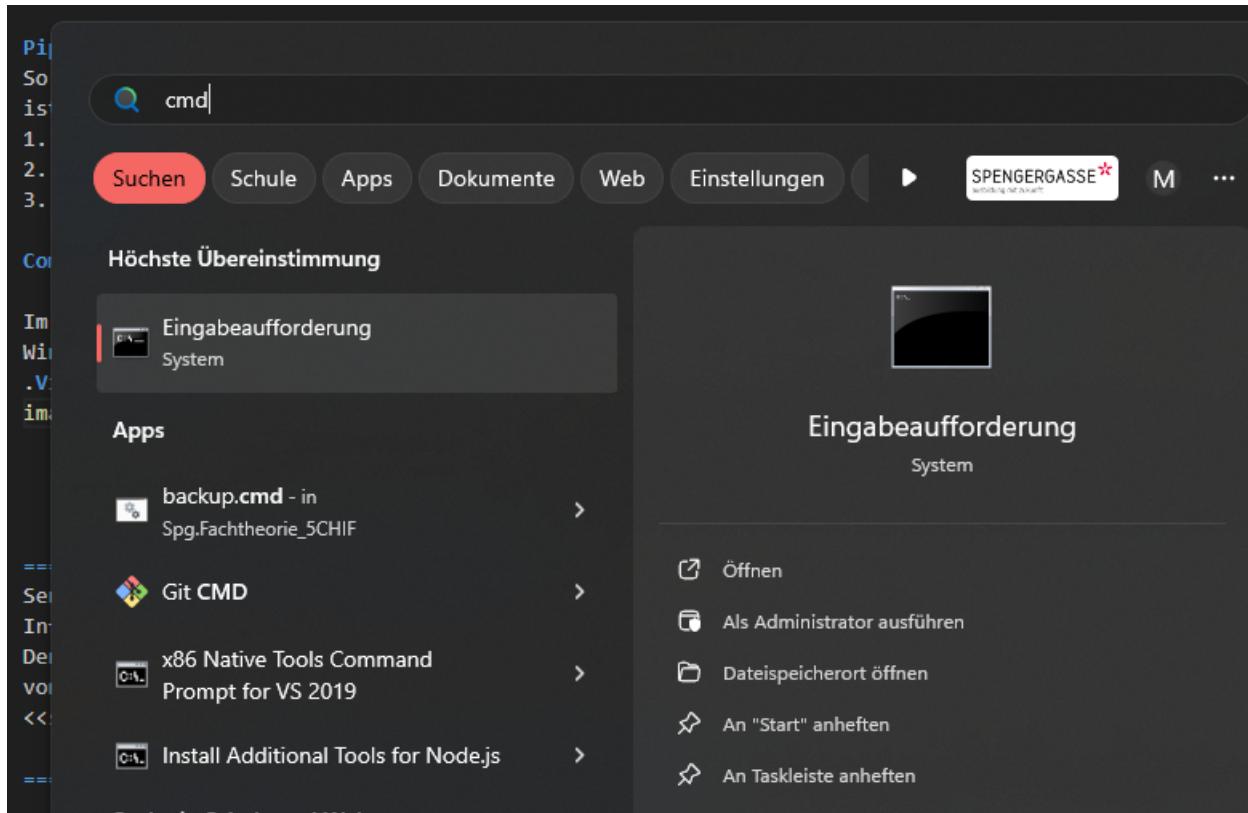


Abbildung 9. CLI in Windows

## AWS

AWS steht für Amazon Web Services und ist eine Cloud-Computing-Plattform von Amazon. Es werden unendlich viele Dienste wie z.B. Datenbanken (Aurora), Rechenleistung (z.B. Lambda), aber auch Kommunikationslösungen (z.B. Chime) bereitgestellt. Diese Dienste helfen Entwicklern dabei serverlose Anwendungen zu erstellen.

## Deployment

Ins Deutsche übersetzt ist es die Bereitstellung, so wird der Prozess genannt, welcher eine Anwendung in die Zielumgebung übertragen wird. Durch diesen Schritt wird die Anwendung für den Verbraucher zur Verfügung gestellt. In CI/CD wird das Deployment automatisiert.

## 1.2 Was sind serverlose Anwendungen

Serverlose Anwendungen (engl. serverless Applications) erleichtern es Entwicklern ihre Anwendung effizient und unkomplizierter umzusetzen. Der Cloud Anbieter stellt dabei die Infrastruktur, welche zur Codeausführung, Skalierung und Verwaltung des Programmes benötigt wird, zur Verfügung. Der Begriff serverlos kommt daher, dass die Aufgaben rund um Bereitstellung und Verwaltung des Codes unsichtbar sind und nicht weil keine Server verwendet werden. Mithilfe von serverless Applications können sich Entwicklerteams viel besser auf den Code konzentrieren und dadurch Software schneller und mit höherer Qualität produzieren. (vgl. [1])



## 1.3 Warum Automatisierung von Software-Deployments

- **Geschwindigkeit und Effizienz:**

Manuelle Deployments kosten Zeit und sind fehleranfällig, mithilfe einer Automatisierung von Software-Deployments, kann Zeit gespart werden und Unternehmen können ihre Produkte schneller an die Marktveränderungen anpassen, ebenfalls können sie eher fehlerfreie Software gewährleisten.

- **Menschliche Fehler:**

Bei einem manuellen Prozess können Fehler, Inkonsistenzen oder andere Probleme auftreten, welche wiederum Zeit in Anspruch nehmen würden. Vordefinierte Skripts und Pipelines können konsistent ausgeführt werden und ermöglichen somit automatisierte Deployments.

- **Skalierbarkeit:**

Bei riesigen Projekten wie z.B. bei Microsoft-Architekturen oder Google Anwendungen ist es unmöglich alle Dienste manuell zu aktualisieren, weil es einfach zu viele werden. Mithilfe der Automatisierung, wird es mögliche Deployments skalierbar und parallel durchzuführen, dies ist besonders bei Cloud-Umgebungen wie z.B. AWS von Vorteil. (vgl. [2])

## 2. Grundlagen

### 2.1 Was ist CI/CD

#### 2.1.1 Definition

Ci/CD ist ein Begriff aus der Softwareentwicklung und steht für Continuous Integration / Continuous Deployment (kontinuierliche Integration / kontinuierliche Bereitstellung) und manchmal für Continuous Delivery (kontinuierliche Auslieferung). Dabei handelt es sich um eine Methode, welche genutzt werden kann um den Entwicklungsprozess zu automatisieren und die Qualität beziehungsweise die Geschwindigkeit der Softwarebereitstellung zu verbessern.

#### 2.1.2 Agile Softwareentwicklung mit CI/CD

Sowohl agile Softwareentwicklung als auch CI/CD, zielen darauf ab einen Entwicklungsprozess flexibel, schnell und mit hoher Qualität abzuwickeln, daher ergänzen sie sich so gut.

- **Agile vs Scrum:**

Der große Unterschied zwischen Agile und Scrum liegt darin, dass Agile eine Philosophie ist, welche die Scrum Methode umsetzt. **Also einfach gesagt:** Agile ist die Idee und Scrum eine Umsetzungsmöglichkeiten dieser Idee.

- **Agile:**

Agile beschreibt einen flexiblen Ansatz für Softwareentwicklung und Projektmanagement, dieser legt Wert auf schnelle Anpassungen, Zusammenarbeit und kontinuierliche Verbesserung. Dabei versucht man die Arbeit in kleine Happen aufzuteilen, welche dafür sorgen sollen, dass das Teams regelmäßig funktionierende Ergebnisse liefern und das Feedback besser einarbeiten können. **Also kurz gesagt:** Man macht keine festen Pläne auf lange Zeit, sondern setzt

sich mehrere kleine Ziele auf dem Weg zur Fertigstellung des Projektes. (vgl. [3]).

- **Scrum:**

Scrum ist eine Methode bzw. ein Framework, welches die Philosophie des agilen Arbeitens umsetzt. Mithilfe von Scrum können Teams ihre Arbeit aufteilen und strukturieren. Scrum wird meist von Softwareentwicklungsteams genutzt, kann allerdings auch von allen möglichen Projekten angewandt werden, um bestmöglich ein Ziel zu erreichen. Dafür gibt es viele Meetings wie z.B. das Sprint Planning, hier werden Aufgaben festgelegt, das Daily Scrum, ein tägliches Meeting, das Sprint Review, in dem Ergebnisse vorgestellt werden und das Sprint Retro, hier werden mögliche Verbesserungen identifiziert und besprochen. (vgl. [4]).

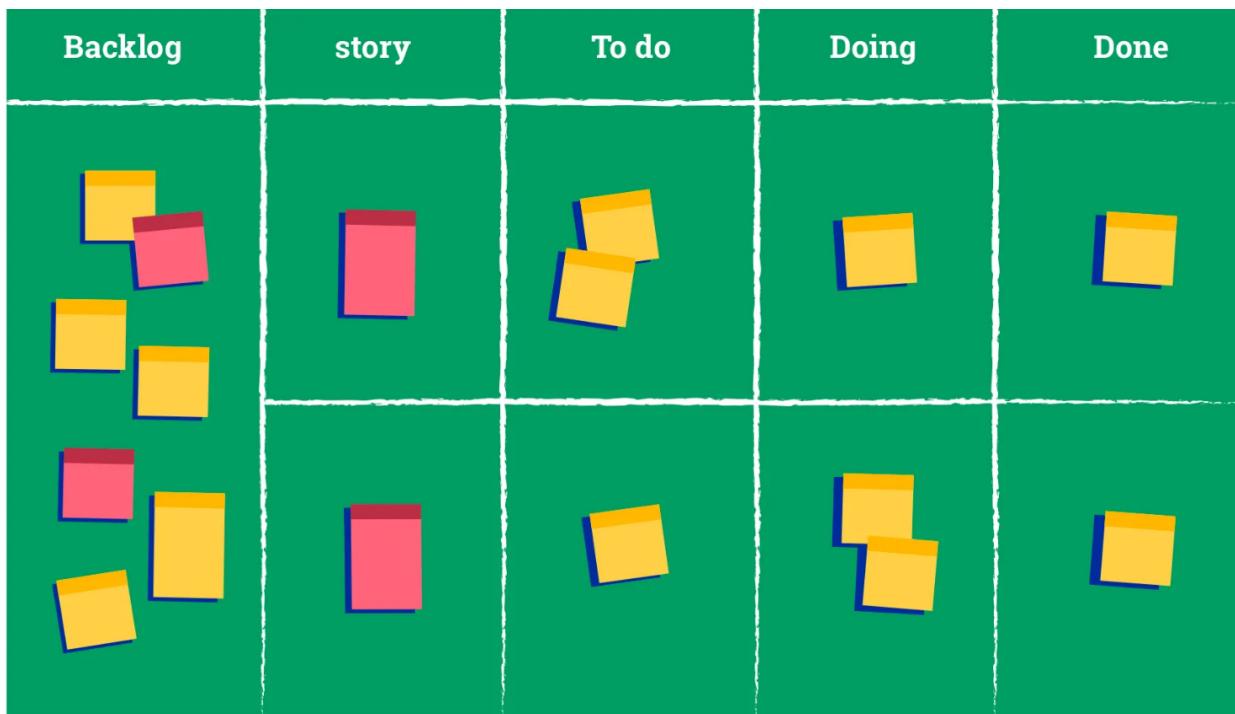


Abbildung 10. Praktische Umsetzung eines Scrum Boards (vgl. [5]).

In diesem Bild sieht man ein Scrum Board, aufgeteilt in Backlog, Story, to do, doing und done.

**Im Backlog** stehen alle anstehenden Aufgaben, welche noch nicht im aktuellen Sprint eingeplant sind.

**Story:** Diese User stories sind für den aktuellen Sprint angesetzt worden.

**To Do:** Alle geplanten, aber noch nicht begonnenen Aufgaben, des Sprints. Falls eine Aufgabe erledigt wurde, wählt man hieraus eine neue.

**Doing:** Aufgaben an welchen gerade gearbeitet wird. Hier wird gezeigt woran das Team gerade arbeitet.

**Done:** Sobald eine Aufgabe erledigt wurde, verschiebt man sie hierhin.,

### 2.1.3 Alternative Kanban

Kanban ist eine alternative zu Scrum und benutzt ebenfalls agiles Arbeiten. Mit Kanban wird der Fokus allerdings mehr auf einen kontinuierlichen Arbeitsfluss gelegt und weniger auf feste Zyklen (z.B. Scrum-Sprints). Dabei gibt es wie bei Scrum auch ein Board, allerdings können sich die Teammitglieder jederzeit eine neue Aufgabe zuweisen, da es eben keine festen Zeitzyklen gibt. (vgl.



[6]). So sieht ein Kanban-Board live, während der Entwicklung eines Projektes aus:

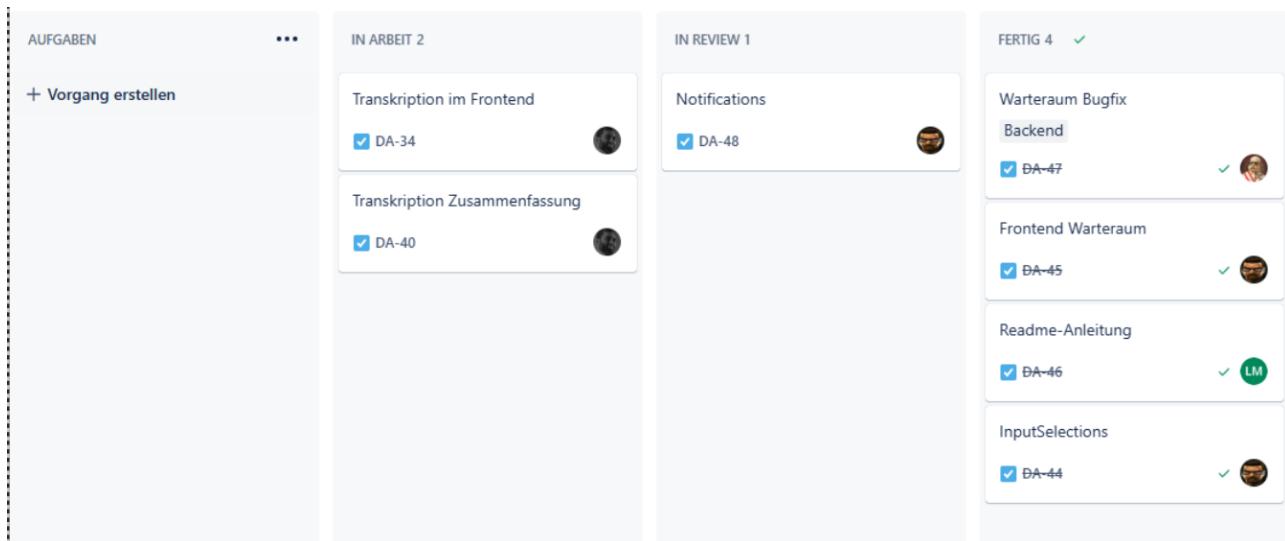


Abbildung 11. Kanban Board am Ende eines Projektes

## 2.1.4 relevante Tools

### GitLab CI/CD:

Während die allermeisten CI-Plattformen zusätzliche Tools benötigen, um eine vollständige CI/CD-Funktion bereitstellen zu können, vereint GitLab alles miteinander. Viele Unternehmen müssen oft komplexe und teure Toolchains verwalten, welche verschiedene Systeme wie SCM (z.B. Github), Test, Bereitstellungs und Sicherheitstools miteinander verbinden. Der Vorteil von GitLab ist, dass der gesamte DevSecOps-Lebenszyklus (Development, Security, Operation) in einer Anwendung ist. (vgl. [7]).

### Docker:

Docker ist eine Open-Source-Plattform, welche für die Containerisierung (vgl. [8]) verwendet wird. Mithilfe von Docker kann man Anwendungen und Abhängigkeiten in isolierte Container packen und diese somit bündeln. Für die Anwendung von CI/CD Pipelines wie GitLab CI/CD spielt Docker eine wichtige Rolle: Durch Docker können standardisierte, reproduzierbare Tests und Builds, beschleunigte Prozesse durch schnelle Container-Starts und minimierte Umgebungsprobleme, ermöglicht werden. Die Effizienz wird durch parallele Testausführungen und direkte Deployments gesteigert werden. (vgl. [9])

## 2.2 Serverless Computing und AWS Lambda:

### 2.2.1 Definition

Hierbei handelt es sich um ein Cloud-Computing-Modell, bei welchem der Cloud-Anbieter die Serverinfrastruktur dynamisch verwaltet und Ressourcen nach Bedarf bereitstellt, der Nutzer muss dabei weder den physischen Server noch die Wartung kontrollieren. Man zahlt genau so viel wie man auch verwendet. Der Amazon Web Service (AWS) bietet einer der beliebtesten Plattformen für Serverless Computing an, namens AWS Lambda. Mithilfe von AWS Lambda können Entwickler Code in Form von Funktionen ausführen, welche dann automatisch auf Ereignisse reagieren z.B.



HTTP-Anfragen, dabei muss der User den Server nicht manuell verwalten. Durch die automatische Skalierung, je nach Workload, kann die Effizienz und Flexibilität erhöht werden. Der Zusammenhang zu Serverless Applications besteht darin, dass diese genau dieses Prinzip, vom Serverless Computing verwenden. Dies setzen sie unter anderem dadurch um, dass sie oben genannten Lambda-Funktionen mit anderen Cloud-Diensten kombinieren und dadurch vollständige Anwendungen ohne Server zu erstellen. (vgl. [10])

### 2.2.2 Vor und Nachteile von serverlosen Architekturen

Bisher wurden in dieser Arbeit einige Vorteile von serverlosen Architekturen genannt, hier werden nocheinmal alle aufgezählt:

#### Vorteile

1. **Keine Serververwaltung** - Der Cloud-Anbieter (in unserem Fall AWS Lambda) übernimmt die gesamte Infrastruktur und das Entwicklerteam kann sich voll und ganz auf den Code konzentrieren.
2. **Automatische Skalierung** - Je nach Nachfrage skalieren serverlose Architekturen automatisch. Bei AWS Lambda z.B. wird die Kapazität dynamisch an die Anzahl der Anfragen angepasst, bei diesem Schritt braucht es keine manuellen Eingriffe. Dadurch spart sich das Entwicklerteam viel Energie und Zeit.
3. **Schnellere Entwicklung** - Dadurch, dass sich der Cloud-Anbieter um die gesamte Infrastruktur kümmert, konzentriert sich das Team nur auf den Code und kann diesen schneller bereitstellen und Projekte somit schneller produzieren.
4. **Kosten** - Man zahlt nur für die auch tatsächliche Nutzung (z.B. Ausführungszeit). Dies kann vor allem für Seiten, welche eher unregelmäßigen "Traffic" haben von Vorteil sein.

#### Nachteile

Es gibt allerdings auch einige Nachteile, welche hier auch genannt werden sollen:

1. **Cold Start-Problematik** - Falls eine Funktion einmal über längere Zeit nicht genutzt wurde, kann diese bei einem neuen Aufruf für Verzögerungen führen, da sie neu initialisiert werden muss. Bei Anwendungen, in welchen die Latenz wichtig ist, kann dies störend sein.
2. **Vendor Lock-in** - Durch die Nutzung von serverlosen Diensten bindet sich das Entwicklerteam an diesen Anbieter und der Code & die Architektur sind meistens auf genau diese APIs und Dienste zugeschnitten. Daher kann der Wechsel zu einem anderen Anbieter für Probleme sorgen und sehr aufwendig sein.
3. **Eingeschränkte Kontrolle** - Das Team hinter einem Projekt hat weniger Einfluss auf die Infrastruktur, was die Performance Optimierung und das Debugging erschwert und ein deutlicher Nachteil sein kann.

### 2.2.3 Überblick über AWS-Serverless-Dienste

#### 1. AWS Lambda:

Amazon Web Services stellte AWS Lambda erstmalig Ende 2014 vor und setze somit einen



großen Schritt in die Richtung des serverless Computing. Mittlerweile ist Lambda einer der größten Marktteilnehmer in diesem Bereich und ermöglicht Millionen von Usern ihre Projekte ohne Serververwaltung zu betreiben. Als eine der ersten Dienste in seinem Bereich hat AWS Lambda eine große Veränderung in der Branche hervorgerufen und unter anderem dafür gesorgt, dass Cloud-Anwendungen mittlerweile so einen wichtigen Stellenwert haben.

#### 2. Amazon API Gateway:

Dieser Dienst wird dafür verwendet um APIs zu erstellen, zu veröffentlichen und zu verwalten, welche mit anderen AWS-Diensten verbunden werden können und ergänzt somit AWS Lambda. Ebenfalls kann man damit RESTful- und WebSockets-APIs bereitstellen.

#### 3. Amazon S3:

Dies wird als objektbasierter Speicher genutzt, welcher ebenfalls keine Serververwaltung erfordert, dieser Speicher kann genutzt werden um ereignisgesteuerte Workflows zu ermöglichen. Ein Beispiel wäre ein Date-Upload in S3, welcher wiederum eine bestimmte Lambda-Funktion triggern würde, welche wieder irgendetwas ausführt.

#### 4. Amazon DynamoDB:

DynamoDB ist eine serverlose NoSQL-Datenbank und bildet einen Schlüsselbaustein für die Datenspeicherung mit AWS Lambda. Die Datenbank ist gut geeignet für dynamische Workloads, da sie automatisch skaliert und niedrige Latenz bietet.

## 3. Best Practices für CI/CD-Pipelines in AWS mit GitLab CI

### 3.1 Technische Integration und Automatisierung

#### 3.1.1 Integration mit AWS-Diensten optimieren

Um eine effiziente CI/CD Pipeline erstellen zu können ist eine nahtlose Integration von GitLab CI mit AWS-Diensten dringend notwendig. Vor allem wenn man AWS Lambda als Hauptkomponenten nutzt sollte die Pipeline so konfiguriert sein, dass diese ohne Probleme mit anderen AWS-Diensten interagieren kann. Dies kann man bewerkstelligen, indem man die AWS Command Line (CLI) oder das AWS Software Development Kit einer bestimmten GitLab-Konfigurationsdatei einbindet. (vgl. [11])

```
.gitlab-ci.yml
```

Ein Beispiel um dies umzusetzen wäre mithilfe von der Definition von einem Job, dieser muss Lambda-Funktionen automatisch aktualisieren oder bereitstellen. Dies kann umgesetzt werden, indem die AWS CLI mit bestimmten Befehlen aufgerufen wird, sobald ein neuer Commit vorliegt. Ein Beispiel (vgl. [12]) für so einen Befehl wäre:

```
aws lambda update-function-code
```



### 3.1.2 Automatisierung von Builds, Tests und Deployments

Dieser Schritt ist besonders wichtig in modernen CI/CD-Pipelines und dadurch auch ein Kernbestandteil von serverlosen Architekturen. Mithilfe von Automatisierung kann man, wie oben schon erwähnt, die Entwicklungszeit verkürzen, die Zuverlässigkeit erhöhen und manuelle Eingriffe vermeiden. Innerhalb einer CI/CD Pipeline kann dies durch definierte Jobs (vgl. [\[Job\]](#)) umgesetzt werden.

#### Build-Prozess:

Hier gibt es ein Beispiel für einen Job, welcher für die Bereitstellung verantwortlich ist. Dieser wird in der .gitlab-ci.yml-Datei umgesetzt und könnte so aussehen:

```
build_lambda:  
  stage: build  
  script:  
    - npm install  
    - zip -r lambda.zip .  
  artifacts:  
    paths:  
      - lambda.zip
```

Der Vorteil von so einem Job ist, dass er bei jedem Commit ausgeführt wird und der Code dadurch jederzeit für Tests oder das Deployment bereit ist. 1. build\_lambda ist der "Name" des Jobs. 2. Hier wird angegeben zu welcher Phase der Job gehört, indem Fall ist es ein Build-Prozess deswegen "build" 3. einfach ein Schlüsselwort, alles nach "script" wird in einer Shell-Umgebung (also CLI) ausgeführt 4. npm install ist ein Befehl, welcher die Abhängigkeiten eines Projektes installiert. 5. Dieser Befehl erstellt ein ZIP-Archiv namens lambda.zip 6. "artifacts" ist das nächste Schlüsselwort. Es definiert welche Dateien als "Artefakte" gespeichert werden, sobald der Job abgeschlossen wurde. Ohne diese "Artefakte" würde das Ergebnis des Jobs verloren gehen. In den Zeilen 7 & 8 wird definiert, welche Datei als "Artefakt" gespeichert werden soll, in diesem Fall ist es nur die vorher erstellte lambda.zip Datei.

#### Test-Prozess:

Im nächsten Schritt kümmert man sich darum alle möglichen Tests, wie Unit & Integrationstest zu automatisieren. Dies muss vor dem Deployment gemacht werden, um die Funktionalität der Lambda-Funktion zu validieren. Dafür kann auch ein separater Job geschrieben werden, welcher wie folgt aussehen könnte:

```
test_lambda:  
  stage: test  
  script:  
    - npm test  
    - aws lambda invoke --function-name PlatzhalterFürEinenFunktionsNam  
  output.json  
  dependencies:
```



- build\_lambda

In den ersten zwei Schritten wird der Job definiert und als test vermerkt. Daraufhin werden mit dem Befehl "npm test" alle unter dem Skript "test" definierten Tests ausgeführt, welche in der Datei "package.json" zu finden sind. Hier ein Beispiel:

```
package.json: "test": "jest"
```

Dieser Befehl in Zeile 5 muss unterteilt werden, um verständlich zu sein. Der 1. relevante Abschnitt ist :

```
aws lambda invoke
```

Das ist ein AWS-CLI-Befehl, welcher eine bestimmte Lambda-Funktion ausführt. Daraufhin folgt:

```
--function-name PlatzhalterFürEinenFunktionsNam
```

Hier gibt man den Namen der Funktion an, welche getestet werden soll, sie muss logischerweise bereits existieren. Und das letzte Stück des Befehls lautet:

```
output.json
```

Hier gibt man den Dateinamen an, in welchen der Output der Lambda-Funktion geschrieben werden soll. "dependencies" in Zeile 6 ist ein weiters Schlüsselwort, dieses definiert von welchen anderen Jobs der Job "test\_lambda" abhängig ist. In der letzten Zeile, welcher der 6. Zeile unterliegt, steht der Name des Jobs, von dem "test\_lambda" abhängt, in diesem Fall ist es "build\_lambda" unserem Build-Job.

## Deployment-Prozess

Nachdem die Tests erfolgreich durchgelaufen sind, kann ein Deployment-Job definiert werden.

```
deploy_lambda:  
  stage: deploy  
  script:  
    - aws lambda update-function-code --function-name MeineLambdaFunktion --zip  
      -file file:///lambda.zip  
  environment:  
    name: production  
  dependencies:  
    - build_lambda  
  only:
```



- main

Der Job wird "deploy\_lamda" genannt und in die "deploy" Phase eingeteilt. Dieser lange Befehl, welcher sich im "script" Abschnitt befindet muss ebenfalls unterteilt werden.

```
aws lambda update-function-code
```

Das ist ein AWS-CLI-Befehl, dieser ersetzt den Code einer bestehenden Lambda-Funktion.

```
--function-name MeineLambdaFunktion
```

Hier wird der Name von der Funktion angegeben, welche aktualisiert werden soll. Sie muss ebenfalls bereits existieren. Und im letzten Teil:

```
--zip-file fileb://lambda.zip
```

Wird der Pfad der Datei angegeben, welche den neuen Code enthält. In diesem Fall ist es jene Datei, welche zuvor im "build\_lambda-Job" erstellt wurde. In Zeile 5 steht abermals ein Schlüsselwort, dieses definiert in welcher Umgebung der Job deployed wird. Dies dient dazu um verfolgen zu können, wohin Code ausgerollt wird. In diesem Fall ist es "production". Daraufhin folgt wieder das Schlüsselwort "dependencies" in diesem Fall wird festgelegt, dass "deploy\_lamda" abhängig vom Job "build\_lamda" ist. In der Zeile danach folgt ein neues Schlüsselwort und zwar "only". In diesem Fall wird definiert, dass der Job nur ausgeführt werden soll, wenn Änderungen in den main Branch gepusht werden.

## 3.2 Sicherheit und Infrastruktur

### 3.2.1 Sicherheit und Zugriffskontrolle gewährleisten

Sicherheit ist wichtig um die Zugriffe auf Daten und Cloud-Ressourcen schützen zu können. Einen ersten Schritt, den man einleiten kann, wäre die Erstellung von (vgl. [13]) (Identity and Access Management) mit so wenig Berechtigungen wie möglich und sie sollten nur die notwendigen Funktionen erlauben wie z.B. das Update von Code. Die Zuweisung kann man durch AWS-Credentials (vgl. [14]) durchführen, diese werden sicher in GitLab CI/CD Variablen gespeichert. Ein Beispiel für so eine Variable wäre : "AWS\_ACCESS\_KEY\_ID", dies wird auch als Umgebungsvariable bezeichnet und dient dazu sich bei den AWS-Diensten authentifizieren zu können. Sie ist so verschlüsselt, dass sie nur innerhalb der Pipeline einsehbar ist, um unbefugten Zugriff zu vermeiden.

Man kann auch einen Test-Job erstellen, welcher die Authentifizierung durchführt. Dieser könnte wie folgt aussehen:

```
test_api:
```



```
stage: test
script:
  - curl -H "Authorization: Bearer $API_TOKEN" $API_GATEWAY_URL
```

Die ersten zwei Zeilen definieren den Namen des Jobs als "test\_api" und ordnen ihn zu "test-stage" hinzu. Daraufhin wird ein script ausgeführt, welches den übergebenen Token überprüft.

1. **curl**, ist ein Befehl, mit dem HTTP-Anfragen an Server gesendet.

2. **-H "Authorization: Bearer \$API\_TOKEN"**

Hier wird der Anfrage ein "header" hinzugefügt

3. **\$API\_GATEWAY\_URL**

Ist eine Umgebungsvariable, welche die gebrauchte URL enthält.

### 3.2.2 Infrastruktur als Code einsetzen

Infrastruktur als Code (kurz: IaC, engl. Infrastructure as Code) ermöglicht es Infrastruktur mithilfe von Code zu verwalten und bereitzustellen, anstatt von manuellen Prozessen. Dafür müssen Konfigurationsdateien erstellt werden, diese müssen alle Infrastrukturspezifikation enthalten. Dies vereinfacht die Bearbeitung, Verteilung und einheitliche Provisionierung von Umgebungen, unterstützt das Konfigurationsmanagement und verhindert undokumentierte Änderungen. Ein wichtiger Bestandteil ist die Versionskontrolle, dadurch das Konfigurationsdateien wie Quellcode verwaltet werden, wird die Automatisierung und die Modularität gefördert.

Es gibt zwei verschiedene Methoden um IaC zu verwirklichen: **Deklarative:** Hierbei wird der gewünschte Endzustand der Infrastruktur definiert. Man gibt an welche AWS-Ressourcen benötigt werden und wie diese konfiguriert sein sollen, darauf in übernimmt ein IaC-Tool die Umsetzung. Dabei vergleicht das Tool den aktuellen Zustand mit dem gewünscht und nimmt alle Änderungen vor. Mithilfe des Tools wird die Verwaltung leichter, da es eine Übersicht schafft und unter anderem auch beim Abbau der Infrastruktur hilft. Durch die eigenständige Erkennung von Änderungen und Anwendung von Lambda-Triggers wird die Nutzung und Wartung der GitLab CI-Pipeline um einiges effizienter.

**Imperative:** In dieser Methode werden die genauen Befehle spezifiziert. Bei dieser Methodik müssen die Schritte in der richtigen Reihenfolge angegeben um eine funktionierende Infrastruktur aufzubauen. Die Komplexität und Fehleranfälligkeit ist bei dieser Methode allerdings um einiges höher, da man selber festlegen muss, welche Änderungen wie implementiert werden müssen. (vgl. [15]).

## 3.3 Qualitätssicherung und Fehlerbehebung

Das Thema der Qualitätssicherung und Fehlerbehebung ist abermals ein sehr dringendes. Ohne diesem wäre die Stabilität und Zuverlässigkeit von serverlosen Anwendungen mit AWS Lambda in CI/CD Pipelines nicht garantiert. Mithilfe dieser Best Practices wird sichergestellt, dass Fehler rechtzeitig erkannt und korrigiert werden. Die Grundlage dafür wird folgendermaßen gebildet:



### 3.3.1 Monitoring und Logging integrieren

Bei der Nutzung von serverlosen Architekturen haben Entwickler keine direkte Kontrolle über die Infrastruktur, daher ist die Überwachung besonders wichtig. Dabei hilft AWS CloudWatch, es hilft einem dabei Lambda-Ausführungen in Echtzeit zu überwachen. Dies ist abermals mit einem Job umsetzbar, diese könnte wie folgt aussehen:

```
monitor_lambda:  
  stage: monitor  
  script:  
    - aws logs filter-log-events --log-group-name /aws/lambda/MeineLambdaFunktion  
      --filter-pattern "ERROR" > errors.json  
    - if [ -s errors.json ]; then exit 1; fi
```

(vgl. [16]).

Hier wird ein Job namens "monitor\_lambda", welcher der "monitor-stage" angehört, definiert. Der Befehl nach "script" ist etwas komplizierter und muss daher unterteilt werden:

#### 1.Zeile:

1. "aws" ist der Befehl, welcher die AWS CLI aufruft.
2. "logs filter-log-events", dieser Befehl wird speziell für CloudWatch Logs verwendet und filtert die Events basierend auf den Kriterien.
3. "--log-group-name /aws/lambda/MeineLambdaFunktion", hier wird die Log-Gruppe angegeben, aus welcher die Logs aufgerufen werden sollen.
4. "--filter-pattern "ERROR", hier wird definiert wonach die Logs gefiltert werden sollen. In diesem Fall werden alle Log-Einträge nach dem Wort "ERROR" gefiltert.
5. "> errors.json", dabei wird gesorgt, dass die Ausgabe des Befehls in eine Datei geschrieben wird, hier wäre es error.json.

#### 2.Zeile:

1. "if", ist eine Anweisung innerhalb der Bash (CLI), diese prüft ob eine Bedingung erfüllt ist, daraufhin wird eine Aktion ausgeführt. (Wenn das und das stimmt, passiert das und das).
2. "[ -s error.json]", das ist ein Test-Befehl, er prüft ob die Datei überhaupt existiert.
3. "then exit 1", erst dann, wenn das "if" zutrifft, wird dieser Befehl ausgeführt. exit 1 sorgt dafür, dass das Skript mit einem Fehler beendet wird, darauhin markiert die Pipeline den Job als fehlerhaft.
4. "fi", schließt die "if" Anweisung ab.

### 3.3.2 Fehlerbehandlung und Rollback-Strategien

Für den Fall das Fehler bei der Bereitstellung oder Ausführung von Lambda-Funktionen auftreten, müssen starke Fehlerbehandlungspläne vorhanden sein um den Schaden, kleinstmöglich zu halten und die Anwendung zu schützen. Dafür könnte man z.B. automatische Rollbacks konfigurieren.

```
deploy_lambda:
```



```
stage: deploy
script:
  - aws lambda update-function-code --function-name MeineLambdaFunktion --zip
    -file fileb://lambda.zip
rollback_lambda:
  stage: deploy
  script:
    - aws lambda update-function-code --function-name MeineLambdaFunktion
    --function-version PREVIOUS
  when: on_failure
```

(vgl. [17]). Hier werden 2 Jobs erstellt, welche beide der "deploy-stage" angehören.

1. deploy\_lambda: Diesem wird ein Befehl übergeben, welcher die gegebene Lambda-Funktion (in diesem Fall "MeineLambdaFunktion"), aktualisiert, dies wird umgesetzt indem der Code innerhalb der "lambda.zip" Datei hochgeladen wird.
2. rollback\_lambda: Diesem Job wird ein Befehl übergeben, dieser sorgt dafür, dass die vorherige Version hochgeladen wird. In diesem Beispiel steht der Platzhalter "PREVIOUS", in einem echten Anwendungsbeispiel müsste dort die gewollte Datei stehen, welche vom Job eingesetzt werden soll.
3. when: on\_failure: Sorgt dafür, dass "rollback\_lambda" nur ausgeführt wird, wenn einer der vorherigen Jobs (in diesem Fall "deploy\_lambda") fehlschlägt.

## 4. Anwendungsfall: Videomeeting-Lösungen

### 4.1 Ausgangslage und Problemstellung

Die Firma Naos benötigte eine sichere Videomeeting-Lösung, da sie keine geeignete Meetinglösung für Kunden haben. Die bestehenden Tools sind unzureichend, vor allem bezogen auf Datenschutz, ebenfalls fehlte der Firma die Möglichkeit auf Aufzeichnungen. (vgl. [18]).

### 4.2 Zielsetzung des Diplomprojekts

Das Ziel war es eine gebrandete 1:1-Videomeeting-Anwendung mit AWS Chime, Next.js/React, Tailwind, AWS Lambda, Aurora Serverless und Elastic Beanstalk zu entwickeln. Unsere Anwendung bietet feste Meetings-URLs, Warteräume, SuperTokens-Authentifizierung, Videokonferenzen, die automatische Aufzeichnung, Transkription und Zusammenfassung mit OpenAI. (vgl. [18]).

### 4.3 Automatisierte Bereitstellung mit CI/CD Pipeline

Ein wichtiger Punkt der Themenstellung ist die automatisierte Bereitstellung durch eine CI/CD Pipeline, in Kapitel 2.1.4 steht einiges an Theorie dazu. Aber wie schaut so eine Pipeline, welche mit den Best Practices aus Kapitel 3 umgesetzt wurde, in der Praxis aus. Diese wurde wie in Kapitel 1.1 erklärt in drei definierte Phasen aufgeteilt.



## Build-Stage

Hier wird das Frontend, bestehend aus Next.js/React-Code für das Frontend gebaut. In dieser Phase werden auch die Lambda-Funktionen vorbereitet.

```
build_job:  
  stage: build  
  image: node:18 # Node.js-Image für Next.js  
  script:  
    # Hier wird das Frontend gebaut  
    - cd ioe-video-call/src # Verzeichnis mit dem Next.js-Code  
    - npm install           # Hier installiert man alle Abhängigkeiten  
    - npm run build         # Next.js build  
    # Lambda-Funktion vorbereiten  
    - cd ../lambda          # Hier gibt man das Verzeichnis seiner Lambda-Funktionen  
  artifacts:  
    paths:  
      - src/.next/           # Frontend-Artefakt  
      - lambda/lambda.zip   # Lambda-Artefakt  
  expire_in: 1 hour        # Artefakte für spätere Jobs speichern
```

(vgl. [19]).

## Test-Stage

Wir wollen die Qualitätssicherung, welche in Kapitel 3.3 beschrieben wird, sicherstellen, daher müssen Unit-Tests genauso wie Integrationstest durchgeführt werden. Hier wurde mithilfe von test\_chime\_auth ein Meeting erstellt und die SuperTokens-Authentifizierung getestet.

```
test_chime_auth:  
  stage: test  
  image: amazon/aws-cli:latest # AWS CLI für Chime-Aufrufe  
  variables:  
    AWS_REGION: "us-east-1"  
  script:  
    # AWS CLI konfigurieren  
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID  
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY  
    - aws configure set region $AWS_REGION  
    # Chime-Meeting erstellen  
    - MEETING_RESPONSE=$(aws chime create-meeting --title "TestMeetingNaos"  
    --region $AWS_REGION --query 'Meeting.MeetingId' --output text)  
    - echo "Meeting ID: $MEETING_RESPONSE"  
    # Authentifizierung mit SuperTokens testen (simulierte API-Anfrage)  
    - curl -H "Authorization: Bearer $SUPERTOKENS_TOKEN" $MEETING_API_URL --fail
```



## dependencies:

- build\_job # Benötigt die Artefakte aus dem Build-Job

(vgl. [20]).

### Deployment-Stage

AWS Elastic Beanstalk stellt das Frontend für unser Projekt bereit und aktualisiert alle Lambda-Funktionen.

```
deploy_job:  
  stage: deploy  
  image: amazon/aws-cli:latest  
  variables:  
    AWS_REGION: "us-east-1"  
  script:  
    # AWS CLI konfigurieren  
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID  
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY  
    - aws configure set region $AWS_REGION  
    # Frontend auf Elastic Beanstalk bereitstellen  
    - cd ioe-video-call/src  
    - zip -r application.zip .next/      # Frontend-Artefakte in ZIP packen  
    - aws s3 cp application.zip s3://naos-meeting-app-bucket/ # ZIP in S3 hochladen  
    - aws elasticbeanstalk update-environment --environment-name NaosAppEnv  
--version-label $CI_COMMIT_SHA --application-name NaosApp  
    # Lambda-Funktion aktualisieren  
    - cd ../lambda  
    - aws lambda update-function-code --function-name TranskriptionFunktion --zip  
-file file:///lambda.zip  
  environment:  
    name: production  
  dependencies:  
    - build_job # Benötigt die Artefakte aus dem Build-Job  
  only:  
    - main # Nur auf main-Branch ausführen
```

(vgl. [21]).

## 4.4 Anwendung der Best Practices

Einige der in Kapitel 3 erklärten Best Practices wurden auf unsere Videomeeting-Anwendung angewendet. Die Jobs, welcher für die Umsetzung dieser Best Practices erstellt wurden, dienen dazu die Anforderungen der Partnerfirma Naos zu unterstützen.



#### 4.1 Integration mithilfe von AWS-Diensten optimieren (3.1.1)

```
test_aurora_integration:
  stage: test
  image: amazon/aws-cli:latest
  variables:
    AWS_REGION: "eu"
  script:
    # AWS CLI konfigurieren
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY
    - aws configure set region $AWS_REGION
    # Testabfrage an Aurora Serverless
    - aws rds-data execute-statement --resource-arn $AURORA_ARN --database
      naos_meetings --sql "INSERT INTO meetings (meeting_id, title) VALUES ('test123',
      'TestMeeting')"
    - aws rds-data execute-statement --resource-arn $AURORA_ARN --database
      naos_meetings --sql "SELECT * FROM meetings WHERE meeting_id = 'test123'" --query
      'records' --output text
  dependencies:
    - build_job
```

(vgl. [22]). Mithilfe von diesem Job kann überprüft werden, ob die Metadaten richtig und fehlerlos gespeichert werden.

#### 4.2 Infrastruktur als Code (IaC, 3.2.2)

```
deploy_infrastructure:
  stage: deploy
  image: amazon/aws-cli:latest
  variables:
    AWS_REGION: "eu"
  script:
    # AWS CLI konfigurieren
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY
    - aws configure set region $AWS_REGION
    # Infrastruktur mit CloudFormation bereitstellen
    - aws cloudformation deploy --template-file naos-infra.yml --stack-name
      NaosAppStack --capabilities CAPABILITY_IAM --region $AWS_REGION
  only:
    - main
```

(vgl. [23]).

Durch diesen Job wird ein sicheres und konsistentes environment (Umgebung) sichergestellt. Das



passiert indem der Job die Infrastruktur wie z.B. Aurora Serverless und Lambda bereitstellt.

#### 4.3 Monitoring und Logging integrieren (3.3.1)

```
monitor_lambda_logs:
  stage: monitor
  image: amazon/aws-cli:latest
  variables:
    AWS_REGION: "eu"
  script:
    # AWS CLI konfigurieren
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY
    - aws configure set region $AWS_REGION
    # CloudWatch-Logs auf Fehler prüfen
    - aws logs filter-log-events --log-group-name /aws/lambda/TranskriptionFunktion
      --filter-pattern "ERROR" --region $AWS_REGION > errors.json
      - if [ -s errors.json ]; then echo "Fehler gefunden in Logs"; exit 1; fi
  dependencies:
    - deploy_job
```

(vgl. [24]). Hier werden die Lambda-Funktionen, welche für die Transkription benötigt werden überwacht und überprüft die CloudWatch-Logs auf Fehler, dadurch können potenzielle Probleme rechtzeitig erkannt und behoben werden.

#### 4.4 Fehlerbehandlung und Rollback (3.3.2)

```
rollback_beanstalk:
  stage: deploy
  image: amazon/aws-cli:latest
  variables:
    AWS_REGION: "eu"
  script:
    # AWS CLI konfigurieren
    - aws configure set aws_access_key_id $AWS_ACCESS_KEY_ID
    - aws configure set aws_secret_access_key $AWS_SECRET_ACCESS_KEY
    - aws configure set region $AWS_REGION
    # Rollback auf vorherige Version
    - aws elasticbeanstalk update-environment --environment-name NaosAppEnv
      --version-label $PREVIOUS_VERSION --region $AWS_REGION
  when: on_failure
  dependencies:
    - deploy_job
```

(vgl. [25]). Für den Fall, dass das Deployment fehlschlägt, wird durch diesen Job ein Rollback für



AWS Elastic Beanstalk erstellt.

## Fazit

Im Rahmen des Diplomprojekts für die Firma Naos wurde eine Videomeeting-Lösung entwickelt, die spezifische Anforderungen wie Datenschutz, feste Meeting-URLs, Aufzeichnungsfunktionen und Authentifizierung erfüllt. Durch die Implementierung einer CI/CD-Pipeline, die in Build-, Test- und Deployment-Phasen strukturiert ist, konnte die Bereitstellung automatisiert und beschleunigt werden. Best Practices wie die Optimierung der Integration mit AWS-Diensten, der Einsatz von Infrastruktur als Code (IaC) mittels CloudFormation, das Monitoring mit CloudWatch und Roll-back-Strategien bei Fehlern wurden erfolgreich angewendet. Diese Maßnahmen haben nicht nur die Zuverlässigkeit und Skalierbarkeit der Anwendung erhöht, sondern auch menschliche Fehler minimiert und eine kontinuierliche Anpassung an neue Anforderungen ermöglicht.

Die Arbeit verdeutlicht die Relevanz der vorgestellten Ansätze in einer Zeit, in der die Nachfrage nach skalierbaren und automatisierten Videokommunikationslösungen durch die Digitalisierung und die Corona-Pandemie gestiegen ist. Sie bietet einen praktischen Leitfaden für Entwickler, wie serverlose Architekturen mit AWS und GitLab CI kombiniert werden können, um effiziente und wartungsarme Anwendungen zu schaffen. Die erfolgreiche Umsetzung im Anwendungsfall zeigt, dass die Kombination aus serverless Computing und CI/CD eine leistungsstarke Methode ist, die es Entwicklern ermöglicht, sich auf die Code-Entwicklung zu konzentrieren, während die Infrastrukturverwaltung dem Cloud-Anbieter überlassen wird.

Zusammenfassend adressiert die Arbeit die Themenstellung umfassend, indem sie sowohl die technischen Grundlagen als auch deren praktische Anwendung beleuchtet. Die vorgestellten Best Practices sind nicht nur auf Videomeeting-Lösungen beschränkt, sondern können auch auf andere serverlose Anwendungen übertragen werden. Sie legt damit einen Grundstein für zukünftige Projekte und Weiterentwicklungen im Bereich der automatisierten Bereitstellung und Skalierung serverloser Architekturen (vgl. [26]).

## Literaturverzeichnis

- [1] Serverloses Computing. URL: [azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-serverless-computing/](https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-serverless-computing/) (abgerufen am 05.03.2025).
- [2] "Skalierbarkeit, Automatisierung von Deployments" Newman, S. (2015). Building Microservices (Buch).
- [3] Agile. URL: [www.atlassian.com/de/agile/scrum](https://www.atlassian.com/de/agile/scrum), bei "Agile vs. Scrum".
- [4] Scrum. URL: [www.atlassian.com/de/agile/scrum](https://www.atlassian.com/de/agile/scrum), bei "Was ist Scrum?".
- [5] ScrumBoard. URL: [www.zohowebstatic.com/sites/zweb/images/sprints/seo/sb-how-img2.webp](https://www.zohowebstatic.com/sites/zweb/images/sprints/seo/sb-how-img2.webp)
- [6] Kanban. URL: [www.atlassian.com/de/agile/kanban](https://www.atlassian.com/de/agile/kanban)
- [7] GitLab. URL: [about.gitlab.com/de-de/topics/ci-cd/](https://about.gitlab.com/de-de/topics/ci-cd/)



- [8] Containerization. URL: [aws.amazon.com/de/what-is/containerization/](https://aws.amazon.com/de/what-is/containerization/)
- [9] Docker. URL: [www.redhat.com/de/topics/containers/what-is-docker](https://www.redhat.com/de/topics/containers/what-is-docker)
- [10] Serverless Computing. URL: [docs.aws.amazon.com/lambda/](https://docs.aws.amazon.com/lambda/)
- [11] Configuration options. URL: [docs.gitlab.com/ci/yaml/](https://docs.gitlab.com/ci/yaml/)
- [12] Lambda-Optimisierung. URL: [docs.aws.amazon.com/sdk-for-java/latest/developer-guide/lambda-optimize-starttime.html](https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/lambda-optimize-starttime.html)
- [13] Rollenerstellung/verwaltung. URL: [aws.amazon.com/de/iam/details/manage-roles/](https://aws.amazon.com/de/iam/details/manage-roles/)
- [14] Credentials. URL: [aws.amazon.com/de/iam/details/manage-roles/](https://aws.amazon.com/de/iam/details/manage-roles/)
- [15] Infrastrucuter as Code, IaC. URL: [www.redhat.com/de/topics/automation/what-is-infrastructure-as-code-iac](https://www.redhat.com/de/topics/automation/what-is-infrastructure-as-code-iac)
- [16] MonitorJob. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [17] DeployJob. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [18] Ausgangslage/ Zielsetzung. URL: [aba.bildung.gv.at/themen/details/9e9130d0-3f93-4fea-9141-581361d26d6b](https://aba.bildung.gv.at/themen/details/9e9130d0-3f93-4fea-9141-581361d26d6b)
- [19] BuildJob. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [20] Test Chime Job.xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [21] Test Deploy Job. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [22] Test Aurora Job. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [23] Deploy Infrastructure Job. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [24] Monitor Logs Job. xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [25] Rollback Job.xAiGrok3 Prompt: "Erstell mir bitte die notwendigen Jobs, welche so auch in einer CI/CD-Pipeline funktionieren würden". URL: [grok.com/](https://grok.com/)
- [26] Fazit. xAiGrok3 Prompt: "kannst du mir zu dieser Arbeit ein Fazit schreiben, es soll bitte die Themenstellung aufgreifen" URL: [grok.com/](https://grok.com/)



# Themenstellung von Maximilian Schwarz

KI-gestützte Transkription und Analyse von Videomeetings: Untersuchung der Anwendung von OpenAI Whisper und GPT zur Verbesserung von Meeting-Aufzeichnungen, Zusammenfassungen und Handlungsempfehlungen.

## 1. Einleitung

### 1.1 Problemstellung

Durch die fortschreitende Digitalisierung der Berufswelt sind Videomeetings zu einem essentiellen Element der Kommunikation in Firmen und Institutionen geworden. Vor allem seit der globalen Pandemie 2020 hat die Nutzung von Plattformen wie Zoom, Microsoft Teams oder Webex stark zugenommen, was zu einem exponentiellen Anstieg der aufgezeichneten Besprechungen geführt hat. Obwohl diese Art der Kommunikation immer häufiger genutzt wird, bestehen dennoch große Schwierigkeiten bei der Nachbereitung und Auswertung solcher Besprechungen. Eine der größten Herausforderungen besteht darin, aus langwierigen Meetings wichtige Informationen und Beschlüsse effizient herauszufiltern. Manuelle Mitschriften benötigen viel Zeit und sind anfällig für Fehler. Herkömmliche Meeting-Aufzeichnungen hingegen sind oft unstrukturiert und schwer zu durchsuchen. Hier kommen KI-Technologien wie OpenAI Whisper und GPT ins Spiel, die vielversprechende Ansätze zur Automatisierung und Optimierung dieses Prozesses bieten.

Diese Arbeit befasst sich mit dem Potenzial von OpenAI Whisper und GPT zur Verbesserung der Meeting-Dokumentation, den daraus resultierenden Vorteilen für Unternehmen sowie den Einschränkungen, die diese Technologien derzeit noch aufweisen.

### 1.2 Zielsetzung

Es soll eine effiziente Lösung entwickelt werden, die Audiodaten aus Meetings automatisch transkribiert und analysiert, um prägnante Zusammenfassungen und umsetzbare Handlungsempfehlungen zu erstellen. Es wird dabei untersucht, wie OpenAI Whisper als Transkriptionswerkzeug und GPT als Analyse- und Generierungstool verwendet werden kann, um die Nachbearbeitung zu verbessern. Die Ergebnisse sollen dazu beitragen, Zeit zu sparen und die Qualität der Meeting-Dokumentation zu verbessern, indem redundante Informationen verringert und wesentliche Inhalte betont werden. Zusätzlich soll untersucht werden, wie effizient diese KI-gestützten Methoden sind und welche Vorteile sie bieten – mit Blick auf Datenschutz, Skalierbarkeit und Genauigkeit.



## 2. Grundlagen

### 2.1 Künstliche Intelligenz und Sprachverarbeitung

Die Entwicklung von Systemen, die in der Lage sind, Aufgaben zu bewältigen, die normalerweise menschliche Intelligenz erfordern, wie Problemlösung, Mustererkennung oder Sprachverarbeitung, fällt unter den Begriff der Künstlichen Intelligenz (KI). Ein wesentlicher Bereich der KI ist die natürliche Sprachverarbeitung (Natural Language Processing), die Maschinen befähigt, menschliche Sprache zu analysieren, zu verstehen und zu erzeugen. Diese Technologie wird in Bereichen wie automatischer Transkription, Textzusammenfassung und Generierung von Handlungsempfehlungen weit eingesetzt – Aspekte, die für die Dokumentation von Meetings besonders relevant sind.

Das auf neuronalen Netzwerken basierende Transformer-Modell [1] stellt einen Fortschritt in der Sprachverarbeitung dar. Es erfasst durch den Mechanismus „Attention“ kontextuelle Zusammenhänge in Texten effizient. Diese Architektur hat die Leistungsfähigkeit von KI-Systemen erheblich verbessert und dient als Grundlage für moderne Modelle wie OpenAI Whisper und GPT. Whisper ist ein ASR-System (Automatic Speech Recognition), das gesprochene Sprache in Text umsetzt und sich durch eine hohe Präzision auszeichnet, selbst bei unterschiedlichen Audioqualitäten. GPT hingegen spezialisiert sich auf die Erzeugung und Verarbeitung von Texten. Es kann zum Beispiel Transkripte zusammenfassen oder kontextbezogene Vorschläge machen.

Diese Technologien sind für Meeting-Aufzeichnungen von Bedeutung, da sie unstrukturierte Audiodaten in strukturierte, nutzbare Informationen umwandeln können. Whisper kann Videokonferenzen in Echtzeit transkribieren, während GPT die daraus resultierenden Transkripte analysiert und prägnante Protokolle oder Aktionspunkte erstellt. Einem Artikel auf Towards Data Science zufolge liegt die Genauigkeit aktueller Spracherkennungssysteme unter idealen Bedingungen bei über 95 %, wobei Herausforderungen wie Hintergrundgeräusche oder Dialekte weiterhin bestehen. Die Vereinigung dieser Technologien birgt allerdings ein großes Potenzial für eine Effizienzsteigerung in der Nachbearbeitung von Meetings.

### 2.2 Automatische Transkription: Prinzipien und Herausforderungen

Der Prozess der automatischen Transkription, auch bekannt als Automatic Speech Recognition (ASR), beinhaltet die Umwandlung gesprochener Sprache in geschriebenen Text mithilfe von Algorithmen. In den letzten Jahren hat sich diese Technologie erheblich weiterentwickelt, was auf Fortschritte im Bereich der Künstlichen Intelligenz zurückzuführen ist, insbesondere durch Deep Learning und neuronale Netzwerke. Trotzdem existieren Herausforderungen, die in unterschiedlichen Anwendungsszenarien Auswirkungen auf die Genauigkeit und Verlässlichkeit haben. Nachfolgend werden die wesentlichen Prinzipien der automatischen Transkription und die damit verbundenen Schwierigkeiten beschrieben. [2]



## 2.2.1 Prinzipien der automatischen Transkription

Die automatische Transkription funktioniert durch drei Hauptkomponenten:

Akustisches Modell: Dieses Modell untersucht Audiosignale und konvertiert sie in phonetische Einheiten. Es ist die Basis für das Erkennen der gesprochene Laute.+ Sprachmodell: Das Sprachmodell ermittelt die Wahrscheinlichkeit von Wortabfolgen und gewährleistet, dass die transkribierten Sätze grammatisch korrekt und sinnvoll sind.+ Dekodierungsalgorithmus: Ein Algorithmus wie der Viterbi-Algorithmus vereint die Resultate des akustischen und des Sprachmodells, um die wahrscheinlichste Textsequenz zu generieren.+

Moderne ASR-Systeme wie OpenAI Whisper verwenden Transformer-Modelle, die durch Selbst-aufmerksamkeit-Mechanismen kontextuelle Zusammenhänge in den Audiodaten identifizieren. Dies führt zu einer höheren Präzision, vor allem bei langen oder komplexen Sätzen. Zudem ist Whisper dafür konzipiert, eine große Anzahl von Sprachen und Dialekten zu verarbeiten. Dadurch eignet es sich für internationale Anwendungen wie Meetings besonders nützlich macht.

## 2.2.2 Herausforderungen der automatischen Transkription

Trotz der technologischen Entwicklungen haben ASR-Systeme mit verschiedenen Herausforderungen zu kämpfen, die ihre Leistungsfähigkeit beeinträchtigen können:

Hintergrundgeräusche und Audioqualität: Störgeräusche, minderwertige Mikrofone oder Sprecher, die sich überlappen, reduzieren oft die Erkennungsgenauigkeit. Auch fortschrittliche Systeme haben Schwierigkeiten, wenn die Audioqualität stark beeinträchtigt ist. Akzente und Dialekte: Wenn das Modell nicht auf diese Vielfalt trainiert wurde, können unterschiedliche Aussprachen oder regionale Varianten Fehler verursachen, insbesondere. Dies ist in globalen Zusammenhängen mit Sprechern aus unterschiedlichen Ländern von besonderer Bedeutung. In Feldern wie der Medizin oder Technik kommen spezielle Ausdrücke vor, die im allgemeinen Wortschatz nicht enthalten sind. Oft ist es notwendig, das Modell anzupassen. Sprecheridentifikation: Bei Meetings und ähnlichen Situationen mit mehreren Sprechern gestaltet sich die korrekte Identifizierung der einzelnen Sprecher sowie die entsprechende Zuordnung in der Transkription als herausfordernd, was unklare Protokolle zur Folge haben kann.

In der Praxis ist die Genauigkeit von ASR-Systemen unter suboptimalen Bedingungen oft geringer als 90 %. Dies verdeutlicht, dass die Technologie fortlaufend weiterentwickelt und auf spezifische Anforderungen zugeschnitten werden muss.

## 2.3 KI-gestützte Textanalyse und deren Anwendung

Der Begriff „KI-gestützte Textanalyse“ umfasst die Verwendung von Künstlicher Intelligenz (KI) zur Automatisierung der Verarbeitung, Analyse und Interpretation von Textdaten. Komplexe Muster, Zusammenhänge und Bedeutungen in Texten können durch Algorithmen und Modelle, die auf maschinellem Lernen und insbesondere Deep Learning basieren, erkannt werden. In den vergangenen Jahren hat diese Technologie bedeutende Fortschritte erzielt und findet Anwendung in vielen Bereichen, wie zum Beispiel im Marketing, im Kundenservice, in der Forschung oder zur Optimierung der Dokumentation von Videomeetings. Die grundlegenden Prinzipien und die bedeu-



tendsten Anwendungen dieser Technologie werden im Folgenden beschrieben.

### Basisprinzipien der KI-gestützten Textanalyse

Die Textanalyse mithilfe von KI gründet sich auf Modellen, die mit umfangreichen Daten trainiert werden, um menschliche Sprache zu begreifen und zu handhaben. Von besonderer Bedeutung sind dabei neuronale Netzwerke wie Transformer-Modelle, die durch ihre Fähigkeit zur Selbstattention kontextuelle Zusammenhänge in Texten identifizieren können. Diese Modelle werden zuerst auf großen Datensätzen vorgenommen und danach für spezifische Aufgaben optimiert.

Die grundlegenden Techniken umfassen:

- **Tokenisierung:**

Zergliederung von Text in kleinere Bestandteile. [3]

- **Embeddings:**

Transformation von Wörtern oder Sätzen in Vektoren, die semantische Beziehungen darstellen.

- **Klassifikation und Clustering:**

Einteilung oder Gruppierung von Texten anhand identifizierter Muster.

- **Generierung:**

Erzeugen neuer, zusammenhängender Texte gestützt auf Eingaben.

Mit diesen Methoden ist es möglich, unstrukturierte Textdaten in strukturierte und verwertbare Informationen zu transformieren. Verwendung der auf KI-gestützter Textanalyse basierenden Anwendungen

### 2.3.1 Anwendungen der KI-gestützten Textanalyse

Die KI-gestützte Textanalyse kann in vielen verschiedenen Bereichen genutzt werden. Die wichtigsten Anwendungen, die auch für Meeting-Dokumentationen von Bedeutung sind, finden Sie hier:

Analyse von Emotionen Mit dieser Technik wird die Stimmung oder Meinung in Texten wie Kundenbewertungen oder Meeting-Transkripten identifiziert. In Meetings kann sie verwendet werden, um die Einstellung der Teilnehmer zu bestimmten Themen zu untersuchen oder problematische Diskussionspunkte auszumachen. Textkategorisierung Texte werden automatisch kategorisiert, beispielsweise zur Erkennung von Spam oder zur Zuordnung von Meeting-Inhalten zu Agenda-Punkten. Das vereinfacht die Strukturierung von Protokollen. Named Entity Recognition (NER) NER erkennt und kategorisiert benannte Entitäten wie Personen, Orte oder Organisationen. In Meeting-Transkripten erleichtert dies die Extraktion wichtiger Akteure oder Referenzen, was bei der Nachverfolgung von Aufgaben hilfreich ist. Textabzug Kernpunkte werden aus langen Texten extrahiert. Dies spart insbesondere bei Videomeetings Zeit, da prägnante Zusammenfassungen aus umfangreichen Transkripten erstellt werden. Erstellung von Handlungsempfehlungen KI kann aus dem Inhalt von Besprechungen konkrete Aufgaben oder Folgeschritte ableiten. Dadurch wird die Protokollierung automatisiert und die Umsetzung der Ergebnisse unterstützt.



Darüber hinaus findet KI-gestützte Textanalyse Anwendung in Chatbots und virtuellen Assistenten, die in der Lage sind, natürliche Sprache zu verstehen und darauf zu reagieren. Dies führt zu einer Verbesserung des Kundenservices.

## 3. Stand der Technik

### 3.1 Aktuelle Methoden zur Transkription von Videomeetings

In der modernen Arbeitswelt ist die Transkription von Videomeetings äußerst wichtig, da sie die Nachverfolgung von Besprechungen, die Dokumentation von Entscheidungen und die Zugänglichkeit von Inhalten ermöglicht. Die Genauigkeit, Effizienz und Kosten unterscheiden sich zwischen den verschiedenen Methoden zur Transkription. Im Folgenden erfolgt eine Beschreibung der am weitesten verbreiteten Methoden, die derzeit in der Praxis angewandt werden.

#### 3.1.1 Manuelle Transkription

Human Transkribierende wandeln bei der manuellen Transkription gesprochene Sprache in Text um. Mit dieser Methode ist die Präzision am höchsten, weil Menschen den Kontext, Nuancen, Fachtermini und sogar überlappende Sprecher oder Hintergrundgeräusche besser erfassen können. Allerdings benötigt sie viel Zeit und Geld: Für die Transkription einer Stunde Audio braucht ein Transkribent im Schnitt vier bis sechs Stunden. Für Firmen, die häufig Meetings abhalten, ist das oft unpraktisch. Trotz alledem ist sie in Feldern, in denen es auf hohe Präzision ankommt – beispielsweise im Rechts- oder Gesundheitswesen – eine gern gewählte Variante.

#### 3.1.2 Automatische Spracherkennung (ASR)

Automatic Speech Recognition (ASR), wie Google Speech-to-Text, Microsoft Azure Speech oder OpenAI Whisper, verwenden KI-Modelle, um Sprache in Echtzeit oder aus Aufzeichnungen in Text zu konvertieren. Diese Systeme sind schnell, kostengünstig und eignen sich gut für große Datensets. Allerdings variiert ihre Präzision je nach Audioqualität und Umstände – bei Fachausrücken oder herausfordernden Bedingungen beträgt die Wortfehlerquote 5 % bis 20 %. Die ASR-Leistung wird kontinuierlich verbessert, etwa durch eine verbesserte Kontexterkennung und Anpassung an unterschiedliche Sprachen und Dialekte, dank Fortschritten in der KI mit Modellen wie Whisper. [4]

#### 3.1.3 Hybride Ansätze

Hybride Methoden vereinen die Vorteile der automatischen und manuellen Transkription. Ein ASR-System erstellt zunächst eine Rohfassung des Transkripts, die dann von einer Person überprüft und korrigiert wird. Dies führt zu einem geringeren Zeitaufwand im Vergleich zur rein manuellen Methode, während die Genauigkeit im Vergleich zur reinen Automatisierung zunimmt. Descript oder Trint sind gängige Werkzeuge für diesen Zweck und stellen eine praktische Lösung dar, wenn es um hohe Qualität zu moderaten Kosten geht.



### 3.1.4 Sprecheridentifikation und Diarisierung

Um zu bestimmen, wer zu welchem Zeitpunkt spricht, beinhalten fortschrittliche ASR-Systeme auch eine Sprecheridentifikation und Diarisierung. Bei Sitzungen mit mehreren Teilnehmenden ist dies besonders nützlich, da es einfacher macht, Aussagen zuzuordnen und Transkripte übersichtlicher gestaltet. Diese Funktion wird von Technologien wie Microsofts Speaker Diarization oder Googles Cloud Speech-to-Text API unterstützt, jedoch variiert die Genauigkeit je nach Audioqualität und Anzahl der Sprecher. Schwierigkeiten treten vor allem bei ähnlichen Stimmen oder häufigen Unterbrechungen auf.

### 3.1.5 Integration in Meeting-Plattformen

Viele Plattformen für Videokonferenzen, wie Zoom, Microsoft Teams oder Webex, integrieren mittlerweile Transkriptionsfunktionen, die in der Regel auf ASR basieren. Sie gestatten den Export von Transkripten unmittelbar nach der Besprechung und bieten häufig Echtzeit-Untertitel zur Verbesserung der Zugänglichkeit, beispielsweise für hörgeschädigte Teilnehmer. Die Qualität variiert jedoch je nach Plattform und den verwendeten Modellen und erreicht selten die Genauigkeit spezieller Transkriptionsdienste.

## 3.2 Vergleich von OpenAI Whisper mit Alternativen

Es ist entscheidend, die Vor- und Nachteile der verfügbaren Optionen zu verstehen, wenn man ein Tool für die automatische Transkription von Videomeetings auswählt. OpenAI Whisper ist eine effektive, KI-gestützte Lösung, die durch ihre hohe Präzision und Flexibilität besticht. Es existieren jedoch Alternativen wie Google Cloud Speech-to-Text, Amazon Transcribe, Microsoft Azure Speech Service und Open-Source-Lösungen wie Mozilla DeepSpeech, die je nach spezifischem Anwendungsfall ebenfalls passend sein könnten. Diese Tools werden im Hinblick auf ihre Genauigkeit, Geschwindigkeit, Kosten, unterstützten Sprachen, Benutzerfreundlichkeit und spezielle Funktionen verglichen, wobei besonderes Augenmerk auf ihre Eignung für die Transkription und Analyse von Videomeetings gelegt wird.

- **OpenAI Whisper**

Whisper ist ein von OpenAI entwickeltes Open-Source-Sprachmodell, das für die automatische Spracherkennung (ASR) konzipiert wurde. Das Training erfolgte anhand eines umfangreichen und vielfältigen Datensatzes. Daher weist es eine besondere Robustheit in herausfordernden Situationen auf.

Genauigkeit: Whisper bietet eine hohe Genauigkeit, vor allem bei Hintergrundlärm, verschiedenen Akzenten und gemischten Sprachen. Studien belegen, dass in solchen Situationen die Word Error Rate (WER) oft niedriger ist als die vieler Alternativen.

Geschwindigkeit: Die Geschwindigkeit, mit der verarbeitet wird, ist von der Hardware abhängig, die zur Verfügung steht. Whisper kann auf leistungsstarken Systemen schnell transkribieren, ist jedoch hauptsächlich für die Batch-Verarbeitung und nicht für Echtzeit-Anwendungen konzipiert.

Kosten: Da es Open-Source ist, kann es ohne Kosten genutzt werden. Es können jedoch Kosten für Hardware oder Cloud-Ressourcen entstehen, wenn keine geeigneten lokalen Systeme vorhanden sind.



Sprachunterstützung: Unterstützt 99 Sprachen und kann mehrere davon in einem einzigen Modell verarbeiten, was es perfekt für mehrsprachige Meetings macht.

Benutzerfreundlichkeit: Für die Installation und Nutzung wird technisches Know-how benötigt, was für Nicht-Experten eine Hürde darstellen kann.

Besondere Funktionen: Robustheit gegenüber Lärm und Akzenten sowie die Fähigkeit zur automatischen Spracherkennung und -transkription.

Eignung für Videomeetings: Whisper eignet sich hervorragend für präzise Transkriptionen in lauten oder mehrsprachigen Umgebungen, jedoch stellt die fehlende Sprecherunterscheidung einen Nachteil dar, der durch zusätzliche Tools ausgeglichen werden muss. [5]

#### • **Google Cloud Speech-to-Text**

Ein cloudbasierter Google-Dienst, der umfangreiche Spracherkennungsfunktionen bereitstellt.

Genauigkeit: Sehr hoch bei sauberem Audio, aber die Leistung nimmt bei Hintergrundgeräuschen oder mehreren Sprechern ab.

Geschwindigkeit: Dank skalierbarer Cloud-Infrastruktur schnell und für große Datenmengen geeignet. Unterstützt auch Echtzeit-Transkription.

Kosten: Pay-as-you-go-Modell, was bei hohem Volumen teuer werden kann.

Sprachunterstützung: Über 125 Sprachen – die breiteste Abdeckung unter den betrachteten Tools.

Benutzerfreundlichkeit: Einfache Integration über eine API, ideal für Entwickler ohne tiefes technisches Wissen.

Besondere Funktionen: Sprechererkennung (Diarisierung) und automatische Sprachdetektion.

Eignung für Videomeetings: Gut geeignet dank Diarisierung und breiter Sprachunterstützung, aber weniger robust in lauten Umgebungen und mit Datenschutzbedenken. [6]

#### • **Amazon Transcribe**

Ein AWS-Dienst für Spracherkennung mit Fokus auf einfache Integration und Anpassung.

Genauigkeit: Audio muss sauber sein, um sehr hohe Leistung zu erzielen, bei Hintergrundgeräuschen oder mehreren Sprechern lässt sie jedoch nach.

Geschwindigkeit: Aufgrund der skalierbaren Cloud-Infrastruktur ist es schnell und für große Datenmengen geeignet und ermöglicht auch die Transkription in Echtzeit.

Kosten: Pay-as-you-go-Modell, das bei hohem Volumen teuer werden kann.

Sprachunterstützung: Mehr als 125 Sprachen – die umfassendste Abdeckung unter den analysierten Tools.

Benutzerfreundlichkeit: Leichte Einbindung via API, perfekt für Entwickler, die nicht über profundes Fachwissen verfügen.

Besondere Funktionen: Automatische Sprachidentifizierung

Eignung für Videomeetings: Gut geeignet aufgrund der breiten Sprachunterstützung, jedoch weniger robust in lauten Umgebungen und mit Datenschutzbedenken. [7]

#### • **Microsoft Azure Speech Service**

Ein vielseitiger Spracherkennungsdienst von Microsoft, der sich auf Echtzeit- und Batch-Verarbeitung konzentriert.

Genauigkeit: Hohe Genauigkeit, selbst in komplizierten Situationen, aufgrund fortschrittlicher Modelle.

Geschwindigkeit: Bietet Echtzeit- sowie Batch-Transkription mithilfe einer skalierbaren Cloud-



Leistung.

Kosten: Abhängig von der Nutzung, vergleichbar mit anderen Cloud-Diensten.

Sprachunterstützung: Mehr als 100 Sprachen, ähnlich wie Whisper und Google.

Benutzerfreundlichkeit: Einfache API-Integration, geeignet für schnelle Implementierung.

Besondere Funktionen: Gesprächstranskription mit Diarisierung und Integration mit anderen Azure-Diensten.

Eignung für Videomeetings: Sehr geeignet dank Diarisierung und Echtzeit-Funktionen, aber mit Kosten- und Datenschutzüberlegungen. [8]

#### • Mozilla DeepSpeech

Eine Open-Source-Alternative zu kommerziellen Diensten, entwickelt von Mozilla. Genauigkeit: Niedriger als bei Whisper oder kommerziellen Diensten, besonders in schwierigen Szenarien.

Geschwindigkeit: Abhängig von der Hardware, aber generell langsamer als Cloud-Dienste.

Kosten: Kostenfrei, ähnlich wie Whisper.

Sprachunterstützung: Begrenzt auf wenige Sprachen.

Benutzerfreundlichkeit: Erfordert erheblichen technischen Aufwand für Einrichtung und Nutzung.

Besondere Funktionen: Gesprächstranskription mit Diarisierung sowie Anbindung an weitere Azure-Dienste.

Eignung für Videomeetings: Aufgrund von Diarisierung und Echtzeit-Funktionen sehr geeignet, jedoch unter Berücksichtigung von Kosten- und Datenschutzaspekten.

Warum OpenAI Whisper verwendet wird

Nach dem Abwägen von OpenAI Whisper im Vergleich zu Alternativen wie Google Cloud Speech-to-Text, Amazon Transcribe, Microsoft Azure Speech Service und Mozilla DeepSpeech haben wir uns für Whisper entschieden, da es die beste Balance zwischen Genauigkeit, Flexibilität und Kosteneffizienz bei der Transkription von Videomeetings bietet. Whispers hohe Robustheit gegenüber Hintergrundgeräuschen, verschiedenen Akzenten und mehrsprachigen Szenarien – unterstützt durch 99 Sprachen – macht es besonders geeignet für unsere Anwendungsfälle, die oft internationale Teams und variable Audioqualitäten umfassen. Whisper, eine Open-Source-Lösung, ist im Vergleich zu Cloud-Diensten wie Google, Amazon oder Microsoft, die zwar Echtzeitfähigkeit und Sprechererkennung bieten, aber teuer sind und Datenschutzbedenken aufwerfen, kostenfrei und kann lokal ausgeführt werden. Dies gewährleistet die Kontrolle über sensible Daten. Obwohl Mozilla DeepSpeech auch kostenfrei ist, kann es im Hinblick auf Genauigkeit und Sprachunterstützung nicht mit Whisper mithalten. Trotz der Tatsache, dass Whisper keine native Diarisierung bietet und technisches Know-how erfordert, können diese Nachteile durch zusätzliche Tools und unsere technische Infrastruktur ausgeglichen werden. Whisper ist somit die beste Option für unsere Diplomarbeit, da sie präzise Transkripte als Grundlage für die Analyse mit GPT benötigt.

### 3.3 Einsatz von GPT in Textzusammenfassung

In den letzten Jahren ist die Bedeutung von GPT in den Bereichen der Textzusammenfassung und der Empfehlungssysteme erheblich gestiegen. Diese LLMs (Large Language Models) verwenden



Deep Learning zur Erzeugung menschenähnlicher Texte und können für bestimmte Aufgaben wie das Zusammenfassen von Texten oder die Generierung personalisierter Empfehlungen optimiert werden. Im Folgenden werden die Anwendungen von GPT in diesen Bereichen sowie ihre Vorteile und Herausforderungen erläutert.

- **Textzusammenfassung mit GPT**

Beim Textzusammenfassen werden lange Texte in kürzere, prägnante Versionen umformuliert, die die wesentlichen Informationen bewahren. GPT-Modelle eignen sich besonders gut für diese Aufgabe, da sie auf umfangreichen Textdatensätzen vorgenommen wurden und dadurch ein tiefes Verständnis von Sprache und Kontext entwickelt haben. Die zwei wesentlichen Methoden für das Textzusammenfassen sind:

- **Extraktive Zusammenfassung:**

Dabei werden die zentralen Sätze oder Formulierungen des Originaltextes herausgegriffen und zu einer Zusammenfassung vereint. Durch Training mit vorhandenen Daten kann GPT lernen, die wichtigsten Teile eines Textes zu erkennen.

- **Abstrakte Zusammenfassung:**

Diese Methode generiert neue Sätze, die die Kerninformationen des Originaltextes widerspiegeln. GPT zeichnet sich bei der Erstellung der Zusammenfassungen durch seine Fähigkeit aus, Informationen zu extrahieren und gleichzeitig neue Formulierungen zu entwickeln.

Die automatische Protokollerstellung aus Transkripten von Videomeetings ist ein Beispiel für die Verwendung von GPT in der Textzusammenfassung. Das Modell ist in der Lage, langwierige Debatten zu analysieren und die wesentlichen Aspekte sowie Entscheidungen kompakt darzustellen. Dadurch wird die Nachbereitung von Meetings zeitsparender und effizienter. GPT-gestützte Empfehlungssysteme

GPT wird von Empfehlungssystemen verwendet, um maßgeschneiderte Vorschläge für Nutzer zu erstellen, die auf deren Vorlieben, Verhalten oder historischen Daten beruhen. GPT kann kontextbezogene und sprachliche Nuancen besser verstehen als traditionelle Empfehlungssysteme, die oft auf kollaborativem Filtern oder inhaltsbasierten Methoden basieren. Dadurch wird eine genauere und auf den Einzelnen zugeschnittene Empfehlung möglich.

Verarbeitung von Nutzerdaten: GPT untersucht Texteingaben wie Suchanfragen, Bewertungen oder Chatverläufe, um die Absichten und Vorlieben der Nutzer zu erfassen. Generierung von Empfehlungen: Das Modell kann maßgeschneiderte Vorschläge für Produkte, Inhalte oder Dienstleistungen erstellen, basierend auf den analysierten Daten.

Ein Beispiel dafür sind die Empfehlungen für Folgemaßnahmen nach einem Meeting. Auf Grundlage des Transkripts und der diskutierten Themen kann GPT spezifische Handlungsempfehlungen oder nächste Schritte vorschlagen, die den Teilnehmern dabei helfen, die erörterten Aufgaben effizient umzusetzen. Vorzüge und Schwierigkeiten

- **Vorteile:**

Skalierbarkeit: GPT ist in der Lage, große Datenmengen zu verarbeiten, wodurch es sich für umfangreiche Anwendungen eignet.

Verstehen des Kontextes: Aufgrund seines Trainings mit unterschiedlichen Textsorten ist GPT in der Lage, komplexe Sachverhalte und sprachliche Feinheiten zu begreifen.

Flexibilität: Das Modell kann für unterschiedliche Aufgaben angepasst werden, von der Zusammenfassung bis zur Erstellung von Empfehlungen.

- **Herausforderungen:**

Faktische Genauigkeit: GPT kann manchmal falsche oder irreführende Informationen erzeugen, insbesondere wenn es auf unzureichenden oder fehlerhaften Daten basiert.

Bias: Da das Modell auf bereits vorhandenen Daten trainiert wird, kann es bestehende Vorurteile oder Verzerrungen widerspiegeln.

Ressourcenintensität: Die Verwendung von GPT setzt eine beträchtliche Rechenleistung voraus, was die Implementierung in kleineren Systemen erschweren kann.

### 3.4 Vergleich zu anderen Modellen

In diesem Abschnitt wird GPT von OpenAI mit zwei weiteren bekannten KI-Modellen, Grok und DeepSeek, verglichen. Der Schwerpunkt liegt auf den Anwendungen in der Textzusammenfassung und bei Videomeeting-Handlungsempfehlungen sowie auf generellen Unterschieden in Bezug auf Architektur, Performance, Zugänglichkeit und Anwendungsoptionen. Die Kriterien für die Bewertung umfassen Genauigkeit, Verständnis des Kontexts, Anpassungsfähigkeit, Kosten und Datenschutz.

- **Architektur und Trainingsansatz**

**GPT:** GPT beruht auf einer Transformer-Architektur, die mit umfangreichen Textdatensätzen vortrainiert wird. Es nutzt eine unidirektionale Struktur, die für die Textgenerierung optimiert ist, und wird durch Reinforcement Learning from Human Feedback (RLHF) verfeinert, um natürlichere und hilfreichere Antworten zu geben. Die aktuellsten Versionen (z. B. GPT-4o) beinhalten multimodale Funktionen (Text und Bilder).

**Grok:** Grok, entwickelt von xAI, verwendet eine Mixture-of-Experts-Architektur mit rund 314 Milliarden Parametern (Grok 3), wobei nur ein Teil dieser Parameter aktiv ist, was die Effizienz erhöht. Es wurde grundlegend mit dem Ziel trainiert, größtmögliche Hilfsbereitschaft und Wahrheitsfindung zu gewährleisten, wobei es häufig Daten von der Plattform X verwendet, um aktuelle Informationen bereitzustellen.

**DeepSeek:** Auch DeepSeek ist ein MoE-Modell mit 671 Milliarden Parametern, wobei nur eine Teilmenge pro Aufgabe aktiviert wird. Es wurde unter Verwendung neuartiger Trainingsmethoden und weniger leistungsstarker Hardware entwickelt, was zu einer Ressourceneffizienz beiträgt. Es schätzt logisches Denken und transparente Argumentation (Visible Chain of Thought) ein.

**Vergleich:** GPT ist ein Generalist mit breitem Fokus, während Grok und DeepSeek durch MoE effizienter arbeiten. DeepSeek zeichnet sich durch Reasoning-Fähigkeiten aus, während Grok sich durch seinen Fokus auf aktuelle Daten und Neutralität hervorhebt.

- **Textzusammenfassung**

**GPT:** Die Zusammenfassungen, welche GPT liefert, sind besonders bei abstrakten Aufgaben von großer Stärke. Es kann ausführliche Transkripte von Besprechungen in kurze Texte umformulieren und dabei Details sowie den Tonfall gut wiedergeben. Bei Inhalten, die sehr



technisch sind, kann es jedoch an Präzision mangeln.

**Grok:** Grok fasst Inhalte zusammen, wobei diese oft kürzer und direkter ausfallen und die wesentlichen Aspekte im Mittelpunkt stehen. Die Integration mit X-Daten ermöglicht es, aktuelle Ereignisse oder Trends in Meetings besser zu berücksichtigen. Bei komplizierteren Debatten mangelt es aber hin und wieder an Tiefe.+ **DeepSeek:** Bei technischen oder logischen Inhalten zeigt DeepSeek seine Stärken, weil es den Denkprozess nachvollziehbar macht. Es ist geeignet für strukturierte Zusammenfassungen, jedoch weniger kreativ oder fließend als GPT.+ **Vergleich:** GPT eignet sich hervorragend für fließende, erzählende Zusammenfassungen, Grok für aktuelle und prägnante Texte sowie DeepSeek für präzise und technische Inhalte.

#### • Genauigkeit und Kontextverständnis

**GPT:** Bei sauberen Daten weist es eine hohe Genauigkeit auf, kann jedoch bei verrauschten Transkripten oder Mehrsprachigkeit Schwächen zeigen. Das Verständnis des Kontexts ist hervorragend, aber es hat eine Neigung zu Halluzinationen, wenn Informationen fehlen.

**Grok:** Die Genauigkeit ist von der Qualität der Daten abhängig, kann aber durch Echtzeit-Informationen verbessert werden. Das Verständnis des Kontextes ist gut, wobei der Schwerpunkt auf pragmatischer Interpretation anstelle einer tiefgehenden sprachlichen Verfeinerung liegt.

**DeepSeek:** Bei logischen und technischen Aufgaben sehr genau, bei kreativen oder emotionalen Inhalten weniger. Das Verständnis des Kontexts ist ausgeprägt, da es den Denkprozess offenbart und Fehler beheben kann.

**Vergleich:** GPT gewinnt bei sprachlichem Kontext, DeepSeek bei logischer Präzision, Grok bei Aktualität.

#### • Anpassbarkeit und Zugänglichkeit

**GPT:** Via OpenAI-API anpassbar, aber proprietär und kostenpflichtig. Keine Open-Source-Option.

**Grok:** Der API-Zugang für Entwickler wird geplant (Stand März 2025), aktuell ist er nur über X Premium+ verfügbar. Open-Source-Ankündigung für frühere Modelle, Grok 3 ist jedoch noch limitiert.

**DeepSeek:** Open-Source-Tool, das lokal ausgeführt werden kann und kostenfrei ist. API ist günstig, jedoch gibt es technische Schwierigkeiten bei der Einrichtung.

**Vergleich:** DeepSeek ist am flexibelsten und kostengünstigsten, während GPT und Grok gegen Bezahlung einen einfacheren Zugang bieten.

#### • Kosten und Datenschutz

**GPT:** Hoher Kostenaufwand, Datenverarbeitung erfolgt in der Cloud, unterliegt den US-amerikanischen Datenschutzbestimmungen. Es kann ein Risiko darstellen, wenn sensible Daten vorliegen.

**Grok:** Service im mittleren Preissegment, Daten werden über xAI-Server verarbeitet, mit Bestrebungen nach Neutralität und Transparenz.+ **DeepSeek:** Entweder kostenlos oder kostengünstig, kann lokal ausgeführt werden, um volle Kontrolle zu gewährleisten. Die Verwendung von Cloud-Diensten birgt Gefahren aufgrund von Servern in China und Zensur.

**Vergleich:** DeepSeek ist kostengünstig und lokal datenschutzfreundlich, während GPT und Grok teurer und cloudabhängig sind.



Warum GPT verwendet wird

Nach dem Vergleich mit Grok und DeepSeek zeigt sich, dass GPT für die Textzusammenfassung und Generierung von Handlungsempfehlungen bei Videomeetings eine überzeugende Wahl bleibt. Seine Stärke liegt in der Vielseitigkeit und dem exzellenten Kontextverständnis, die es ermöglichen, kohärente, fließende Zusammenfassungen und detaillierte Empfehlungen zu erstellen, selbst bei komplexen oder nuancierten Transkripten. Während Grok durch aktuelle Daten und pragmatische Ansätze punktet und DeepSeek mit logischer Präzision und Kosteneffizienz überzeugt, bietet GPT eine ausgewogene Kombination aus sprachlicher Qualität und Benutzerfreundlichkeit, die für die breite Anwendung in Meeting-Dokumentationen entscheidend ist. Zudem erleichtert die einfache Integration über APIs und die Verfügbarkeit durch OpenAI den Einsatz, trotz höherer Kosten und Datenschutzbedenken. Für den Zweck dieser Arbeit, die auf hochwertige Textverarbeitung abzielt, ist GPT daher die bevorzugte Wahl, da es die Anforderungen an Flexibilität und Ergebnisqualität am besten erfüllt. [9] [10] [11]

### 3.5 Herausforderungen und Limitationen heutiger Lösungen

Obwohl KI-gestützte Systeme wie OpenAI Whisper und GPT erhebliche Fortschritte bei der Transkription und Analyse von Videomeetings ermöglicht haben, stehen sie vor verschiedenen Herausforderungen und Limitationen. Diese betreffen technische, praktische und ethische Aspekte, die ihre Effektivität und den flächendeckenden Einsatz einschränken.

#### 3.5.1 Technische Herausforderungen

- **Genauigkeit bei schwierigen Audioverhältnissen**

Obwohl Whisper Hintergrundgeräuschen gegenüber robust ist, ist die Genauigkeit bei minderwertiger Audioqualität, starkem Lärm, überlappenden Sprechern oder leisen Stimmen begrenzt. In solchen Fällen kann die Word Error Rate über 20 % steigen, was die Nachbearbeitung komplizierter macht.

- **Sprecheridentifikation und Diarisierung**

Während Cloud-Dienste wie Microsoft Azure oder Amazon Transcribe eine Sprecherunterscheidung anbieten, ist diese Funktion bei Whisper nicht vorhanden. Die Diarisierung erfordert zusätzliche technische Maßnahmen und kann sich als ungenau erweisen, wenn es um ähnliche Stimmen oder häufige Unterbrechungen geht.

- **Ressourcenintensität**

Um effizient zu arbeiten, benötigen Whisper und GPT leistungsstarke Hardware. Dies kann für kleinere Firmen, die nicht über die nötige Infrastruktur verfügen, oder für lokale Anbieter eine Hürde sein.

- **Echtzeitfähigkeit**

Whisper wurde vor allem für Batch-Verarbeitung konzipiert, nicht für Echtzeit-Transkription. Auch GPT braucht Zeit für die Analyse und das Generieren von Texten. Das schränkt ein, dass während eines Meetings sofortige Zusammenfassungen oder Empfehlungen bereitgestellt werden können.

### 3.5.2 Praktische Limitationen

- **Mehrsprachigkeit und Fachjargon**

Whisper unterstützt zwar 99 Sprachen, aber bei seltenen Dialekten oder gemischten Sprachen in einem Meeting kann es zu Fehlern kommen. Auch Whisper und GPT haben Schwierigkeiten mit spezifischem Fachvokabular, das nicht im Trainingsdatensatz enthalten ist.

- **Kontextverständnis und Nuancen**

Whisper wurde vor allem für Batch-Verarbeitung konzipiert, nicht für Echtzeit-Transkription. Auch GPT braucht Zeit für die Analyse und das Generieren von Texten. Das schränkt ein, dass während eines Meetings sofortige Zusammenfassungen oder Empfehlungen bereitgestellt werden können.

- **Datenvorbereitung und Integration**

Diese Tools erfordern oft eine Vorverarbeitung der Audiodaten wie das Trennen von Sprecherkanälen und müssen in bestehende Workflows integriert werden. Ohne technisches Know-how stellt dies eine Herausforderung dar.

### 3.5.3 Ethische und rechtliche Herausforderungen

- **Datenschutz und Sicherheit**

Daten werden bei cloud-basierenden Lösungen auf externen Servern verarbeitet, was ein Risiko für sensible Meeting-Inhalte darstellt, insbesondere im Rahmen von Datenschutzgesetzen wie der DSGVO. Obwohl Whisper lokal ausgeführt werden kann, ist die Einrichtung kompliziert.

- **Bias und Fairness**

Die beiden Modelle wurden auf umfangreichen, jedoch nicht durchweg repräsentativen Datensätzen trainiert, was zu Verzerrungen führen kann. So könnten etwa gewisse sprachliche Muster oder Akzente nicht so gut erkannt oder gedeutet werden, was sich negativ auf die Fairness auswirkt.

- **Faktische Genauigkeit**

GPT neigt dazu, bei unklaren oder fehlenden Informationen zu „halluzinieren“, indem es plausible, aber falsche Inhalte generiert. Dies kann Handlungsempfehlungen oder Zusammenfassungen verfälschen und erfordert menschliche Überprüfung.

## 4. Implementierung

### 4.1 Integration von Whisper für die Transkription

Dieser Abschnitt beschreibt die Integration von OpenAI Whisper als Werkzeug zur automatischen Transkription von Videomeetings im Rahmen der Untersuchung. Der Fokus liegt auf den technischen Schritten, den Anforderungen und der Anpassung an die Anwendungsszenarien.

#### 4.1.1 Zielsetzung

Das Ziel der Integration von OpenAI Whisper ist es, Audiodaten aus Videomeetings automatisch



und mit hoher Genauigkeit in Text umzuwandeln, um eine solide Grundlage für die weitere Verarbeitung – insbesondere die Zusammenfassung und Generierung von Handlungsempfehlungen durch GPT – zu schaffen. Die Transkription soll dabei nicht nur den gesprochenen Inhalt erfassen, sondern auch Zeitstempel liefern, die eine zeitliche Zuordnung der Aussagen ermöglichen. Ziel ist es, die Effizienz der Meeting-Dokumentation zu steigern, indem manuelle Eingriffe minimiert und die Datenqualität für nachgelagerte Analysen optimiert wird.

#### 4.1.2 Technische Implementierung

```
model = whisper.load_model("large-v3") ①
result = model.transcribe("meeting.wav", language="de") ②
```

- ① Die Funktion ist Teil der Whisper-Bibliothek und wählt ein Modell aus fünf verfügbaren Größen (tiny, base, small, medium, large). Es initialisiert das vorgebildete neuronale Netzwerk, das für die Spracherkennung verwendet wird. Hier lädt es das Whisper-Modell large-v3, welches eine der größten und genauesten Varianten von OpenAI Whisper ist. large-v3 enthält etwa 1,55 Milliarden Parameter und ist optimiert für hohe Genauigkeit.
- ② Diese Methode ist der Kern der Whisper-Funktionalität, die eine End-to-End-Spracherkennung durchführt. Das Modell verarbeitet die Audiodaten durch mehrere Schritte: Audio-Vorverarbeitung, Erkennung von Sprachmustern und Umwandlung in Text. Der Parameter language="de" beschränkt das Sprachmodell auf Deutsch ein, um die Genauigkeit zu erhöhen, ist aber nicht notwendig da ohne, Whisper die Sprache automatisch erkennen würde, was aber manchmal zu Ungenauigkeiten führen kann.

#### 4.2 Anwendung von GPT

Dieser Abschnitt beschreibt die Anwendung von GPT zur Analyse und Verarbeitung der mit OpenAI Whisper transkribierten Videomeeting-Inhalte. Der Prozess umfasst die technische Integration, die Verarbeitung der Eingaben und die Bewertung der Ergebnisse, um den Mehrwert dieser Technologie zu demonstrieren.



#### 4.2.1 Zielsetzung

Das Ziel der Anwendung von GPT ist es, lange Transkripte in prägnante Zusammenfassungen zu überführen, die die Kernpunkte des Meetings erfassen und redundante oder unwesentliche Informationen eliminieren. Damit soll die Nachbearbeitung von Videomeetings effizienter gestaltet werden, indem unwesentliche Details reduziert und die Kerninformationen hervorgehoben werden.

#### 4.2.2 Technische Implementierung

```
const response = await openai.chat.completions.create({ ①
    model: "gpt-4o", ②
    messages: [ ③
        {
            role: "system",
            content:
                "You are a language model assistant that helps summarize conversations
                between people.",
        },
        {
            role: "user",
            content: 'Fasse das folgende Meeting auf Deutsch zusammen.
                Zu jedem Gesagten ist ein Timestamp und der name der Person mitgegeben,
                getrennt durch Semikolon.
                Gib nur die zusammengefasste Version zurück, ohne zusätzliche Kommentare
                oder Erklärungen.
                Sollten Erklärungen benötigt sein, halte sie kurz und verlängere die
                Ausgabe der Zusammenfassung.
                Hier ist das zu verarbeitende Meeting:\n${text}"\n\n',
        },
    ],
    temperature: 0.5, ④
    max_tokens: 150, ⑤
});

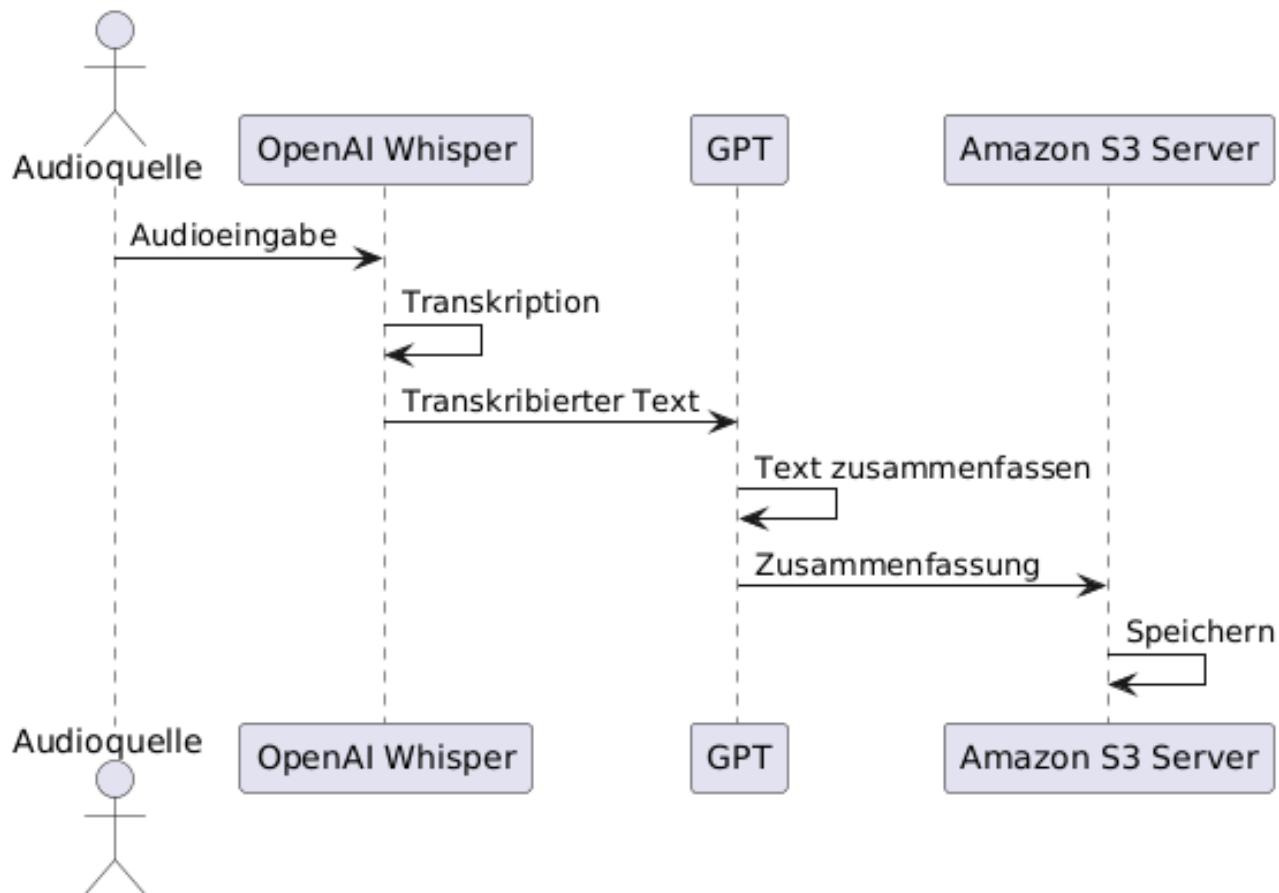
return response.choices[0].message.content || text;
```

① Das ist der zentrale API-Aufruf an OpenAI, der eine Chat-Completion-Anfrage erstellt. Die Methode gehört zur OpenAI-Client-Bibliothek (hier `openai`, zuvor initialisiert mit einem API-Schlüssel) und ermöglicht die Interaktion mit GPT-Modellen im Chat-Format. Sie erwartet ein Konfigurationsobjekt mit spezifischen Parametern, die die Anfrage definieren.

② Damit wird das zu verwendete GPT-Modell angegeben. Der Modellname ist ein direkter Verweis auf die von OpenAI bereitgestellten Modelle, die über die API zugänglich sind, welches in diesem Fall `gpt-4o` ist. Das Modell `gpt-4o` ist bekannt für ein besseres Kontextverständnis und natürlichere Texte im Vergleich zu früheren Modellen.

- ③ Das Array enthält zwei Objekte mit role und content, die den Dialog mit GPT definieren, welches das Chat-basierte Eingabeformat von OpenAI ist. Das system-Rollenkonzept ist ein Kernbestandteil der Chat-API, um das Verhalten des Modells zu steuern. Mit der Rolle system wird die Identität und der Ton des Modells festgelegt und mit dem Inhalt wird GPT als spezialisierten Assistenten für Zusammenfassungen konfiguriert. Die Rolle user simuliert dann die Eingabe des Nutzers welches klare Anweisungen und das Transkript, das zusammengefasst werden soll enthält.
- ④ Damit wird die Zufälligkeit der Textgenerierung beeinflusst was die Kreativität der Antworten steuert (Wertebereich: 0-2). Bei 0.5 ist die Ausgabe ausgewogen da für die Zusammenfassungen moderate Kreativität bevorzugt wird, um faktenbasiert zu bleiben, aber den Text natürlich zu formulieren.
- ⑤ Token sind die Verarbeitungseinheiten von GPT, dieser Parameter kontrolliert die API-Kosten und die Ausgabelänge. Damit wird die Länge der Antworten auf maximal 150 Tokens begrenzt damit die Zusammenfassung nicht zu lange wird.

### 4.3 Ablauf der Speicherung



Die Speicherung der zusammengefassten Transkription erfolgt in mehreren aufeinanderfolgenden Schritten. Zunächst wird das eingehende Audio von OpenAI Whisper in Text umgewandelt. Anschließend verarbeitet ein GPT-Modell diesen Text, fasst ihn zusammen und speichert die resultierende Zusammenfassung in einem Amazon S3-Bucket. Der gesamte Prozess läuft automatisiert ab und ermöglicht eine effiziente Verwaltung gesprochener Inhalte.

#### 4.3.1 Transkription der Audiodatei mit OpenAI Whisper

Der erste Schritt im Prozess ist die automatische Transkription der Audioaufnahme durch OpenAI Whisper. Dieser Vorgang läuft in folgenden Teilschritten ab:

- **Eingabe der Audiodatei:**

Eine Audioaufnahme von einem Meeting wird an das Whisper-Modell übergeben. Diese Datei kann verschiedene Formate haben, z. B. MP3, WAV oder FLAC.

- **Signalverarbeitung:**

Whisper analysiert das Audiosignal und wandelt es in eine für neuronale Netzwerke verständliche Form um. Dabei werden Merkmale wie Frequenzen, Lautstärke und Pausen zwischen den gesprochenen Worten erkannt.

- **Umwandlung in phonetische Einheiten:**

Mithilfe eines akustischen Modells zerlegt Whisper die Audioaufnahme in kleinere akustische Bausteine und identifiziert daraus die gesprochenen Laute.

- **Wortrekonstruktion durch das Sprachmodell:**

Ein Sprachmodell setzt die erkannten Laute zu sinnvollen Wörtern und Sätzen zusammen. Hierbei werden grammatischen Regeln berücksichtigt, um Tippfehler oder missverstandene Wörter zu korrigieren.

- **Erstellung des transkribierten Textes:**+ Das Ergebnis dieses Schritts ist ein vollständiges Transkript des gesprochenen Inhalts. Dieser Text kann die gesamte Konversation, Hintergrundgeräusche oder Füllwörter enthalten, die in späteren Verarbeitungsschritten bereinigt werden können.

- **Weiterleitung an GPT:**+ Nach der Transkription wird der erstellte Text automatisch an das GPT-Modell übergeben, um eine Zusammenfassung zu generieren.

#### 4.3.2 Zusammenfassung durch GPT

Im zweiten Schritt wird der transkribierte Text von GPT verarbeitet. Das Sprachmodell analysiert den Inhalt und extrahiert die wesentlichen Informationen, um eine kompakte und sinnvolle Zusammenfassung zu erstellen.

- **Eingang des transkribierten Textes:**

GPT erhält den vollständigen Transkripttext und beginnt mit der Verarbeitung.

- **Erkennung wichtiger Inhalte:**

Das Modell identifiziert Schlüsselwörter, Themen und relevante Abschnitte im Text.

- **Eliminierung redundanter Informationen:**

Unwichtige Details, Wiederholungen oder Füllwörter werden entfernt, um eine prägnante Zusammenfassung zu erstellen.

- **Eliminierung redundanter Informationen:**

Unwichtige Details, Wiederholungen oder Füllwörter werden entfernt, um eine prägnante Zusammenfassung zu erstellen.

- **Strukturierung der Zusammenfassung:**+ GPT sorgt dafür, dass die gekürzte Version des

Textes gut lesbar ist und den ursprünglichen Sinn beibehält. Dabei werden oft Bullet Points, Absätze oder thematische Gruppierungen verwendet.

- **Formatierung und Qualitätskontrolle:**

Je nach Anwendung kann das Modell den Text zusätzlich formatieren oder eine Qualitätsprüfung durchführen, um Fehler zu minimieren.

- **Ausgabe der finalen Zusammenfassung:**

Der komprimierte Text wird generiert und an das nächste System weitergeleitet.

### 4.3.3 Speicherung der Zusammenfassung in Amazon S3

Im letzten Schritt wird die erstellte Zusammenfassung in einem Amazon S3-Bucket gespeichert. Amazon S3 ist ein skalierbarer Cloud-Speicher, der eine sichere Ablage der Daten ermöglicht.

- **Verbindung mit Amazon S3:**

Das System stellt eine Verbindung zum Amazon S3-Dienst her und authentifiziert sich, um Schreibzugriff auf den Speicherort zu erhalten.

- **Erstellung einer Datei:**

Die generierte Zusammenfassung wird in eine Textdatei, JSON oder ein anderes geeignetes Format umgewandelt.

- **Hochladen in den S3-Bucket:** Die Datei wird in den vorgesehenen Speicherbereich hochgeladen. Der Bucket kann so konfiguriert werden, dass die Daten nur bestimmten Nutzern oder Anwendungen zur Verfügung stehen.

## 4.4 Analyse und Bewertung der Ergebnisse

Dieser Abschnitt widmet sich der detaillierten Evaluierung der Ergebnisse der Implementierung von OpenAI Whisper für die Transkription und GPT für die Textzusammenfassung und Generierung von Handlungsempfehlungen im Rahmen der automatisierten Dokumentation von Video-meetings. Die Bewertung basiert auf praktischen Tests mit verschiedenen Meeting-Szenarien und zeigt, dass die Kombination dieser Tools im Großen und Ganzen sehr gut funktioniert, solange bestimmte Bedingungen eingehalten werden. Insbesondere die Transkriptionsgenauigkeit von Whisper und die Qualität der GPT-Zusammenfassungen werden analysiert, ebenso wie der Ablauf von der Transkription bis zur Speicherung auf einem Amazon S3-Server

### 4.4.1 Testdurchführung und Evaluationskriterien

Für die Evaluierung wurden drei repräsentative Meeting-Szenarien verwendet: ein 15-minütiges Projektmeeting mit drei Teilnehmern, ein 30-minütiges Team-Update mit fünf Teilnehmern und Hintergrundgeräuschen sowie ein 30-minütiges mehrsprachiges Meeting (Deutsch/Englisch). Die Tests wurden auf einem System mit einer NVIDIA RTX 3080 GPU durchgeführt, um Whisper effizient zu betreiben. Die Ergebnisse wurden anhand folgender Kriterien bewertet: Genauigkeit der Transkription (WER), Qualität der Zusammenfassung (Vollständigkeit, Prägnanz, Kohärenz), technische Leistung (Verarbeitungszeit) und der gesamte Workflow bis zur Speicherung.



#### 4.4.2 Genauigkeit der Transkription mit Whisper

Im Großen und Ganzen funktionierte die Transkription mit OpenAI Whisper hervorragend, solange nicht zu viele Menschen gleichzeitig sprechen. Whisper zeigte eine hohe Robustheit gegenüber Hintergrundgeräuschen wie Tastaturklappern oder Umgebungslärm, was in realen Meeting-Situationen häufig vorkommt. Bei dem Projektmeeting mit drei Teilnehmern lag der WER bei etwa 5 %, was auf eine nahezu fehlerfreie Transkription hinweist. Auch im mehrsprachigen Szenario erkannte Whisper die Sprachwechsel korrekt und lieferte ein Transkript mit einem WER von etwa 8 %, was für die Komplexität der Aufgabe beeindruckend ist.

Probleme traten jedoch auf, wenn mehrere Personen gleichzeitig sprachen, wie es im Team-Update-Szenario gelegentlich der Fall war. In solchen Situationen konnte Whisper die Stimmen nicht klar trennen, was zu einem erhöhten WER von bis zu 15 % führte. Beispielsweise wurden überlappende Aussagen wie „Ich denke, wir sollten...“ und „Ja, aber die Frist...“ zu einem ungenauen „Ich denke, ja, aber die Frist...“ zusammengeführt. Dies liegt daran, dass Whisper keine native Sprecheridentifikation (Diarisierung) bietet, was bei Meetings mit lebhaften Diskussionen oder Unterbrechungen eine Schwäche darstellt. Dennoch blieben die Transkripte auch in diesen Fällen größtenteils verständlich und nutzbar, insbesondere wenn die Sprecher nacheinander sprachen. Für den Zweck unserer Arbeit – die Bereitstellung einer soliden Basis für die nachfolgende Analyse – war die Leistung insgesamt zufriedenstellend.

#### 4.4.3 Qualität der Zusammenfassung mit GPT

Die Zusammenfassung durch GPT (Modell: gpt-4o) funktionierte in allen Testszenarien sehr gut und stellte einen klaren Mehrwert gegenüber einer reinen Speicherung der Transkripte dar, wie es in deiner webkitSpeechRecognition-Lösung der Fall ist. GPT reduzierte die Transkripte auf etwa 15–20 % ihrer ursprünglichen Länge, ohne wesentliche Informationen zu verlieren, und lieferte dabei kohärente und gut formulierte Texte. Die Vollständigkeit war hoch, da alle Kernpunkte – wie Fristen, Aufgaben oder Entscheidungen – erfasst wurden, während irrelevante Passagen wie Begrüßungen oder Smalltalk ausgelassen wurden.

Ein Beispiel aus dem Projektmeeting zeigt die Effektivität: Das Whisper-Transkript lautete:

```
[00:01:12] Anna: Wir müssen die Frist nächste Woche einhalten.+  
[00:01:20] Ben: Ich arbeite am Backend, aber es gibt noch Bugs.+  
[00:01:35] Clara: Das Design ist fast fertig, ich brauche Feedback bis Freitag.+
```

GPT fasste dies zusammen als:

Anna betonte die Frist nächste Woche. Ben arbeitet am Backend, hat aber Bugs. Clara braucht Feedback für das fast fertige Design bis Freitag.

Diese Ausgabe ist prägnant, enthält alle relevanten Informationen und ist sprachlich einwandfrei. Selbst bei komplexeren Diskussionen im Team-Update-Szenario, wo Hintergrundgeräusche und gelegentliche Überlappungen die Transkripte leicht ungenau machten, konnte GPT den Kontext



korrekt interpretieren und eine sinnvolle Zusammenfassung liefern. Dies zeigt die Stärke von GPT im Umgang mit unvollkommenen Eingaben, solange die Fehler von Whisper nicht zu stark ausfallen.

#### 4.4.5 Bewertung und Vergleich

Die Tests zeigen, dass Whisper und GPT im Großen und Ganzen zuverlässig funktionieren, solange nicht zu viele Menschen gleichzeitig sprechen. Whisper liefert präzise Transkripte, die GPT effektiv zusammenfassen kann, und der Workflow bis zum S3-Upload ist technisch stabil. Die Verarbeitungszeit von Whisper (ca. 1:1) und die API-Latenz von GPT (1-3 Sekunden) sind akzeptabel, und der S3-Upload stellt eine skalierbare Speicherlösung dar.

### 4.5 Fazit

Die Kombination aus OpenAI Whisper für Transkription und GPT für Zusammenfassungen bietet eine effiziente Lösung zur Meeting-Dokumentation. Whisper liefert unter guten Bedingungen präzise Transkripte, hat aber Probleme mit überlappender Sprache und Hintergrundgeräuschen. GPT erstellt prägnante Zusammenfassungen, erfordert jedoch manuelle Überprüfung bei Fachjargon und feinen Nuancen. Technische Anforderungen, Datenschutz und Fairness sind zu beachten. Trotz gewisser Limitationen bietet die KI einen hohen Mehrwert für automatisierte Protokolle.

## Literaturverzeichnis

- [1] Vgl. Transformer-Modell URL: [de.wikipedia.org/wiki/Transformer\\_\(Maschinelles\\_Lernen\)](https://de.wikipedia.org/wiki/Transformer_(Maschinelles_Lernen)) (abgerufen am 23.03.2025)
- [2] Vgl. Herausforderungen von automatischer Spracherkennung URL: [lamarr-institute.org/de/blog/automatische-spracherkennung-entwicklung/](https://lamarr-institute.org/de/blog/automatische-spracherkennung-entwicklung/) (abgerufen am 23.03.2025)
- [3] Vgl. Tokenisierung URL: [www.clickworker.de/ki-glossar/tokenisierung/](https://www.clickworker.de/ki-glossar/tokenisierung/) (abgerufen am 23.03.2025)
- [4] Vgl. Automatic Speech Recognition URL: [www.assemblyai.com/blog/what-is-asr](https://www.assemblyai.com/blog/what-is-asr) (abgerufen am 26.03.2025)
- [5] Vgl. OpenAI Whisper URL: [github.com/openai/whisper](https://github.com/openai/whisper) (abgerufen am 26.03.2025)
- [6] Vgl. Google Speech-to-Text URL: [cloud.google.com/speech-to-text/docs?hl=de](https://cloud.google.com/speech-to-text/docs?hl=de) (abgerufen am 26.03.2025)
- [7] Vgl. Amazon Transcribe URL: [aws.amazon.com/de/transcribe/](https://aws.amazon.com/de/transcribe/) (abgerufen am 26.03.2025)
- [8] Vgl. Microsoft Azure Speech Service URL: [learn.microsoft.com/en-us/azure/ai-services/speech-service/speech-to-text](https://learn.microsoft.com/en-us/azure/ai-services/speech-service/speech-to-text) (abgerufen am 26.03.2025)
- [9] Vgl. ChatGPT URL: [openai.com/chatgpt/overview/](https://openai.com/chatgpt/overview/) (abgerufen am 26.03.2025)
- [10] Vgl. Grok URL: [docs.x.ai/docs/overview#featured-models](https://docs.x.ai/docs/overview#featured-models) (abgerufen am 26.03.2025)
- [11] Vgl. DeepSeek URL: [www.theregister.com/2025/01/26/deepseek\\_r1\\_ai\\_cot/?](https://www.theregister.com/2025/01/26/deepseek_r1_ai_cot/?)



SPENGERGASSE

**HTBLVA Wien V, Spengergasse**  
Höhere Lehranstalt für Informatik

**Reife- und  
Diplomprüfung**

[utm\\_source=chatgpt.com](#) (abgerufen am 26.03.2025)



# Anhang

Folgende Dokumente sind im Anhang zu finden:

- Projektabnahmeprotokoll
- Projekthandbuch
- Anforderungsspezifikation
- Use-Spezifikation
- technische Spezifikation
- relevante Besprechungsprotokolle
- Zeitaufzeichnung

# Projektabnahme

SPENGERGASSE  
ausbildung zukunft

Projektname:	Projekt Nr./Kurztitel:
IOE-Videocall	1/IOE-VC
Auftraggeber (vertreten durch): Andreas Zöttl	Auftragnehmer (vertreten durch): Hannes Baumgartner(GF), Armin Schleicher(Technischer Leiter)
Tag der Abnahme: 20.03.2025	Abgenommene Version (optional): Version 2.0

Bei der heute erfolgten gemeinsamen Abnahme wurde festgestellt, dass sich die erbrachte Leistung bis auf die nachfolgend bezeichneten Mängel (lt. Fehlerklassenbeschreibung siehe Anhang), im vertragsgemäß zu erstellenden Zustand befand.

Mängel / Fehler (Zusammenfassung)	Behebungsfrist	Klasse

Bemerkungen und Vorbehalte:

Zur vollen Zufriedenheit des Auftragnehmers in hoher Eigenverantwortung und professioneller Arbeitsweise realisiert.

Gesamtprodukt abgenommen (zutreffendes ankreuzen):

<input checked="" type="checkbox"/> JA	EINGESCHRÄNKTE	NEIN
--	----------------	------

20.3.25

Datum & Unterschrift (Auftraggeber)

06.03.25

Datum & Unterschrift (Auftragnehmer)

**Erläuterung Fehlerklassen:** Im Zuge der Abnahme werden eventuell bestehende Mängel in Fehlerklassen eingeteilt.

**Fehlerklasse 1:** Die zweckmäßige Nutzung (wirtschaftlich sinnvolle Nutzung) ist durch solche Fehler nicht möglich, unzumutbar eingeschränkt oder nicht möglich.

**Fehlerklasse 2:** Die zweckmäßige Nutzung ist nicht so weit beeinträchtigt, dass das System nicht dennoch verwendet werden könnte, allenfalls unter Einbeziehung zwischenzeitlichen Fehlerumgehungsrouter. Diese Fehler werden, so weit wie möglich, während des Abnahmetests behoben.

**Fehlerklasse 3:** Die zweckmäßige Nutzung ist durch diesen Fehler nicht oder nur unwesentlich eingeschränkt.

Die Zuordnung der Fehler in eine der obigen Fehlerklassen erfolgt einvernehmlich zwischen den Vertragspartnern.

Bei Fehlern der Fehlerklasse 1 handelt es sich um "erhebliche Abweichungen", bei Fehlern der Fehlerklasse 2 und 3 um "unerhebliche Abweichungen". Bei erheblichen Abweichungen wird der Test unmittelbar nach Behebung der Abweichung durchgeführt.

Eine unerhebliche Abweichung berechtigt den Auftraggeber nicht zur Verweigerung der Abnahme. Nach der Abnahme verbleibende Fehler aller Fehlerklassen werden im Rahmen der Gewährleistung gemäß einem gemeinsam zu erstellenden Zeitplan behoben.

Treten im Rahmen des Funktionstests keine wesentlichen Fehler (Fehlerklasse 1) auf, gilt das abzunehmende Projekt als abgenommen.

Bei Auftreten von wesentlichen Mängeln (Fehlerklasse 1) ist nach Korrektur der Funktionstests zu wiederholen.

**Liste der durchgeführten Abnahmetestfälle:**

Testfalldetails: siehe Testdokumentation bzw. Testfallbeschreibungstabelle

Nr.	Teil/Anw./Akteur	Bezeichnung, kurze Erläuterung	Abn. OK/ FKL.	Kommentare (Anm., Beschreibung Abw., Behebungsfrist)
1	Team	Statische Meeting URL	ok	
2	Team	Warteraum	ok	
3	Team	Verwalteter Gastzugang	ok	
4	Team	Videokonferenz	ok	
5	Aufzeichnungen	Transkription des Calls	ok	Aufzeichnung des Transkripts in die DB wäre wünschenwert. Ehemalige Übermittlung am Ende des Calls an ChatGPT statt laufend. Anforderung war vom PAG allerdings optional.
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				



# Projekthandbuch

IOE-Videocall

Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz  
2024/25

## Inhaltsverzeichnis

1. Projektantrag: .....	4
2. Projektzielplan .....	6
2. Vor und Nachprojektphase .....	7
Vorprojektphase .....	7
Nachprojektphase .....	8
3. Projektumwelt-Analyse.....	9
4. Vorgehensmodell .....	10
5. Taskbeschreibung (Sprint 6, letzter Sprint) .....	12
6. Projektsprintplan .....	13
7. Projektpersonaleinsatzplan .....	14
8. Projektkommunikationsstrukturen .....	15
9. Projekt-Spielregeln .....	17
10. Projektrisikoanalyse .....	18
11. Projektdokumentation .....	20

**Ansprechpartner**

Name	Abteilung	Projektrolle	Telefon Nr.	E-Mail
Bastian Seidl	HIF	Projektleiter	+43 676 4628049	SEI20375 @spengergasse. at
Louis Muhr	HIF	Projektmitglied	+43 670 6043311	Muh22378 @spengergasse. at
Philip Schrenk	HIF	Projektmitglied	+43 681 10280618	Sch2238 @spengergasse. at
Max Schwarz	HIF	Projektmitglied	+43 676 9033726	SCH22384 @spengergasse. at
Andres Zöttl	HIF	Projektbetreuer	-	zoettl @spengergasse. at
Hannes Baumgartner	-	Projektauftraggeber	-	hb@naos.studio

## 1. Projektantrag:

Louis Muhr, Max Schwarz, Bastian Seidl, Philip Schrenk

<b>Thema</b>	Webanwendung für Videocalls zur Verwaltung, Dokumentation und Optimierung von Tutor-Schüler-Meetings.
<b>Team</b>	Bastian Seidl (Leader): Max Schwarz (Member): Louis Muhr (Member): Philip Schrenk (Member):
<b>Startprozess</b>	In der Welt der digitalen Kommunikation gibt es zahlreiche Videokonferenz-Apps. Jedoch gibt es bisher keine spezialisierte Lösung, die gezielt auf die Bedürfnisse von Tutoren und Schülern zugeschnitten ist. Mit unserem Webvideocall-Tool wird es möglich sein, Meetings effizient zu verwalten, indem Gäste zunächst in einem Warteraum landen und Hosts volle Kontrolle über den Zugang haben. Zusätzlich sorgen Live-Transkriptionen und KI-gestützte Zusammenfassungen für eine optimale Dokumentation der Gespräche.
<b>Ziele</b>	Das Ziel dieses Projekts ist die Entwicklung eines benutzerfreundlichen Webvideocall-Tools, das speziell auf die Bedürfnisse von Tutoren und Schülern zugeschnitten ist. Die Anwendung soll eine kontrollierte Teilnahme an Meetings ermöglichen, indem Gäste zunächst in einem Warteraum warten und von den Hosts gezielt eingelassen werden.

	Zudem werden alle grundlegenden Videokonferenzfunktionen wie Mikrofon- und Kamera-Steuerung sowie Bildschirmfreigabe integriert. Durch die Implementierung einer Live-Transkription und KI-gestützter Zusammenfassungen soll eine effiziente Dokumentation der Gespräche gewährleistet werden. Moderne Technologien wie Next.js, Express.js und das AWS Chime SDK sorgen für eine stabile, sichere und skalierbare Lösung.
<b>Main Tasks</b>	Bastian Seidl: <ul style="list-style-type: none"><li>• Backend Model</li><li>• Services</li></ul> Philip Schrenk: <ul style="list-style-type: none"><li>• Frontend</li><li>• Einbindung der Video-Call Komponenten</li><li>• Feste Meeting URLs</li></ul> Maximilian Schwarz: <ul style="list-style-type: none"><li>• Transkription mit KI</li></ul> Louis Muhr: <ul style="list-style-type: none"><li>• Frontend Unterstützung</li><li>• Unit tests</li></ul>
<b>Risiken</b>	<ul style="list-style-type: none"><li>• Fehlendes Wissen</li><li>• Neue Technologie</li><li>• Zeit</li></ul>
<b>Meilensteine</b>	15.10.24 - Architekturplanung und Festlegung der Anforderungen abgeschlossen 15.11.24 - Implementierung des Frontends und der Benutzeroberfläche erstellt 31.12.24 - Backend-Entwicklung und Datenverwaltung durchgeführt 31.01.25 - Integration der KI-Transkription und -Analyse erledigt 15.03.25 - End-to-End-Tests und Optimierung abgeschlossen 09.04.24 - Projektabgabetermin

--	--

## 2. Projektzielplan

Zielart	Projektziele
Ziele:	<ul style="list-style-type: none"><li>• Entwicklung eines Webvideocall-Tools, das speziell für Tutoren und Schüler optimiert ist.</li><li>• Implementierung einer Warteraum-Funktion, sodass Hosts gezielt Gäste ins Meeting lassen können.</li><li>• Bereitstellung grundlegender Videokonferenzfunktionen wie Mikrofon- und Kamera-Steuerung sowie Bildschirmfreigabe.</li><li>• Integration einer Live-Transkription und KI-gestützter Zusammenfassungen zur effizienten Dokumentation von Meetings.</li><li>• Nutzung moderner Technologien (Next.js, Express.js, AWS Chime SDK) für eine stabile, sichere und skalierbare Lösung.</li><li>• Gewährleistung einer intuitiven und einfachen Benutzerführung für alle Nutzergruppen.</li></ul>
Nicht-Ziele:	<ul style="list-style-type: none"><li>• Entwicklung einer öffentlichen oder allgemein zugänglichen Videokonferenzplattform.</li><li>• Bereitstellung einer komplexen Rechte- und Rollenverwaltung über die Host- und Gast-Rollen hinaus.</li><li>• Erstellung einer mobilen App – die Anwendung bleibt eine Weblösung.</li><li>• Langfristige Speicherung oder Verwaltung von Meeting-Aufzeichnungen – das System dient primär der Live-Kommunikation.</li></ul>

## 2. Vor und Nachprojektphase

### Vorprojektphase

#### 1. Problemdefinition und Zielsetzung:

In Zusammenarbeit mit dem Institute of Entrepreneurship (Tochterfirma von Naos GmbH) wurde der Bedarf für ein maßgeschneidertes Videokonferenz-Tool identifiziert. Ziel war es, eine Plattform zu schaffen, die den spezifischen Anforderungen von Tutoren (Hosts) und Schülern (Gästen) gerecht wird. Dabei sollten eine intuitive Bedienung, Zuverlässigkeit und zusätzliche Funktionen wie Live-Transkription und Zusammenfassung im Fokus stehen.

#### 2. Anforderungsanalyse:

Es wurden Interviews oder Workshops mit Stakeholdern (z. B. Tutoren und Vertretern des Instituts) durchgeführt, um die Anforderungen zu spezifizieren. Daraus ergaben sich zentrale Features wie die statische Meeting-URL für Hosts, ein Warteraum für Gäste, Steuerungsmöglichkeiten für Tutoren sowie die Integration von Videokonferenz-Standardfunktionen und KI-gestützter Transkription.

#### 3. Technologie- und Machbarkeitsstudie:

Die Wahl der Technologien (Next.js für das Frontend, Express.js für das Backend und AWS Chime SDK für die Videofunktionalität) wurde getroffen, basierend auf ihrer Skalierbarkeit, Flexibilität und der Möglichkeit, Echtzeit-Kommunikation effizient umzusetzen. Zusätzlich wurde die Integration der OpenAI-API für die Live-Transkription und Zusammenfassung evaluiert.

#### 4. Projektplanung:

Ein Zeitplan wurde erstellt, der Meilensteine wie Prototyp-Entwicklung, Testphase und Deployment umfasste. Aufgaben wurden zwischen Teammitgliedern aufgeteilt (falls es ein Teamprojekt ist), und Ressourcen (z. B. AWS-Konten, Entwicklungsumgebungen) wurden bereitgestellt.

**5. Risikoanalyse:**

Potenzielle Herausforderungen wie Latenzprobleme bei der Videoverbindung, Datenschutzanforderungen (insbesondere bei der Speicherung von Transkripten) und die Stabilität der AWS Chime SDK wurden identifiziert und Maßnahmen zur Risikominimierung geplant.

## Nachprojektphase

**1. Testen und Qualitätssicherung:**

Nach der Entwicklung wurde das Tool in einer Testumgebung mit echten Nutzern (Tutoren und Schülern) evaluiert. Funktionen wie die Warteraumsteuerung, die Stabilität der Videoverbindung und die Genauigkeit der Live-Transkription wurden geprüft. Feedback wurde gesammelt, um letzte Anpassungen vorzunehmen.

**2. Deployment und Einführung:**

Das Webvideocall-Tool wurde auf einer produktiven AWS-Infrastruktur bereitgestellt. Tutoren erhielten ihre statischen Meeting-URLs, und Schüler wurden über den Zugang informiert. Eine kurze Schulung oder Dokumentation wurde erstellt, um die Nutzung zu erleichtern.

**3. Auswertung und Erfolgsmessung:**

Nach der Einführung wurde die Nutzung des Tools analysiert. Metriken wie die Anzahl der durchgeführten Meetings, die Zufriedenheit der Tutoren und Schüler (z. B. durch Umfragen) sowie die Qualität der automatischen Zusammenfassungen wurden bewertet, um den Erfolg des Projekts zu messen.

**4. Wartung und Optimierung:**

Basierend auf Nutzerfeedback und technischen Rückmeldungen (z. B. Performance-Probleme oder Verbesserungsvorschläge) wurden Updates geplant. Beispielsweise könnten zusätzliche Features wie eine Chatfunktion oder eine bessere Anpassung der Transkription an Fachbegriffe in Betracht gezogen werden.

**5. Langfristige Perspektive:**

Das Tool soll als fester Bestandteil der Arbeitsprozesse des Institute of Entrepreneurship etabliert werden. Eine Skalierung für weitere Tochterfirmen von Naos GmbH oder eine Anpassung für andere Zielgruppen (z. B. Unternehmenskunden) könnte in Zukunft untersucht werden.

### 3. Projektumwelt-Analyse

Umwelten	Beziehung	Maßnahmen
Projektauftraggeber: Hannes Baumgartner & Armin Schleicher	Positiv: immer hilfsbereit, froh über ein junges, motiviertes Team.	Regelmäßige Videocalls per Teams. Halfen bei Umsetzung einiger Technologien.
Betreuungslehrer: Andres Zöttl	Positiv: ist zufrieden mit dem Team.	Informierte sich regelmäßig über den Stand des Projektes. Gab Tipps und half mit seiner Vorerfahrung.
Projektleiter: Bastian Seidl	Positiv: Sehr guter Backend-Developer. Nutzte seine Skills für die Entwicklung des Backends.	Regelmäßige Sitzungen mit dem Projektteam, um den Fortschritt abzustimmen und zeitnahe Aufgaben zu besprechen und zu verteilen
Projektmitglied 1: Philip Schrenk	Positiv: Nutze seine Bekanntschaft mit den Partnern der Firma und übernahm die Kommunikation und sehr große Teile des Frontends.	Regelmäßige Sitzungen mit dem Projektteam, um den Fortschritt abzustimmen und zeitnahe Aufgaben zu besprechen und zu verteilen
Projektmitglied 2: Louis Muhr	Positiv: Nutzte seine Testkenntnisse, um die Qualität des Codes zu gewährleisten.	Regelmäßige Sitzungen mit dem Projektteam, um den Fortschritt

		abzustimmen und zeitnahe Aufgaben zu besprechen und zu verteilen
Projektmitglied 3: Max Schwarz	Positiv: Nutze seine Wissbegierde und Disziplin für die Einbindung und Umsetzung der Transkriptionskomponenten.	Regelmäßige Sitzungen mit dem Projektteam, um den Fortschritt abzustimmen und zeitnahe Aufgaben zu besprechen und zu verteilen

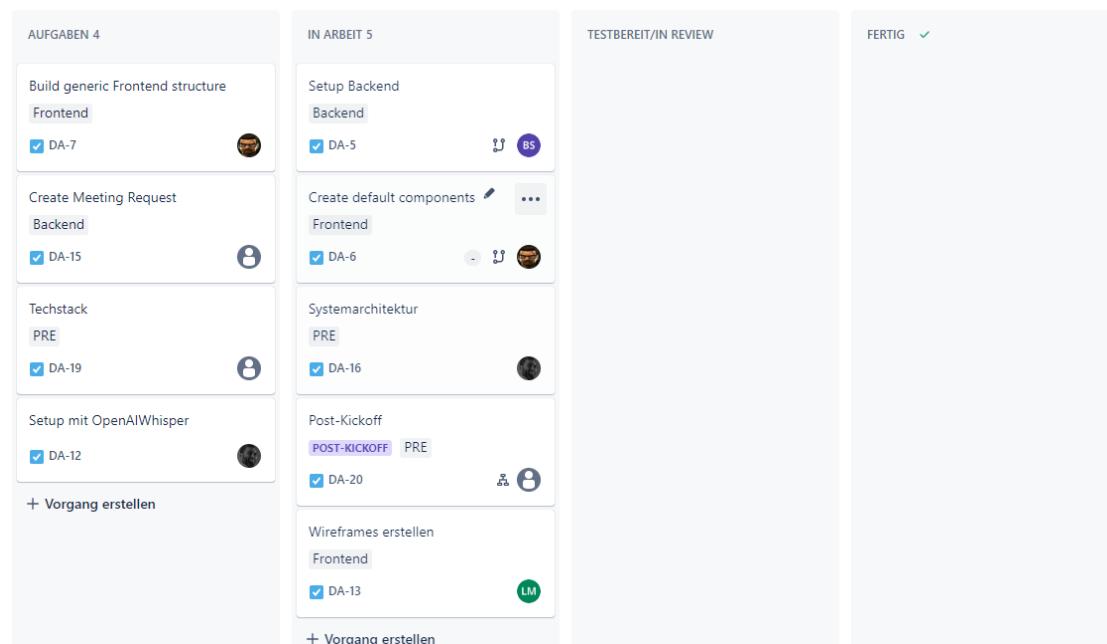
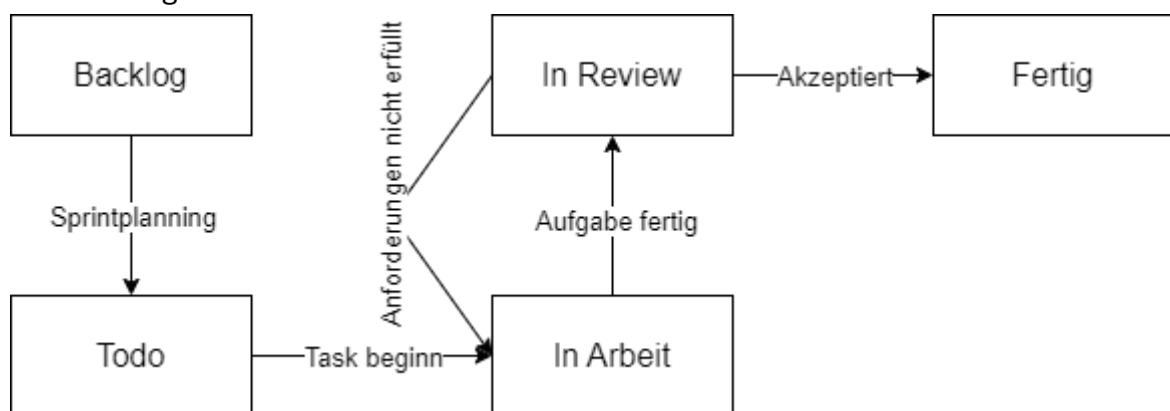
## 4. Vorgehensmodell

Das folgende Bild zeigt ein beispielhaftes Kanban-Board und veranschaulicht den Workflow, den wir in unseren Projekten verwenden. Das Board ist in mehrere Spalten unterteilt, die den Status eines Tasks visualisieren und den Fortschritt von Aufgaben im Team transparent machen.

1. **Backlog:** Zunächst werden neue Tasks im Backlog erstellt. Dies ist die Sammlung aller potenziellen Aufgaben und Ideen, die später umgesetzt werden sollen. Hier sammeln wir alles, was ansteht, jedoch noch nicht priorisiert oder für einen Sprint vorgesehen ist.
2. **To Do:** Nach dem Sprint Planning befinden sich die ausgewählten Tasks zunächst in der Spalte "To Do". Hier liegen alle Aufgaben, die im aktuellen Sprint bearbeitet werden sollen, aber noch nicht gestartet wurden.
3. **In Arbeit:** Sobald ein Teammitglied mit der Bearbeitung eines Tasks beginnt, wird dieser in die Spalte „In Arbeit“ verschoben. Hier wird die eigentliche Arbeit an den Aufgaben erledigt, wobei jedes Teammitglied an seinen individuellen Tasks arbeitet. Diese Spalte bietet eine Übersicht darüber, wer gerade woran arbeitet.
4. **In Review:** Sobald ein Task fertiggestellt ist, wird er in die Spalte „In Review“ verschoben. In dieser Phase überprüft ein anderes Teammitglied die geleistete Arbeit, um sicherzustellen, dass die Qualität stimmt und alle Anforderungen erfüllt sind. Das Review-Prozess dient nicht nur der Fehlerfindung, sondern auch der Sicherstellung von Best Practices und der Einhaltung von Standards.

**5. Fertig:** Nach erfolgreicher Überprüfung wird der Task in die Spalte „Fertig“ verschoben und gilt damit als abgeschlossen. Sollte es während der Review-Phase allerdings Änderungswünsche oder Korrekturbedarf geben, wird der Task wieder zurück in die Spalte „In Arbeit“ verschoben, damit die nötigen Anpassungen vorgenommen werden können.

Dieses Kanban-Board ermöglicht es uns, jederzeit den Status aller Aufgaben im Blick zu behalten und effizient als Team zusammenzuarbeiten. Durch die klare Visualisierung und den definierten Workflow verbessern wir die Transparenz, Kommunikation und Nachverfolgbarkeit innerhalb des Teams.



## 5. Taskbeschreibung (Sprint 6, letzter Sprint)

### Transkription Zusammenfassung

Wird überprüft + Hinzufügen

#### Beschreibung

Teste die Transkription und mache eine Zusammenfassung. Schau ob es vernünftig ist.

Aktionen

#### Details

Zugewiesene Person

Max Schwarz  
Mir zuweisen

### Transkription im Frontend

Wird überprüft + Hinzufügen

#### Beschreibung

Beschreibung bearbeiten

Aktionen

#### Details

Zugewiesene Person

Max Schwarz  
Mir zuweisen

### Readme-Anleitung

Fertig + Hinzufügen

#### Beschreibung

Anleitung zum Starten: in die project root README (die auf gitlab sichtbar ist)

1. Docker aufsetzen (Basti fragen ka wie geht)
2. AWS Command Line installieren
3. Docker starten über das Panel
4. Backend starten (cd backend && npm run dev)
5. Frontend starten (cd ioe-video call && npm run dev)
6. Mit postman einen Host erstellen und dann dessen GUID holen
7. Frontend auf den link /room/{guid}?PARAMETERS (Voller Link steht als Kommentar in der datei [uuid].tsx!!!)

#### Details

Zugewiesene Person

Louis Muhr  
Kein Wert

Stichwort

Kein Wert

Priorität

Kein Wert

Übergeordnet

Ohne

### InputSelections

Fertig + Hinzufügen

#### Beschreibung

Mic, Cam, Output Selections

#### Details

Zugewiesene Person

Philip Sch  
Mir zuweisen

### Warteraum Bugfix

Fertig + Hinzufügen

#### Beschreibung

Beschreibung bearbeiten

Aktionen

#### Details

Zugewiesene Person

Bastian Seidl  
Mir zuweisen

## 6. Projektsprintplan

Sprint Nr.	Bezeichnung	Termin	Nicht fertiggestellte Tasks
1	Sprint 1 abgeschlossen	15.11.2024	/
2	Sprint 2 abgeschlossen	6.12.2024	
3	Sprint 3 abgeschlossen	27.12.2024	
4	Sprint 4 abgeschlossen	24.01.2025	
5	Sprint 5 abgeschlossen	14.02.2025	
6	Sprint 6 abgeschlossen	07.03.2025	

## 7. Projektpersonaleinsatzplan

Sprint	Ressourcenart	Planmenge in PH	Adaptierte Planmenge in PH	Istmenge in PH	Abweichung in PH
Sprint 1	Zeit	80			
Sprint 2	Zeit	95			
Sprint 3	Zeit	95			
Sprint 4	Zeit	95			
Sprint 5	Zeit	95			
Sprint 6	Zeit	90			

## 8. Projektkommunikationsstrukturen

Bezeichnung	Ziele, Inhalte	Teilnehmer	Ort
Besprechung Projektanforderungen	<b>Vorstellung des Diplomprojekts, Anforderungen definieren, Anforderungen des Kunden besprechen , To-Dos abstimmen</b>	Team: Bastian Seidl, Philip Schrenk Louis Muhr, Max Schwarz,	Schule
Besprechung Kooperationsvertrag	<b>Erstellung und Besprechung des Kooperationsvertrags, Klären von Verantwortlichkeiten, Rechtliche Aspekte, Festlegung von Meilensteinen</b>	Naos GmbH, Bastian Seidl, Philip Schrenk Louis Muhr, Max Schwarz,	MS Teams

Systemarchitektur-Sitzung	<b>Systemarchitektur entwerfen, Live-Transkription und OpenAI-Zusammenfassung planen, Datenfluss zwischen Hosts und Gästen definieren, Technische Anforderungen für Videokonferenzfunktionen festlegen, Datenschutzkonzept für Transkription erstellen</b>	Bastian Seidl, Philip Schrenk Louis Muhr, Max Schwarz,	Schule
Sprint Review	<b>Ergebnisse und Teilprodukte vorstellen, Live-Transkription und OpenAI-Zusammenfassung testen, Feedback vom Kunden einholen, Nächste Schritte planen</b>	Anna Müller, Lukas Schmidt, Julia Wagner, Max Klein, Vertreter von Naos GmbH	MS Teams

## 9. Projekt-Spielregeln

Die Sitzungszeit wird eingehalten.

- o Alle Teammitglieder erscheinen pünktlich.
- o Alle Anwesenden müssen die benötigten Unterlagen dabeihaben.
- o Eine Person muss mitprotokollieren und dieses Protokoll an die Mitglieder verschicken.
- Wir lassen einander ausreden und nehmen Bezug auf die Beiträge anderer.
- Wir sind offen und konstruktiv.
- Jeder muss bei den Dokumenten (PHB, Lastenheft, Pflichtenheft, ...) etwas beitragen.
- Wir erledigen unsere Aufgaben bis zum gemeinsam festgelegten Termin.
- Bei Unstimmigkeiten bezüglich Entscheidungen trifft die Projektleiterin diese.
- Die gemeinsam festgelegten Methoden unseres Vorgehensmodells werden eingehalten (siehe Kapitel 4)
- Dokumente werden einheitlich benannt und in den richtigen Ordner im Repository abgelegt

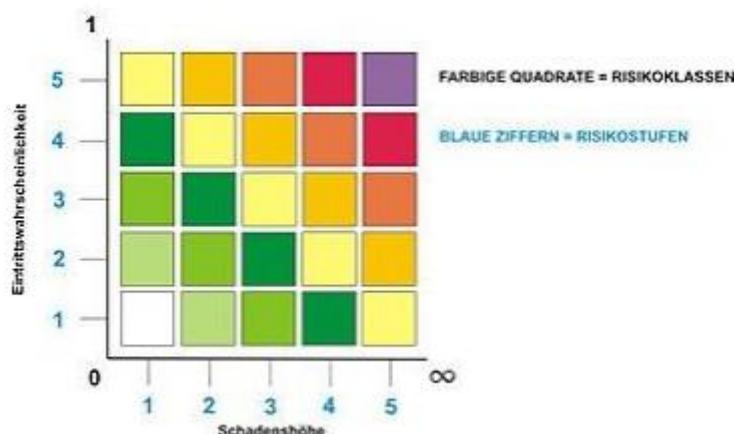
## 10. Projektrisikoanalyse

Risikobeschreibung	Priorität	Risikokosten	Wahrscheinlichkeit	Risikowert	Verzögerung	Maßnahmen	Risikominimierung
Großer Arbeitsaufwand aufgrund von anderen schulischen Verpflichtungen	Grün	50€	15%	7.5€	2	Mithilfe von länger andauernden Sprints ermöglichen wir uns die Aufgaben auf einen längeren Zeitraum aufzuteilen.	/

Technische Risiken Probleme bei der Integration der AWS Chime SDK	Gelb	100€	30%	30€	3	Frühzeitige Tests in einer Sandbox-Umgebung, Schulung des Teams, Unterstützung durch AWS-Support.	/
Summe	-	150€	45%	37.5€	5		/

Weitere Risiken werden zusammen mit den Tasks geplant.

### Risikomatrix:



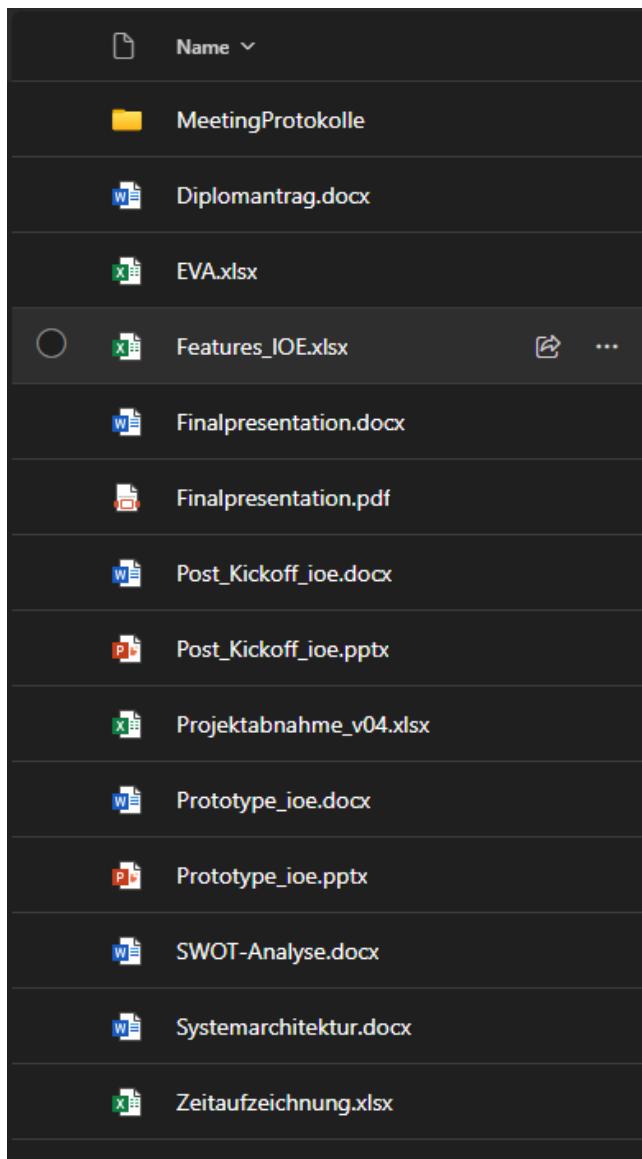
- weiß: ist vernachlässigbar
- hellgrün: sollte immer mal wieder überprüft werden, ist aber vernachlässigbar
- mittelgrün: sollte immer mal wieder überprüft werden
- dunkelgrün: beträchtliches Risiko, muss im Auge behalten werden
- hellgelb: eindeutiges Risiko, muss im Auge behalten werden und regelmäßig überprüft werden
- orange: großes Risiko, muss zwingend oft überprüft werden
- orange-rot: Projektgefährdendes Risiko
- rot: projektgefährdendes Risiko, Projekt sollte möglicherweise beendet werden

- lila: Projekt ist zu risikoreich, sollte beendet werden

Die Eintrittswahrscheinlichkeit wird in 20%-Schritten bestimmt.

## 11. Projektdokumentation

Alle Dokumente werden auf MS Teams gespeichert. Auf diese Dokumente haben nur die Teammitglieder Zugriff.





# Anforderungsspezifikationen

## IOE-Videocall

Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz  
2024/25

## Inhaltsverzeichnis

1. Beschreibung des ist- und Sollzustandes .....	3
1.1 Istzustand .....	3
1.2 Sollzustand .....	3
1.3 Zielbestimmung.....	3
1.4 Projekteinsatz.....	4
1.4.1 Nutzen .....	4
1.4.2 Anwendungsbereich .....	4
1.4.3 Zielgruppen .....	4
2. Funktionale Anforderungen.....	4
3. Nicht funktionale Anforderungen.....	4
4. Produktdaten .....	5
5. Systemarchitektur .....	6
6. Schnittstellen.....	6
7. Lieferumfang.....	7
8. Abnahmekriterien.....	7

## 1. Beschreibung des ist- und Sollzustandes

### 1.1 Istzustand

Die Firma Naos GmbH organisiert regelmäßig Videokonferenzen mit ihren Kunden. Derzeit wird hierfür allgemeine Videokonferenzlösungen verwendet, die nicht optimal auf die Anforderungen dieses Szenarios abgestimmt sind. Dies führt zu mehreren Einschränkungen wie Umständliche Meeting-Verwaltung, Eingeschränkte Funktionalität und Mangelnde Benutzerfreundlichkeit. Diese Nachteile führen zu einem ineffizienten Workflow, erhöhen den organisatorischen Aufwand und beeinträchtigen die Qualität der Kommunikation. Ein maßgeschneidertes Webvideocall-Tool soll diese Probleme beheben und die Prozesse optimieren.

### 1.2 Sollzustand

Das Ziel ist die Entwicklung einer Webapplikation, die eine benutzerfreundliche Plattform für Videokonferenzen bietet. Sie ermöglicht Meetings über eine statische Meeting-URL zu verwalten, und Usern, über einen Warterraum teilzunehmen, inklusive Videokonferenzfunktionen, Live-Transkription und automatischer Zusammenfassungen. Durch die intuitive Bedienung können Tutoren und Schüler die Anwendung ohne lange Einarbeitung nutzen, was den Aufwand reduziert und die Effizienz steigert.

### 1.3 Zielbestimmung

Im Projekt wird angestrebt, eine webbasierte Videokonferenzplattform zu entwickeln, die speziell auf die Bedürfnisse von Naos GmbH zugeschnitten ist. Ziel ist es, die Organisation und Durchführung von Meetings zu vereinfachen, die Zugangskontrolle zu verbessern und durch zusätzliche Funktionen wie Live-Transkription und automatische Zusammenfassungen die Effizienz zu steigern.

### Soll Ziele

- Eine statische Meeting-URL für jeden Host wurde implementiert.
- Ein Warterraum für User, aus dem Hosts Teilnehmer zulassen können, wurde realisiert.
- Videokonferenzfunktionen (Mikrofon ein/aus, Audio ein/aus, Kamera ein/aus, Bildschirmübertragung ein/aus) wurden implementiert.
- Live-Transkription des Meetings wurde integriert.
- Automatische Zusammenfassung des Meetings mithilfe von OpenAI wurde implementiert.
- Eine gesicherte Verbindung über HTTPS wurde eingerichtet.

- Eine benutzerfreundliche Oberfläche, die eine schnelle Einarbeitung ermöglicht, wurde entwickelt.

#### Kann-Ziele

- Anpassbare Wartezeiten im Warteraum für Schüler.
- Anpassung der Transkriptionssprache

#### Nicht-Ziele

- Implementierung einer Chat-Funktion während des Meetings.
- Integration mit externen Kalendertools oder anderen Drittanbieter-Apps.

### 1.4 Projekteinsatz

#### 1.4.1 Nutzen

Der reine Nutzen dieses Projekts besteht darin, eine maßgeschneiderte Videokonferenzlösung bereitzustellen, die speziell auf die Bedürfnisse von Naos GmbH zugeschnitten ist. Durch die Entwicklung eines Webvideocall-Tools mit Next.js, Express.js und der AWS Chime SDK wird die Organisation und Durchführung von Meetings vereinfacht. Funktionen wie ein Warteraum für Zugangskontrolle, Live-Transkription und automatische Zusammenfassungen mit OpenAI optimieren die Effizienz der Nachbereitung und verbessern die Lernerfahrung.

#### 1.4.2 Anwendungsbereich

Angewendet wird das Produkt dann von Mitarbeitern der Naos GmbH, um Kunden eine Plattform für Meetings bereitzustellen.

#### 1.4.3 Zielgruppen

Verwendet wird das Produkt von Mitarbeitern der Naos GmbH und den Kunden

## 2. Funktionale Anforderungen

- Feste Meeting-URLs für Host
- Warteraum für Gäste: Gäste betreten nach dem Zugriff auf die Meeting-URL einen Warteraum und bleiben dort, bis der Host sie genehmigt.
- Vom Host verwalteter Gastzugang: Hosts können Gäste innerhalb der Benutzeroberfläche genehmigen oder ablehnen und so die volle Kontrolle darüber haben, wer dem Meeting beitreten darf.
- Videokonferenz: Die Anwendung bietet Grundlegende Meeting-Funktionen, und der Host kann ein Meeting starten und beenden
- Aufzeichnungen mit Transkription und Zusammenfassung: Meetings werden automatisch aufgezeichnet und in AWS S3 gespeichert; GPT analysiert die Transkripte, um Meeting-Zusammenfassungen, wichtige Erkenntnisse und Handlungspunkte zu generieren

### 3. Nicht funktionale Anforderungen

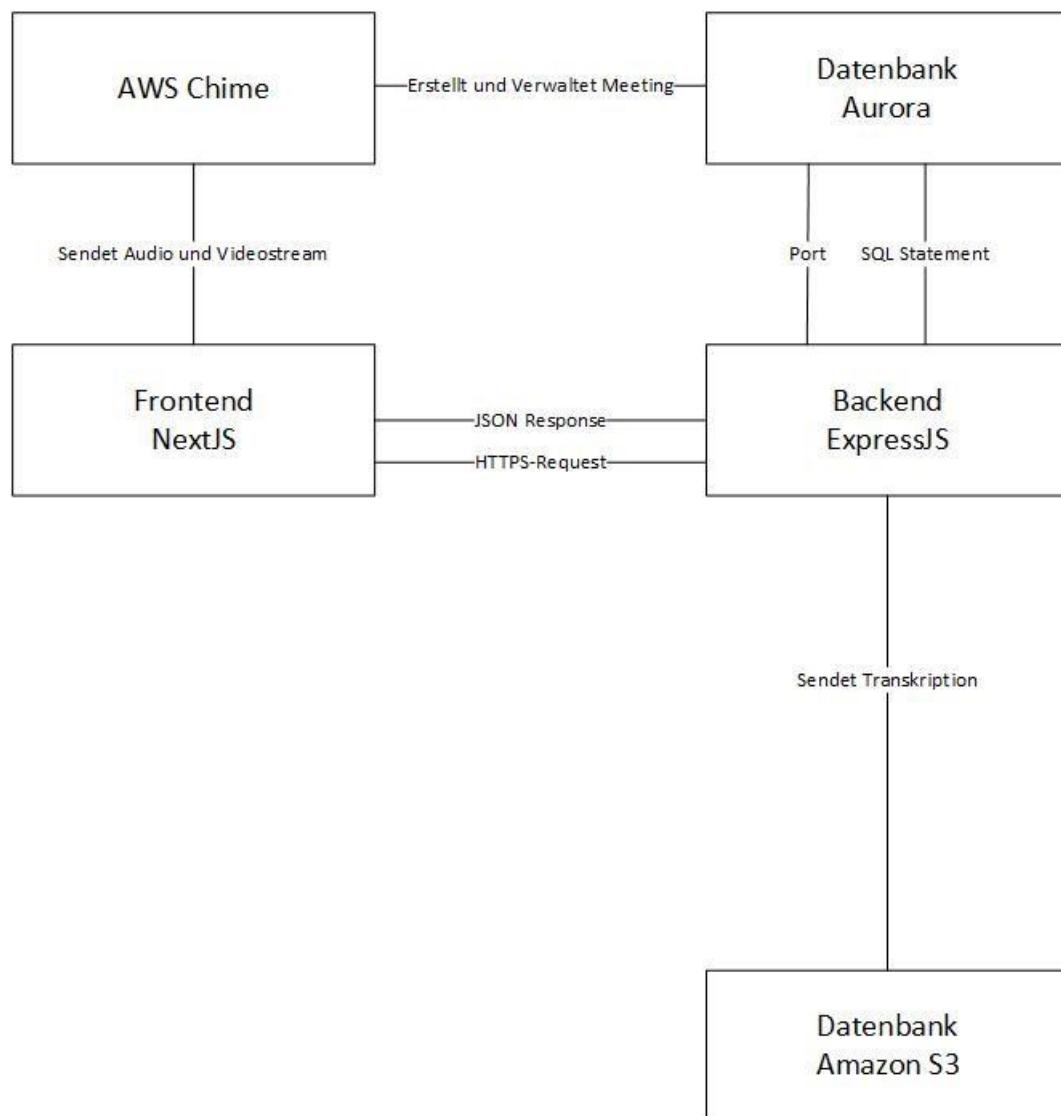
- Die Video- und Audioübertragung sollte mit minimaler Verzögerung erfolgen, um eine flüssige und natürliche Kommunikation zu ermöglichen.
- Alle Daten während der Übertragung verschlüsselt werden, um Vertraulichkeit und Datenschutz zu gewährleisten.
- Die Benutzeroberfläche sollte einfach und verständlich sein, sodass Nutzer sie ohne umfangreiche Schulung bedienen können.
- Das System sollte eine wachsende Anzahl von gleichzeitigen Nutzern und Meetings unterstützen, ohne dass die Leistung leidet.
- Das Tool sollte mit den gängigsten Webbrowsern kompatibel sein.
- Nur die unbedingt notwendigen Daten sollten gesammelt und gespeichert werden, um die Privatsphäre der Nutzer zu schützen.
- Die Speicherung und Verarbeitung von Daten muss den geltenden Datenschutzbestimmungen entsprechen
- Das Tool sollte nahezu jederzeit verfügbar sein um Unterbrechungen während Meetings zu vermeiden.

### 4. Produktdaten

Auf dem Amazon S3 Server werden die durch GPT Zusammengefassten Transkriptionen gespeichert. Nach Ablauf einer bestimmten Zeit wie im Vertrag mit den Kunden beschrieben werden diese dann wieder automatisch gelöscht.

Andere Daten wie Benutzerdaten werden nicht gespeichert und die Live Transkriptionen werden nur zur Verarbeitung benutzt und auch nicht gespeichert.

## 5. Systemarchitektur



## 6. Schnittstellen

Frontend <--> Backend

- Backend API
  - Port: 3000
  - Protokoll: HTTP (JSON über REST)

Backend <--> AWS Chime

- AWS Chime SDK
  - Port: 443
  - Protokoll: WebRTC (für Video- und Audiostreams)

Backend <--> Amazon S3

- Amazon S3 (Dateispeicherung für Transkriptionen & Zusammenfassungen)
  - Port: 443
  - Protokoll: HTTPS (REST API)

Backend <--> AWS Lambda

- AWS Lambda
  - Port: 443
  - Protokoll: HTTPS (REST API)

## 7. Lieferumfang

Der Auftraggeber erhält zur Projektabnahme den Link zum Git-Repository per E-Mail zugeschickt

## 8. Abnahmekriterien

Die Abnahmekriterien gelten als erfüllt, wenn alle Ziele erfüllt wurden.



# Use-Spezifikation

## IOE-Videocall

Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz  
2024/25

## Inhaltsverzeichnis

1. Zusammenfassung.....	3
2. Nutzungskontext .....	4
3. Spezifikation Gebrauchstauglichkeit des Produkts .....	5
3.1 Workflow .....	5
3.2 Mockups .....	6
3.2.1 Der Warteraum.....	6
3.2.2 Meeting-View (mit Beitrittsanfrage) .....	7
3.2.3 Meeting-View mit Teilnehmern inkl. Kamera .....	8
3.2.4 Modal für Einstellungen.....	9
3.3 Design.....	10
3.4 Risikomanagement .....	11
4. Verifizierung Gebrauchstauglichkeit .....	12

## 1. Zusammenfassung

Da das Ziel ist, ein effizientes und sicheres Webvideocall-Tool zu entwickeln, wurde entschieden, dass die Use-Spezifikation entsprechend der Norm IEC 62366 umgesetzt wird, die bestrebt, einen gebrauchstauglichkeitsorientierten Entwicklungsprozess zu bestimmen.

Hierbei werden folgende Kapitel behandelt:

- Der Nutzungskontext, die Nutzer und das physische Prinzip des Produkts
- Die Spezifikation der Gebrauchstauglichkeit des Programms
- Das Verifizieren der Gebrauchstauglichkeit

## 2. Nutzungskontext

Im Falle des Webvideocall-Tools wird die Anwendung auf Servern im AWS-Ökosystem gehostet und ist im Rahmen des Institute of Entrepreneurship (Tochterfirma der Naos GmbH) erreichbar.

Da die Anwendung auf Servern im AWS-Ökosystem gehostet wird, kann sie nur im Netzwerk des Institute of Entrepreneurship genutzt werden. Das heißt, Laptops und Desktop-PCs, die sich im Netzwerk des Unternehmens befinden (Windows, macOS, Linux), können die Anwendung über gängige Browser (Google Chrome, Mozilla Firefox, Microsoft Edge) aufrufen.

Zugänglich ist die Anwendung für alle Mitarbeiter des Institute of Entrepreneurship, die in die Kategorien Tutoren (Hosts) und Schüler (Gäste) unterteilt sind. Tutoren nutzen die Anwendung, um Meetings zu erstellen und zu moderieren, während Schüler diesen Meetings beitreten. Die Anwendung ist darauf ausgelegt, den Anforderungen eines Bildungskontexts gerecht zu werden: Tutoren können Meetings über eine statische URL starten, Schüler treten diesen bei und werden zunächst in einen Warteraum geleitet, bevor sie zugelassen werden. Die Anwendung bietet Videokonferenzfunktionen wie Mikrofon an/aus, Kamera an/aus, Audio an/aus und Bildschirmübertragung an/aus. Zusätzlich wird der Anruf live transkribiert und mithilfe von OpenAI zusammengefasst.

Der Verwendungszweck des Webvideocall-Tools im Institute of Entrepreneurship ist die Bereitstellung einer sicheren und effizienten Plattform für Online- „Unterricht“. Tutoren können so „Schüler“ in Echtzeit unterrichten, während die Transkription und Zusammenfassung durch OpenAI die Nachbereitung und Dokumentation erleichtert. Die Anwendung ist nicht für die Nutzung außerhalb des Unternehmensnetzwerks vorgesehen, da sie speziell auf die Bedürfnisse des Institute of Entrepreneurship zugeschnitten ist.

### 3. Spezifikation Gebrauchstauglichkeit des Produkts

Dieses Kapitel beschreibt die Gebrauchstauglichkeit des Webvideocall-Tools. Ziel ist es, die Anforderungen für Tutoren und Schüler in einem Online-Unterrichtskontext zu erfüllen und klar definierte Prozesse zu etablieren.

#### 3.1 Workflow

Der Workflow beschreibt die Organisation von Online-Meetings im Institute of Entrepreneurship. Um die Usability im vorgesehenen Kontext zu gewährleisten, wurde ein AGFA-Modell erstellt (Akzeptanz, Gebrauchstauglichkeit, Funktionalität, Architektur). Der Workflow beinhaltet das Erstellen eines Meetings durch Tutoren, das Beitreten von Schülern und die Nutzung der Videokonferenzfunktionen. Die Benutzeroberfläche (UI) soll den Workflow optimieren.

##### 1. Meeting erstellen

Tutoren erstellen ein Meeting und erhalten eine statische URL.

##### 2. Schüler beitreten lassen

Schüler rufen die URL auf und landen im Warteraum.

##### 3. Zulassen durch Tutoren

Tutoren lassen Schüler aus dem Warteraum ins Meeting.

##### 4. Videokonferenz starten

Mikrofon, Kamera, Audio und Bildschirmübertragung werden genutzt.

##### 5. Live-Transkription und Zusammenfassung

Der Anruf wird live transkribiert und mit OpenAI zusammengefasst.

##### 6. Meeting beenden

Tutoren beenden das Meeting, und die Zusammenfassung wird gespeichert.

Hierbei ist zu beachten, dass die Funktionalitäten für Tutoren und Schüler unterschiedlich sind. Tutoren haben mehr Kontrolle (z. B. Schüler zulassen), während Schüler eingeschränkte Rechte haben (z. B. Warteraum).

## 3.2 Mockups

### 3.2.1 Der Warteraum



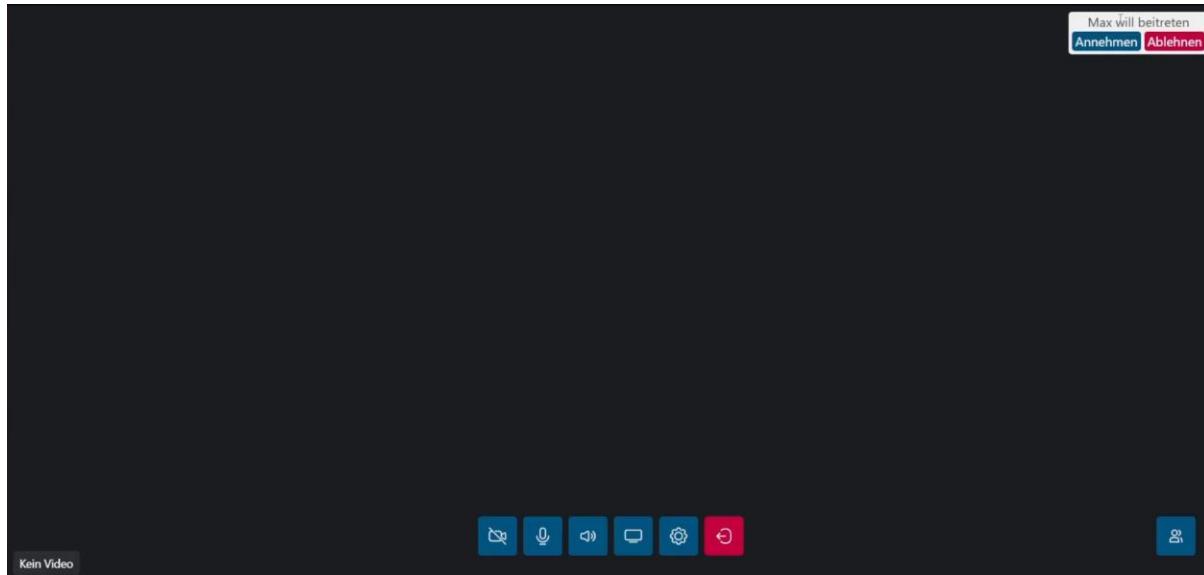
#### Bildbeschreibung:

Die Abbildung zeigt die Warteraum-Ansicht für Schüler (Gäste) nach dem Aufruf der statischen Meeting-URL. Ein Willkommensfenster mit der Überschrift „Willkommen Max“ begrüßt den Schüler und zeigt den Text „Senden Sie eine Beitreitsanfrage, welche von einem Host angenommen wird“. Darunter befindet sich ein blauer Button mit der Aufschrift „BEITRITSANFRAGE SENDEN“. Zusätzlich werden der Titel des Meetings („Test-Meeting“) sowie Start- und Endzeit („1. Dezember 2024 um 11:00 MEZ“ bis „1. Dezember 2024 um 11:00 MEZ“) angezeigt.

#### Erweiterte Erklärung:

Dies ist die Ansicht, die Gäste sehen, nachdem sie die statische Meeting-URL, die ihnen von einem Tutor (Host) bereitgestellt wurde, aufrufen. Die Seite ist unter /rooms/[Id] erreichbar, wobei [Id] die eindeutige Meeting-ID repräsentiert. Sobald ein Schüler die URL öffnet, wird er automatisch in den Warteraum weitergeleitet. Der Warteraum dient als Sicherheitsmechanismus, um sicherzustellen, dass nur autorisierte Teilnehmer am Meeting teilnehmen können. Der Gast sieht eine personalisierte Begrüßung (z. B. „Willkommen Max“), die auf den eingeloggten Benutzer hinweist, sowie den Titel des Meetings und die geplante Zeitspanne. Der Button „BEITRITSANFRAGE SENDEN“ ermöglicht es dem Schüler, eine Anfrage an den Tutor zu senden. Sobald die Anfrage gesendet wurde, ändert sich der Text des Buttons zu „Beitreitsanfrage gesendet“, und der Schüler wartet auf die Freigabe durch den Tutor. Diese Funktion stellt sicher, dass Tutoren die Kontrolle über die Teilnehmerliste behalten und unbefugte Zugriffe verhindert werden. Der Warteraum ist ein zentraler Bestandteil der Benutzererfahrung für Schüler und unterstützt die organisatorische Struktur des Institute of Entrepreneurship, indem er eine klare Trennung zwischen Hosts und Gästen gewährleistet.

## 3.2.2 Meeting-View (mit Beitrittsanfrage)



**Bildbeschreibung:** Die Abbildung zeigt die Ansicht eines Tutors (Hosts) im Meeting, bevor Schüler zugelassen wurden. Der Bildschirm ist größtenteils dunkel, da noch keine Kameras aktiv sind. Am oberen Rand befinden sich zwei Buttons: „Max will beitreten“ (blau) und „Ablehnen“ (rot). Am unteren Rand sind Steuerungselemente für die Videokonferenz sichtbar: Symbole für Mikrofon, Kamera, Bildschirmübertragung und das Beenden des Meetings.

**Erweiterte Erklärung:**

Dies ist die Ansicht, die ein Tutor (Host) sieht, nachdem er ein Meeting über die statische URL erstellt hat und ein Schüler (Gast) eine Beitrittsanfrage gesendet hat. Der Bildschirm zeigt zunächst keine Videoübertragung, da weder die Kamera des Tutors noch die eines Schülers aktiviert ist. Am oberen Rand des Bildschirms erscheint eine Benachrichtigung in Form eines blauen Buttons mit der Aufschrift „Max will beitreten“, was darauf hinweist, dass ein Schüler namens Max eine Beitrittsanfrage gesendet hat. Neben diesem Button gibt es einen roten „Ablehnen“-Button, der es dem Tutor ermöglicht, die Anfrage abzulehnen, falls der Schüler nicht zugelassen werden soll. Diese Funktion gibt Tutoren die volle Kontrolle über die Teilnehmerliste und erhöht die Sicherheit des Meetings. Am unteren Rand des Bildschirms befinden sich die Steuerungselemente der Videokonferenz: Symbole für das Ein- und Ausschalten des Mikrofons, der Kamera, der Bildschirmübertragung sowie ein rotes Symbol zum Beenden des Meetings. Diese Symbole sind intuitiv gestaltet, wobei ein durchgestrichenes Symbol (z. B. ein durchgestrichenes Mikrofon) anzeigt, dass die Funktion deaktiviert ist. Sobald der Tutor auf „Max will beitreten“ klickt, wird der Schüler in das Meeting aufgenommen, und die Videokonferenz kann beginnen. Diese Ansicht stellt sicher, dass Tutoren jederzeit den Überblick über anstehende Beitrittsanfragen behalten und die Meeting-Umgebung aktiv steuern können.

## 3.2.3 Meeting-View mit Teilnehmern inkl. Kamera



**Bildbeschreibung:** Die Abbildung zeigt ein aktives Meeting mit zwei Teilnehmern, deren Kameras eingeschaltet sind. Zwei Videofenster sind sichtbar: eines zeigt einen Schüler (Gast), das andere den Tutor (Host). Am unteren Rand befinden sich die Steuerungselemente für Mikrofon, Kamera, Bildschirmübertragung und das Beenden des Meetings. **Erweiterte Erklärung:**

Dies ist die Ansicht eines laufenden Meetings, nachdem ein Gast vom Tutor (Host) zugelassen wurde und beide ihre Kameras aktiviert haben. Die Benutzeroberfläche zeigt zwei Videofenster: eines für den Tutor und eines für den Schüler, wobei die Namen der Teilnehmer (z. B. „Tutor“) in den jeweiligen Fenstern angezeigt werden. Diese Ansicht demonstriert das responsive Layout der Anwendung, das sich automatisch an die Anzahl der Teilnehmer anpasst. Wenn weitere Schüler beitreten, werden zusätzliche Videofenster hinzugefügt, wobei das Layout dynamisch skaliert, um alle Teilnehmer sichtbar zu halten. Am unteren Rand der Ansicht befinden sich die Steuerungselemente, die sowohl für Tutoren als auch für Schüler verfügbar sind: Symbole zum Ein- und Ausschalten des Mikrofons, der Kamera, der Bildschirmübertragung sowie ein rotes Symbol zum Verlassen oder Beenden des Meetings. Während des Meetings wird der Anruf live transkribiert, und die Transkription kann über ein separates Fenster (nicht abgebildet) eingesehen werden. Nach Abschluss des Meetings wird die Transkription mithilfe von OpenAI zusammengefasst und den Teilnehmern zur Verfügung gestellt. Diese Ansicht stellt sicher, dass Tutoren und Schüler eine klare und intuitive Benutzererfahrung haben, die den Anforderungen eines Online-Unterrichts im Institute of Entrepreneurship entspricht. Die Möglichkeit, Kameras ein- und auszuschalten, ermöglicht Flexibilität und Privatsphäre für alle Beteiligten.

### 3.2.4 Modal für Einstellungen



**Bildbeschreibung:** Die Abbildung zeigt ein Modal-Fenster mit der Überschrift „EINSTELLUNGEN“. Es enthält Dropdown-Menüs für Kameraeinstellungen („Kameraqualität auswählen“, „INTEGRATED CAMERA (04270)“, „MIKROFON AUSWÄHLEN“, „Mikrofon (Realtek(R) Audio)“) und Lautsprechereinstellungen („LAUTSPRECHER AUSWÄHLEN“, „Standard – Lautsprecher (Realtek(R) Audio)“).

#### Erweiterte Erklärung:

Dieses Modal-Fenster wird angezeigt, wenn ein Benutzer (Tutor oder Schüler) auf das Einstellungssymbol klickt, das sich in der Steuerleiste am unteren Rand des Meeting-Bildschirms befindet. Das Modal ermöglicht es den Benutzern, ihre Audio- und Videoeinstellungen anzupassen, um eine optimale Meeting-Erfahrung zu gewährleisten. Unter der Überschrift „EINSTELLUNGEN“ sind drei Hauptbereiche aufgeführt: Kameraeinstellungen, Mikrofoneinstellungen und Lautsprechereinstellungen. Im Bereich „Kameraqualität auswählen“ können Benutzer die Auflösung ihrer Kamera anpassen (z. B. 720p, 1080p), um die Videoqualität an ihre Internetverbindung anzupassen. Das Dropdown-Menü „INTEGRATED CAMERA (04270)“ zeigt die verfügbaren Kameras an, die mit dem Gerät verbunden sind, und ermöglicht es dem Benutzer, zwischen verschiedenen Kameras zu wechseln (z. B. integrierte Kamera oder externe Webcam). Im Bereich „MIKROFON AUSWÄHLEN“ können Benutzer ihr bevorzugtes Mikrofon auswählen, wobei das Standardmikrofon (z. B. „Mikrofon (Realtek(R) Audio)“) voreingestellt ist. Ähnlich können Benutzer im Bereich „LAUTSPRECHER AUSWÄHLEN“ ihre bevorzugten Lautsprecher auswählen, wobei die Standardoption (z. B. „Standard – Lautsprecher (Realtek(R) Audio)“) voreingestellt ist. Nachdem die Einstellungen vorgenommen wurden, können Benutzer das Modal schließen, und die Änderungen werden sofort angewendet. Diese Funktion ist besonders nützlich, um technische Probleme während eines Meetings zu beheben, und stellt sicher, dass sowohl Tutoren als auch Schüler ihre Geräte optimal konfigurieren können, um an einem reibungslosen Online-Unterricht teilzunehmen.

### 3.3 Design

Hier geht es um die spezifischen Benutzeranforderungen. Anforderungen an das Design des Webvideocall-Tools sind die Umsetzung eines klaren und intuitiven Interfaces. Dazu gehört die Anzeige des Warteraums für Tutoren, die Steuerung der Videokonferenzfunktionen (Mikrofon, Kamera, etc.) und die Darstellung der Live-Transkription. Die Benutzeroberfläche nimmt sich ein Beispiel an der Benutzeroberfläche von *Google Meet*.

Die UI soll durch Buttons, Icons, Labels und klare Farbschemata (z. B. Grün für „an“, Rot für „aus“) intuitiv gestaltet sein.

### 3.4 Risikomanagement

Um bei dem Produkt systematische Fehlerquellen zu betrachten, wird eine FMEA als Risikomatrix durchgeführt. Die Risikomatrix ist so aufgebaut, dass sie auch im AGFA-Modell berücksichtigt wird.

Nr	Fehlerart	Fehlerursache	Fehlerfolgen	Auftreten (A)	Bedeutung (B)	Entdeckung (E)	RPZ (A * B * E)	Maßnahmen
1	Verbindungsaustritt	Instabile Internetverbindung	Teilnehmer können nicht am Meeting teilnehmen	6	8	4	192	Verbesserung der Netzwerkredundanz, Fehlermeldungen mit Wiederverbindungsversuch
2	Host kann keine Gäste hereinlassen	Backend-Fehlfunktion oder fehlerhafte API-Anfrage	Verzögerungen oder Abbruch des Meetings	5	7	3	105	Logging und automatische Neustartmechanismen implementieren
3	Audio- oder Videoausfall	Fehlerhafte Gerätekonfiguration oder Browser-Einschränkungen	Kommunikationsprobleme zwischen Teilnehmern	6	6	4	144	Geräte- und Berechtigungsprüfung vor dem Meeting
4	Transkription nicht verfügbar oder fehlerhaft	API-Ausfall oder hohe Latenz von OpenAI	Verlust wichtiger Meeting-Inhalte	4	9	3	108	Alternative Transkriptionsmethode als Backup einbinden
5	Datenschutzproblem bei Aufzeichnung	Falsche Zugriffsrechte oder Leck in der API	Unbefugter Zugriff auf sensiblen Inhalt	3	10	2	60	Strikte Berechtigungskontrolle und Datenverschlüsselung
6	Fehlerhafte Bildschirmübertragung	Inkompatible Browser oder Betriebssysteme	Teilnehmer können Inhalte nicht sehen	5	5	3	75	Browserkompatibilität testen, alternative Screen-Sharing-Methoden anbieten
7	Warteraumfunktion fehlerhaft	Socket Probleme	Unkontrollierter Zugang oder Verzögerungen	5	8	3	120	Debugging-Tools einbinden, Fehlerhandlung verbessern

**Maßnahmen und Optimierungen** basierend auf der Analyse sollten folgende Maßnahmen priorisiert werden:

- **Netzwerkstabilität optimieren** – Bessere Server-Infrastruktur, Failover-Mechanismen
- **Gerätekompatibilität sicherstellen** – Benutzerfreundliche Testtools für Audio- und Videoeinstellungen
- **Sicherheitsmaßnahmen verstärken** – Datenverschlüsselung, Zugriffskontrollen
- **Alternative Transkriptionsmethoden einbinden** – Lokale KI-Modelle als Backup

Durch diese Maßnahmen kann die Stabilität, Benutzerfreundlichkeit und Sicherheit der Anwendung verbessert werden.

## 4. Verifizierung Gebrauchstauglichkeit

Im Gebrauchstauglichkeitsprozess wird verifiziert, ob alle Anforderungen durch automatische UI-Tests und Schüler/Tutoren-Feedback erfüllt sind. Die Tests sollen sicherstellen, dass die Anwendung den Anforderungen entspricht.

<b>Testfall</b>	<b>Beschreibung</b>
Test case ID	1
Test geplant von	Philip Schrenk
Datum des Erstellens	19.03.2025
Titel	Feste Meeting-URLs für Host
Testbeschreibung	Hosts haben eine statische Meeting-URL
Vorbedingung	Eine Internetverbindung ist vorhanden
Testschritte	Es wird DOMAIN/room/[hostid] in die suchleiste eingegeben
Testdaten	localhost:3000/room/PRE
Erwartetes Ergebnis	Man landet im Warteraum des Hosts
Nachbedingung	Man muss beitreten können
Voraussetzung	-

<b>Testfall</b>	<b>Beschreibung</b>
Test case ID	2
Test geplant von	Philip Schrenk
Datum des Erstellens	19.03.2025
Titel	Warteraum für Gäste
Testbeschreibung	Gäste betreten nach dem Zugriff auf die Meeting-URL einen Warteraum und bleiben dort, bis der Host sie genehmigt.
Vorbedingung	Eine Internetverbindung ist vorhanden und ein Host ist bereits im Meeting.
Testschritte	Es wird der Knopf „Beitrittsanfrage senden“ geklickt.
Testdaten	-
Erwartetes Ergebnis	Der Host erhält die Beitrittsanfrage in seiner Oberfläche.
Nachbedingung	Die Anfrage muss akzeptiert werden können.
Voraussetzung	-

<b>Testfall</b>	<b>Beschreibung</b>
Test case ID	3
Test geplant von	Philip Schrenk
Datum des Erstellens	19.03.2025
Titel	Vom Host verwalteter Gastzugang
Testbeschreibung	Hosts können Gäste innerhalb der Benutzeroberfläche genehmigen oder ablehnen und so die volle Kontrolle darüber haben, wer dem Meeting beitreten darf.
Vorbedingung	Test 2 wurde durchgeführt
Testschritte	Der Host klickt, nach Empfangen der Anfrage, entweder auf Akzeptieren oder Ablehnen.
Testdaten	-
Erwartetes Ergebnis	Abgelehnt: Der Guest wird nicht ins Meeting hereingelassen und bekommt die Nachricht „Anfrage abgelehnt“. Angenommen: Der Guest wird ins Meeting hereingelassen und kann nun mit dem Host kommunizieren.
Nachbedingung	-
Voraussetzung	-

<b>Testfall</b>	<b>Beschreibung</b>
Test case ID	4
Test geplant von	Philip Schrenk
Datum des Erstellens	19.03.2025
Titel	Videokonferenz
Testbeschreibung	Die Anwendung bietet Grundlegende Meeting-Funktionen, und der Host kann ein Meeting starten und beenden.
Vorbedingung	Es ist mindestens eine Person im Meeting.
Testschritte	Man betritt das Meeting, aktiviert und deaktiviert die Kamera, das Mikrofon, die Audioausgabe, die Bildschirmübertragung und verlässt das Meeting.
Testdaten	-
Erwartetes Ergebnis	Die jeweiligen Ereignisse treffen ein
Nachbedingung	-
Voraussetzung	-

<b>Testfall</b>	<b>Beschreibung</b>
Test case ID	5
Test geplant von	Philip Schrenk
Datum des Erstellens	19.03.2025
Titel	Aufzeichnungen mit Transkription und Zusammenfassung
Testbeschreibung	Meetings werden automatisch aufgezeichnet und in AWS S3 gespeichert. GPT analysiert die Transkripte, um Meeting-Zusammenfassungen, wichtige Erkenntnisse und Handlungspunkte zu generieren.
Vorbedingung	Es ist mindestens eine Person im Meeting.
Testschritte	Es wird ein Meeting von mindestens 5 Minuten gehalten und ein beliebiges Thema klar und deutlich besprochen, damit die Transkription getestet werden kann.
Testdaten	Es wird über ein interessantes Thema geredet.
Erwartetes Ergebnis	Eine Zusammenfassung der Transkription wird an S3 geschickt und gespeichert.
Nachbedingung	Auf diese Zusammenfassung kann zugegriffen werden.
Voraussetzung	-



# Pflichtenheft

IOE-Videocall

Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz

2024/25

## Zusammenfassung

Dieses Dokument dient der Beschreibung der technischen Realisierung der geforderten Funktionen des Projekts. In diesem Projekt wird eine Webapplikation programmiert, welche eine Videomeeting Anwendung für die Firma zur Verfügung steht. Aufgrund der Verwendung eines agilen Vorgehensmodells, werden erst nach Realisierungen der Funktionen und nicht funktionalen Anforderungen diese hier dokumentiert.

Unvollständige Kapitel beinhalten nur eine kurze Beschreibung

## Inhalt

Zusammenfassung .....	2
Beschreibung des Ist- und Sollzustandes .....	4
Ist-Zustand .....	4
Soll-Zustand .....	4
Software- und Systemarchitektur.....	5
Layout/Benutzeroberfläche .....	6
Der Warteraum .....	6
Meeting-View (mit Beitrittsanfrage) .....	7
Meeting-View mit Teilnehmern inkl. Kamera .....	8
Modal für Einstellungen .....	9
Systemschnittstellen .....	10
Realisierung nicht-funktionaler Anforderungen .....	11
Realisierung der Funktionen .....	12
Abnahmekriterien .....	21

# 1. Beschreibung des Ist- und Sollzustandes

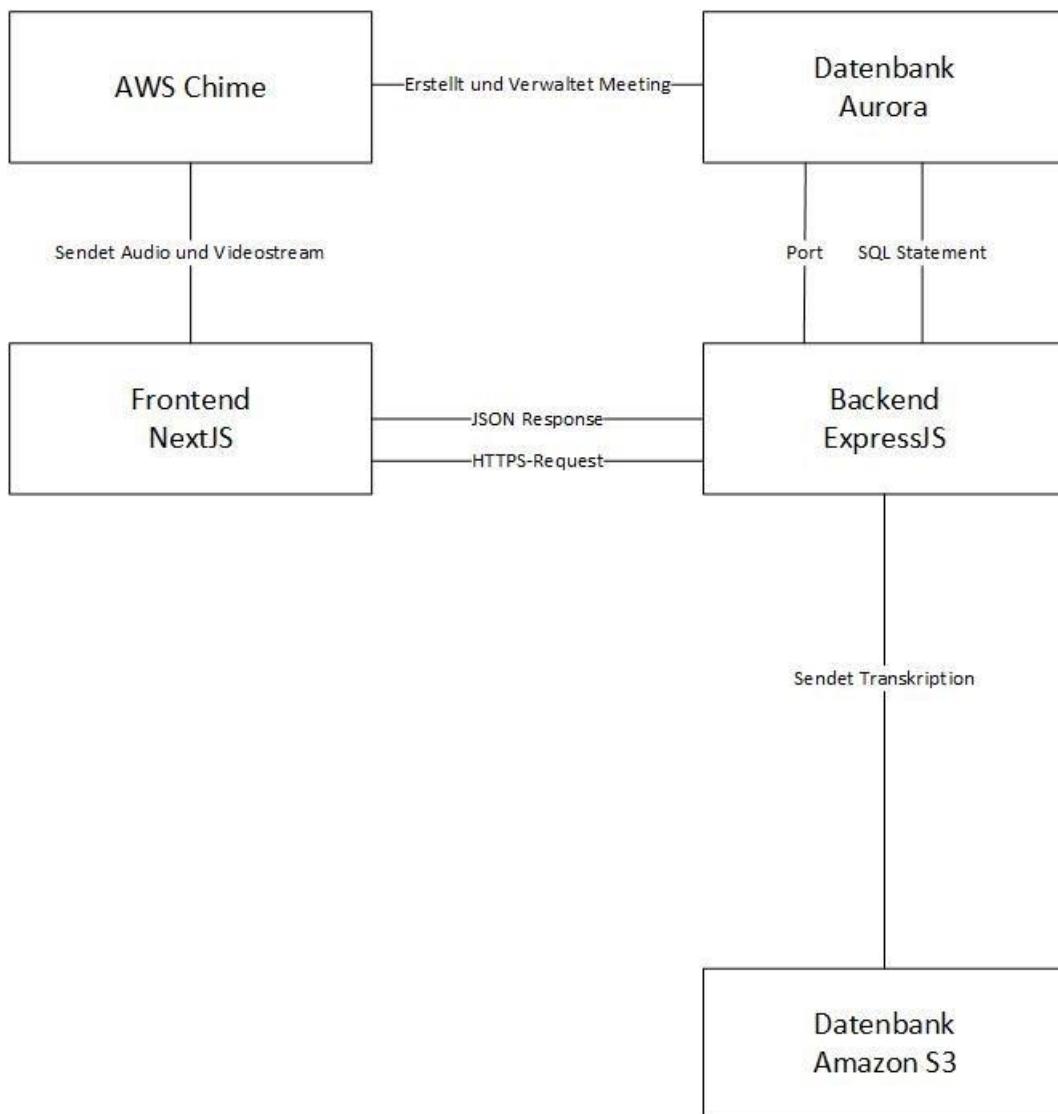
## 1.1 Ist-Zustand

Die Firma Naos GmbH organisiert regelmäßig Videokonferenzen mit ihren Kunden. Derzeit wird hierfür allgemeine Videokonferenzlösungen verwendet, die nicht optimal auf die Anforderungen dieses Szenarios abgestimmt sind. Dies führt zu mehreren Einschränkungen wie Umständliche Meeting-Verwaltung, Eingeschränkte Funktionalität und Mangelnde Benutzerfreundlichkeit. Diese Nachteile führen zu einem ineffizienten Workflow, erhöhen den organisatorischen Aufwand und beeinträchtigen die Qualität der Kommunikation. Ein maßgeschneidertes Webvideocall-Tool soll diese Probleme beheben und die Prozesse optimieren.

## 1.2 Soll-Zustand

Das Ziel ist die Entwicklung einer Webapplikation, die eine benutzerfreundliche Plattform für Videokonferenzen bietet. Sie ermöglicht Meetings über eine statische Meeting-URL zu verwalten, und Usern, über einen Warteraum teilzunehmen, inklusive Videokonferenzfunktionen, Live-Transkription und automatischer Zusammenfassungen. Durch die intuitive Bedienung können Tutoren und Schüler die Anwendung ohne lange Einarbeitung nutzen, was den Aufwand reduziert und die Effizienz steigert.

## 2. Software- und Systemarchitektur



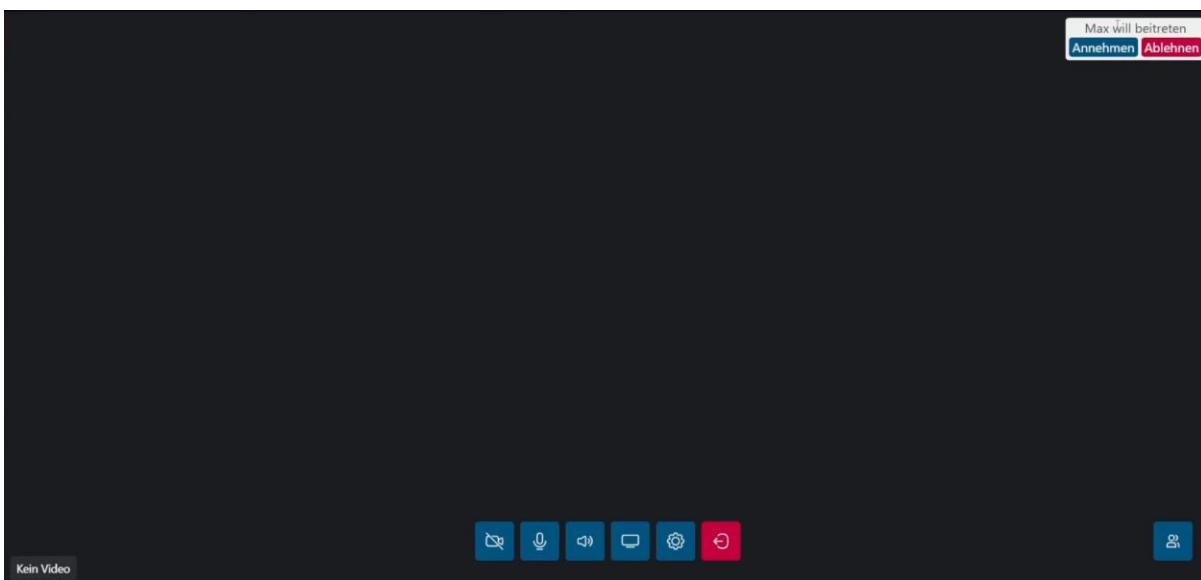
## 3. Layout/Benutzeroberfläche

### 3.1 Der Warteraum



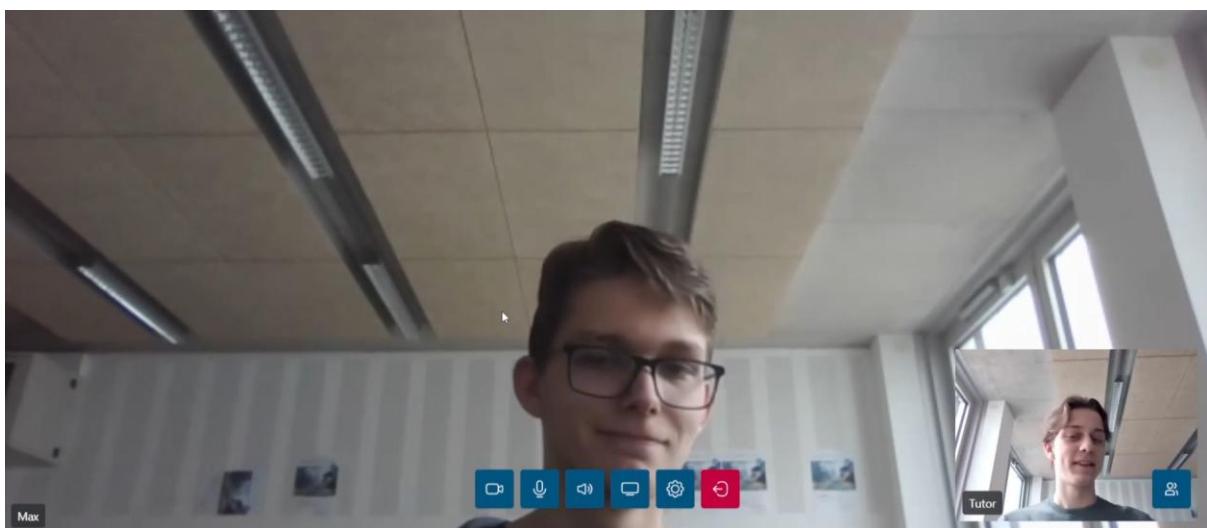
Dies ist die Ansicht, die Gäste sehen, nachdem sie die statische Meeting-URL, die ihnen von einem Tutor (Host) bereitgestellt wurde, aufrufen. Die Seite ist unter /rooms/[Id] erreichbar, wobei [Id] die eindeutige Meeting-ID repräsentiert. Sobald ein Schüler die URL öffnet, wird er automatisch in den Warteraum weitergeleitet. Der Warteraum dient als Sicherheitsmechanismus, um sicherzustellen, dass nur autorisierte Teilnehmer am Meeting teilnehmen können. Der Guest sieht eine personalisierte Begrüßung (z. B. „Willkommen Max“), die auf den eingeloggten Benutzer hinweist, sowie den Titel des Meetings und die geplante Zeitspanne. Der Button „BEITRITSANFRAGE SENDEN“ ermöglicht es dem Schüler, eine Anfrage an den Tutor zu senden. Sobald die Anfrage gesendet wurde, ändert sich der Text des Buttons zu „Beitrittsanfrage gesendet“, und der Schüler wartet auf die Freigabe durch den Tutor. Diese Funktion stellt sicher, dass Tutoren die Kontrolle über die Teilnehmerliste behalten und unbefugte Zugriffe verhindert werden. Der Warteraum ist ein zentraler Bestandteil der Benutzererfahrung für Schüler und unterstützt die organisatorische Struktur des Institute of Entrepreneurship, indem er eine klare Trennung zwischen Hosts und Gästen gewährleistet.

### 3.2 Meeting-View (mit Beitrittsanfrage)



Dies ist die Ansicht, die ein Tutor (Host) sieht, nachdem er ein Meeting über die statische URL erstellt hat und ein Schüler (Gast) eine Beitrittsanfrage gesendet hat. Der Bildschirm zeigt zunächst keine Videoübertragung, da weder die Kamera des Tutors noch die eines Schülers aktiviert ist. Am oberen Rand des Bildschirms erscheint eine Benachrichtigung in Form eines blauen Buttons mit der Aufschrift „Max will beitreten“, was darauf hinweist, dass ein Schüler namens Max eine Beitrittsanfrage gesendet hat. Neben diesem Button gibt es einen roten „Ablehnen“-Button, der es dem Tutor ermöglicht, die Anfrage abzulehnen, falls der Schüler nicht zugelassen werden soll. Diese Funktion gibt Tutoren die volle Kontrolle über die Teilnehmerliste und erhöht die Sicherheit des Meetings. Am unteren Rand des Bildschirms befinden sich die Steuerungselemente der Videokonferenz: Symbole für das Ein- und Ausschalten des Mikrofons, der Kamera, der Bildschirmübertragung sowie ein rotes Symbol zum Beenden des Meetings. Diese Symbole sind intuitiv gestaltet, wobei ein durchgestrichenes Symbol (z. B. ein durchgestrichenes Mikrofon) anzeigt, dass die Funktion deaktiviert ist. Sobald der Tutor auf „Max will beitreten“ klickt, wird der Schüler in das Meeting aufgenommen, und die Videokonferenz kann beginnen. Diese Ansicht stellt sicher, dass Tutoren jederzeit den Überblick über anstehende Beitrittsanfragen behalten und die Meeting-Umgebung aktiv steuern können.

### 3.3 Meeting-View mit Teilnehmern inkl. Kamera



Dies ist die Ansicht eines laufenden Meetings, nachdem ein Guest vom Tutor (Host) zugelassen wurde und beide ihre Kameras aktiviert haben. Die Benutzeroberfläche zeigt zwei Videofenster: eines für den Tutor und eines für den Schüler, wobei die Namen der Teilnehmer (z. B. „Tutor“) in den jeweiligen Fenstern angezeigt werden. Diese Ansicht demonstriert das responsive Layout der Anwendung, das sich automatisch an die Anzahl der Teilnehmer anpasst. Wenn weitere Schüler beitreten, werden zusätzliche Videofenster hinzugefügt, wobei das Layout dynamisch skaliert, um alle Teilnehmer sichtbar zu halten. Am unteren Rand der Ansicht befinden sich die Steuerungselemente, die sowohl für Tutoren als auch für Schüler verfügbar sind: Symbole zum Ein- und Ausschalten des Mikrofons, der Kamera, der Bildschirmübertragung sowie ein rotes Symbol zum Verlassen oder Beenden des Meetings. Während des Meetings wird der Anruf live transkribiert, und die Transkription kann über ein separates Fenster (nicht abgebildet) eingesehen werden. Nach Abschluss des Meetings wird die Transkription mithilfe von OpenAI zusammengefasst und den Teilnehmern zur Verfügung gestellt. Diese Ansicht stellt sicher, dass Tutoren und Schüler eine klare und intuitive Benutzererfahrung haben, die den Anforderungen eines Online-Unterrichts im Institute of Entrepreneurship entspricht. Die Möglichkeit, Kameras ein- und auszuschalten, ermöglicht Flexibilität und Privatsphäre für alle Beteiligten.

### 3.4 Modal für Einstellungen



Dieses Modal-Fenster wird angezeigt, wenn ein Benutzer (Tutor oder Schüler) auf das Einstellungssymbol klickt, das sich in der Steuerleiste am unteren Rand des Meeting-Bildschirms befindet. Das Modal ermöglicht es den Benutzern, ihre Audio- und Videoeinstellungen anzupassen, um eine optimale Meeting-Erfahrung zu gewährleisten. Unter der Überschrift „EINSTELLUNGEN“ sind drei Hauptbereiche aufgeführt: Kameraeinstellungen, Mikrofoneinstellungen und Lautsprechereinstellungen. Im Bereich „Kameraqualität auswählen“ können Benutzer die Auflösung ihrer Kamera anpassen (z. B. 720p, 1080p), um die Videoqualität an ihre Internetverbindung anzupassen. Das Dropdown-Menü „INTEGRATED CAMERA (04270)“ zeigt die verfügbaren Kameras an, die mit dem Gerät verbunden sind, und ermöglicht es dem Benutzer, zwischen verschiedenen Kameras zu wechseln (z. B. integrierte Kamera oder externe Webcam). Im Bereich „MIKROFON AUSWÄHLEN“ können Benutzer ihr bevorzugtes Mikrofon auswählen, wobei das Standardmikrofon (z. B. „Mikrofon (Realtek(R) Audio)“) voreingestellt ist. Ähnlich können Benutzer im Bereich „LAUTSPRECHER AUSWÄHLEN“ ihre bevorzugten Lautsprecher auswählen, wobei die Standardoption (z. B. „Standard – Lautsprecher (Realtek(R) Audio)“) voreingestellt ist. Nachdem die Einstellungen vorgenommen wurden, können Benutzer das Modal schließen, und die Änderungen werden sofort angewendet. Diese Funktion ist besonders nützlich, um technische Probleme während eines Meetings zu beheben, und stellt sicher, dass sowohl Tutoren als auch Schüler ihre Geräte optimal konfigurieren können, um an einem reibungslosen Online-Unterricht teilzunehmen.

## 4. Systemschnittstellen

### 4.1 Intern

Name	Art	Beschreibung	Port
Website	HTTP	Interne Webanwendung	443
API	REST	Backend API für Services	443

### 4.2 Extern

Name	Art	Beschreibung	Port
AWS Chime	API	Cloud-Telefonie- und Videokonferenzdienst	443
Datenbank	SQL	Externe Datenbankverbindung	5432

## 5. Realisierung nicht-funktionaler Anforderungen

- Die Video- und Audioübertragung sollte mit minimaler Verzögerung erfolgen, um eine flüssige und natürliche Kommunikation zu ermöglichen.
- Alle Daten während der Übertragung verschlüsselt werden, um Vertraulichkeit und Datenschutz zu gewährleisten.
- Die Benutzeroberfläche sollte einfach und verständlich sein, sodass Nutzer sie ohne umfangreiche Schulung bedienen können.
- Das System sollte eine wachsende Anzahl von gleichzeitigen Nutzern und Meetings unterstützen, ohne dass die Leistung leidet.
- Das Tool sollte mit den gängigsten Webbrowsern kompatibel sein.
- Nur die unbedingt notwendigen Daten sollten gesammelt und gespeichert werden, um die Privatsphäre der Nutzer zu schützen.
- Die Speicherung und Verarbeitung von Daten muss den geltenden Datenschutzbestimmungen entsprechen

Die Video- und Audioübertragung mit minimaler Verzögerung wurde durch die Integration der AWS Chime SDK realisiert, die WebRTC für Echtzeitkommunikation nutzt, während die Verschlüsselung aller Daten während der Übertragung durch Ende-zu-Ende-Verschlüsselung der SDK gewährleistet wird. Eine einfache und verständliche Benutzeroberfläche wurde mit Next.js und Material-UI-Komponenten entwickelt, um intuitive Navigation ohne Schulung zu ermöglichen. Skalierbarkeit für wachsende Nutzerzahlen und Meetings wurde durch die elastische Infrastruktur von AWS (z. B. Auto-Scaling) sichergestellt, und die Kompatibilität mit gängigen Browsern (Chrome, Firefox, Edge) wurde durch umfassende Tests erreicht. Datenschutz wurde durch Minimierung der Datensammlung (nur Meeting-Metadaten und Transkripte) sowie durch Speicherung auf DSGVO-konformen AWS-Servern in der EU umgesetzt, wobei nur notwendige Daten verarbeitet und durch Zugriffsbeschränkungen geschützt werden.

## 6. Realisierung der Funktionen

- Feste Meeting-URLs für Host
- Warteraum für Gäste: Gäste betreten nach dem Zugriff auf die Meeting-URL einen Warteraum und bleiben dort, bis der Host sie genehmigt.
- Vom Host verwalteter Gastzugang: Hosts können Gäste innerhalb der Benutzeroberfläche genehmigen oder ablehnen und so die volle Kontrolle darüber haben, wer dem Meeting beitreten darf.
- Videokonferenz: Die Anwendung bietet Grundlegende Meeting-Funktionen, und der Host kann ein Meeting starten und beenden
- Aufzeichnungen mit Transkription und Zusammenfassung: Meetings werden automatisch aufgezeichnet und in AWS S3 gespeichert; GPT analysiert die Transkripte, um Meeting-Zusammenfassungen, wichtige Erkenntnisse und Handlungspunkte zu generieren

### 6.1 Feste Meeting URLs für Host

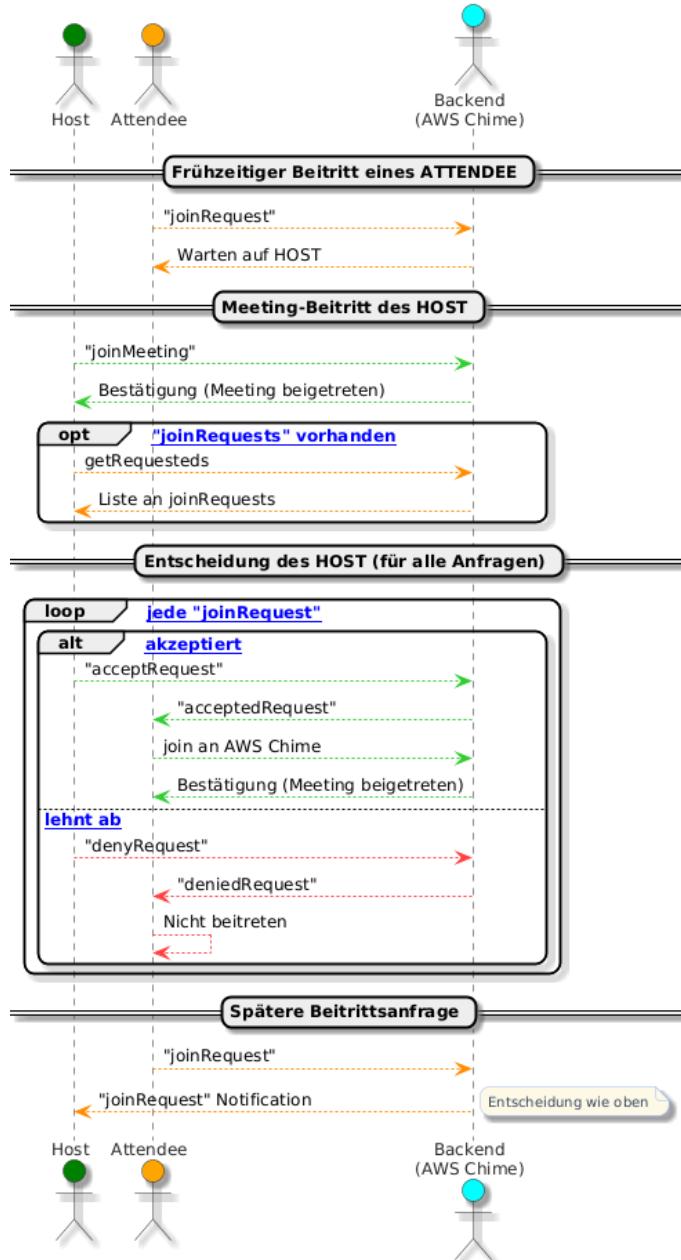
Die Implementierung dieses Features lässt sich dank der dynamischen Seitenfunktion von Next.js auf elegante und effiziente Weise realisieren. Durch die Möglichkeit, Seiten dynamisch basierend auf den bereitgestellten Parametern zu generieren, entfällt die Notwendigkeit statischer Routen, was die Entwicklung vereinfacht und gleichzeitig eine flexible Handhabung der Meeting-URLs ermöglicht.



## 6.2 Warteraum für Gäste

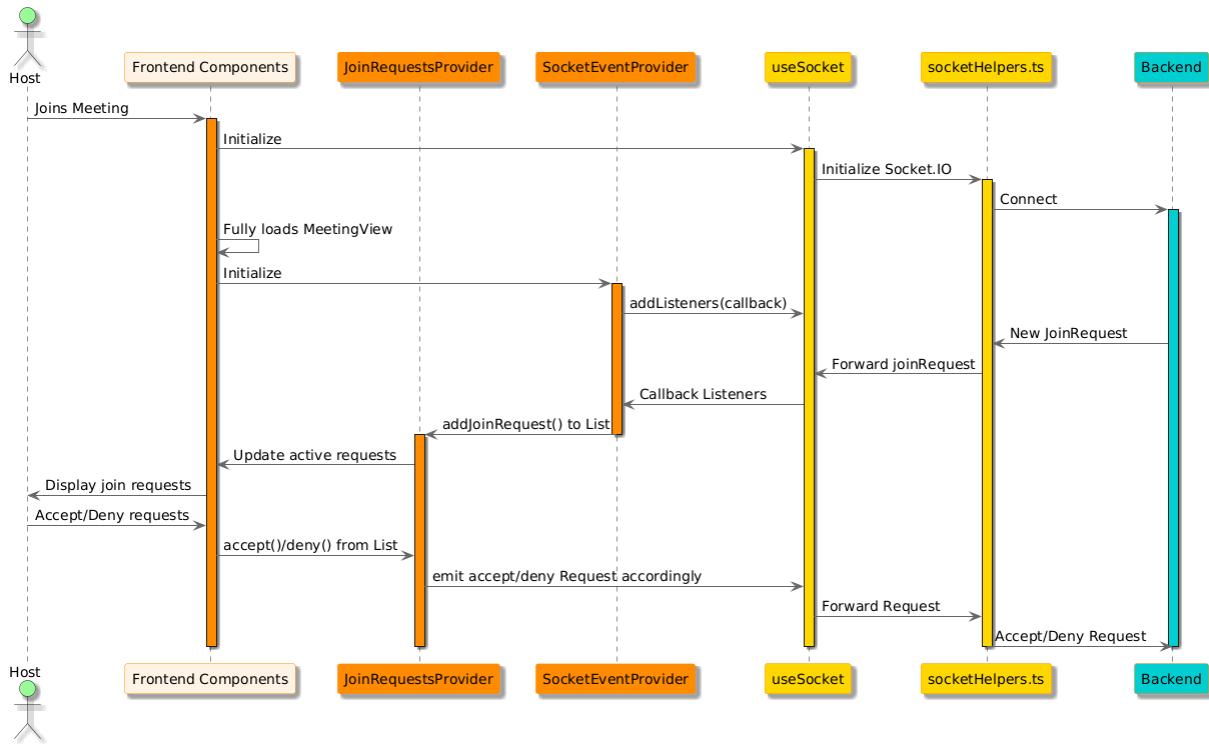
Die folgenden Diagramme können dieses Umgesetzte Feature in Hinsicht auf Ablauf am besten abgebildet werden.

SocketIO ist sowohl ein Backend- als auch ein Frontend-Package, welches das Versenden und Erstellen von Socket-Nachrichten wesentlich vereinfacht. In Absprache mit dem Backend haben wir für diesen Ablauf der Anfragen entschieden, der in Form eines Sequenzdiagrammes in PlantUML dargestellt wurde.



Das nachstehende Diagramm zeigt das Management einer Warteraum-Anfrage im Frontend von Komponent zu Komponent.

1. Sobald ein/e Benutzer/in die Seite betritt, werden die Warteraumkomponenten geladen. Noch werden keine Komponenten geladen, abgesehen vom React Context Provider JoinRequestsProvider, welcher immer aktiv ist und sich nur updated, sobald eine neue Request vom Backend kommt. Daher zur Zeit noch keine konkrete Verbindung zum Backend besteht, ist dieser leer.
2. Unmittelbar nachdem, ein/e Benutzer/in den Knopf "Meeting Starten" oder "Beitrittsanfrage senden" betätigt hat, wird eine Verbindung zu unserem Socket mit Hilfe von useSocket und SocketHelpers und dem Socket von AWS Chime hergestellt. Damit wird die Socket-Message "joinMeeting", oder "joinRequest" für Attendees, zum Beitreten geschickt. Welche im letzten Fall, auch nun allen Hosts angezeigt wird.
3. Beim Laden der Seite wird die Komponente MeetingView, und damit auch SocketEventProvider, geladen und dessen useEffect aufgerufen. Das useEffect, in SocketEventProvider, erstellt Listener für diverse eingehende Socket-Nachrichten, unter anderem "joinRequest". Dieser ist nur nützlich für Hosts, daher auch nur Hosts "joinRequests" empfangen können.
4. Im Falle eines Host-Clients, wartet das System danach auf Beitrittsanfragen. Wenn eine Beitrittsanfrage kommt, wird sie von SocketHelpers abgefangen. Da SocketHelpers als Hook selbst keine Client-Component Features ausführen kann, wird der Listener, samt Callback, in SocketEventProvider ausgeführt.
5. Der Callback im SocketEventProvider ruft danach JoinRequestsProvider auf, um die Liste der Beitrittsanfragen zu aktualisieren. Jede Komponente, die nun diese Liste verwendet, wird automatisch mit den Änderungen aktualisiert, da es sich hier, wie schon erwähnt, um einen React Context Provider handelt.
6. Es ist nun Aufgabe der Frontend-Komponenten, diese Benachrichtigung, samt den Optionen zum Annehmen und Ablehnen, anzuzeigen. Nach der Annahme oder Ablehnung der Anfrage wird die useSocket Methode emit() direkt vom JoinRequestsProvider aufgerufen.



## 6.3 Vom Host verwalteter Gastzugang

```

socket.on('joinRequest', async ({ requested_id, meetingId }: { requested_id: string; meetingId: string }) => {
  var first = await Requested.findOne({ where: { socket_id: socket.id } });
  if (first) {
    console.log('User already has a request!');
    io.to(`meeting-${meetingId}`).emit('joinRequest', { requested_id, meetingId, name: first.name });
    return;
  }
  var requested = await Requested.findOne({ where: { uuid: requested_id, meeting_id: meetingId } });
  if (!requested) {
    console.log('Request not found!');
    return;
  }
  requested.socket_id = socket.id;
  await requested.save();
  io.to(`meeting-${meetingId}`).emit('joinRequest', { requested_id, meetingId, name: requested.name });
  console.log(`Benutzer ${requested_id} hat eine Anfrage gesendet.`);
});

```

Hier fragt der Kunde an, um den Videomeeting beizutreten. Es wird ein Socket Request aus dem Frontend geschickt. Diese Benachrichtigung wird dann an jedem Host im Meeting geschickt. Das erfolgt mit der Room Funktion von socket.io. Mit socket.io kann man verschiedene User in Räume geben und dann individuelle Nachrichten in diese Gruppen versenden.

```
socket.on('acceptRequest', async ({ requested_id, meetingId }: { requested_id: string; meetingId: string }) => {
  const requested = await Requested.findOne({ where: { uuid: requested_id, meeting_id: meetingId } });
  if (requested) {
    var uuid = v4();
    const awsAttendee = await chime.createAttendee({
      MeetingId: meetingId,
      ExternalUserId: uuid
    }).promise();
    if (!awsAttendee.Attendee?.AttendeeId) {
      console.log('Failed to create attendee!');
      return;
    }
    var attendee = await Attendee.create({
      uuid,
      name: requested.name,
      attendee_id: awsAttendee.Attendee.AttendeeId,
      meeting_id: requested.meeting_id,
      isHost: false
    });
    var socket_id = requested.socket_id;
    await requested.destroy();
    const awsMeeting = await chime.getMeeting({ MeetingId: meetingId }).promise();
    if (!awsMeeting.Meeting) {
      console.log('Failed to get meeting!');
      return;
    }
    io.to(socket_id).emit('acceptedRequest', { Meeting: awsMeeting.Meeting, Attendee: awsAttendee.Attendee });
    io.to(`meeting-${meetingId}`).emit('userJoined', attendee.name);
    io.to(`meeting-attendee-${meetingId}`).emit('userJoined', attendee.name);
    socket.join(`meeting-attendee-${meetingId}`);
    return;
  }
  io.to(`meeting-${meetingId}`).emit('acceptedRequest', 'Request denied');
});
```

Anschließend ist es dem Host möglich, jeden Anfragenden anzunehmen, oder auch abzulehnen. Auch hier muss ein Socket Request aus dem Frontend erfolgen.

## 6.4 Videokonferenz

Die Anwendung ermöglicht grundlegende Videokonferenzfunktionen: Mikrofon an/aus, Kamera an/aus, Bildschirmübertragung und Audio-Steuerung. Diese wurden vollständig mit dem AWS Chime SDK implementiert, das als WebRTC-basierte Lösung alle notwendigen Schnittstellen für Audio- und Videostreaming bereitstellt. Die Mediensteuerung erfolgt über das Chime Meeting Session API, welches die Verbindung zu Meeting-Räumen verwaltet, Teilnehmerstatus überwacht und Echtzeit-Änderungen an den Mediengeräten unterstützt. Frontend-seitig wurde dies mit React umgesetzt, indem Benutzeraktionen direkt an Chime API-Methoden gebunden wurden, sodass die Steuerung der Geräte nahtlos funktioniert und Zustandsänderungen sofort im UI reflektiert werden.

Die nachfolgenden Ausschnitte zeigen die Implementierung wichtiger Chime-Komponenten im Frontend und das Beitreten eines Benutzers im Backend:

```
function MyApp({ Component }: AppProps) {
  return (
    <ThemeProvider theme={lightTheme}>
      <GlobalStyles />
      <ChimeUserProvider>
        <JoinRequestsProvider>
          <MeetingProvider>
            <Component />
          </MeetingProvider>
        </JoinRequestsProvider>
      </ChimeUserProvider>
    </ThemeProvider>
  );
}

export default MyApp;
```

```
var awsAttendee = await chime
  .createAttendee({
    MeetingId: meeting.meeting_id,
    ExternalUserId: uuid,
  })
  .promise();
```

```
return (
  <>
  {inMeeting ? (
    <SocketEventProvider>
      <BackgroundBlurProvider>
        <BackgroundReplacementProvider>
          <NotificationProvider>
            <MeetingView username={username!} />
          </NotificationProvider>
        </BackgroundReplacementProvider>
      </BackgroundBlurProvider>
    </SocketEventProvider>
  ) : (
    <>
  )}
)
```

## 6.5 Aufzeichnungen mit Transkription und Zusammenfassung

```
useEffect(() => {
  if (typeof window.webkitSpeechRecognition === "undefined") {
    console.error("Speech recognition not supported");
    return;
  }

  const recognitionInstance = new window.webkitSpeechRecognition();
  recognitionInstance.continuous = true;
  recognitionInstance.interimResults = false;
  recognitionInstance.lang = "de-DE";

  recognitionInstance.onresult = (event: SpeechRecognitionEvent) => {
    const latestResultIndex = event.results.length - 1;
    const latestResult = event.results[latestResultIndex];
    console.log(latestResult);

    if (latestResult.isFinal) [
      const newTranscript = latestResult[0].transcript;
      console.log(newTranscript);

      setTranscript([
        ...transcript,
        {
          speaker: user?.name || "Unbekannt",
          text: newTranscript,
          timestamp: Date.now(),
        },
      ]);
      console.log(transcript);
    ];
  };
};
```

```
recognitionInstance.start();
setRecognition(recognitionInstance);

const interval = setInterval(() => {
  saveTranskription();
}, 150000);

return () => {
  recognitionInstance.stop();
  clearInterval(interval);
};

}, []);

const saveTranskription = async () => {
  const text = transcript
    .map(
      (entry) =>
        `${new Date(entry.timestamp).toLocaleTimeString()} ${{
          entry.speaker
        }}: ${entry.text}`
    )
    .join("\n");
  console.log(text);

  const response = await generatePdf(text);
  if (response.success) console.log("Transkription saved!");
  else console.error("Transkription could not be saved!");

  setTranscript([]);
};
```

Der Code nutzt Spracherkennung in React, um Sprache aufzunehmen. Er prüft, ob der Browser das kann, und startet dann die Aufnahme auf Deutsch. Das Gesprochene wird gespeichert mit Namen, Text und Zeit. Alle 2,5 Minuten wird das Ganze als PDF gesichert und die Liste geleert. Am Ende wird die Aufnahme gestoppt.

```
const openai = new OpenAI({
  apiKey: OPENAI_API_KEY
});

async function summarizeAndCorrectText(text: string): Promise<string> {
  try {
    const response = await openai.chat.completions.create({
      model: "gpt-4o",
      messages: [
        {
          role: "user",
          content: "Fasse den folgenden Text zusammen. Gib nur die zusammengefasste Version zurück, ohne zusätzliche Kommentare oder Erklärungen."
        },
      ],
      temperature: 0.5,
      max_tokens: 150,
    });

    return response.choices[0].message.content || text;
  } catch (error) {
    console.error("Error calling OpenAI API:", error);
    return text;
  }
}

export default router;
```

```
router.post("/generatePDF", async (req: Request, res: Response) => {
  try {
    const { text } = req.body;
    const filePath = `${__dirname}/../../../../pdfexports/transcription.pdf`;
    const fontSize = 16;
    const margin = 50;
    const pageWidth = 600;
    const pageHeight = 800;

    const summarizedText = await summarizeAndCorrectText(text);

    let pdfDoc: PDFDocument;

    if (fs.existsSync(filePath)) {
      const existingPdfBytes = fs.readFileSync(filePath);
      pdfDoc = await PDFDocument.load(existingPdfBytes);
    } else {
      pdfDoc = await PDFDocument.create();
    }

    const page = pdfDoc.addPage([pageWidth, pageHeight]);

    const font = await pdfDoc.embedFont(StandardFonts.Helvetica);

    page.drawText(summarizedText, {
      x: margin,
      y: pageHeight - 100,
      size: fontSize,
      font,
      color: rgb(0, 0, 0),
      maxWidth: pageWidth - 2 * margin,
    });

    const pdfBytes = await pdfDoc.save();
    fs.writeFileSync(filePath, pdfBytes);

    res.json({
      message: "PDF updated successfully",
      filePath,
    });
  } catch (error) {
    console.error("Unexpected Error:", error);
    res.status(500).send("Internal Server Error");
  }
});
```

Der Code erstellt eine POST-Route /generatePDF, die Text aus einer Anfrage nimmt und in ein PDF schreibt. Der Text wird mit OpenAI zusammengefasst, dann in ein neues oder bestehendes PDF an einer festen Position eingefügt. Das PDF wird mit Schriftart Helvetica, Größe 16 und schwarzen Buchstaben gespeichert. Bei Erfolg gibt die Antwort eine Bestätigung und den Dateipfad zurück, bei Fehlern wird ein Serverfehler gemeldet. Die Funktion summarizeAndCorrectText nutzt OpenAI, um den Text zu kürzen, und gibt ihn zurück.

## 7. Abnahmekriterien

### Erfüllung aller Ziele:

- Feste Meeting-URLs für Host
- Warteraum für Gäste: Gäste betreten nach dem Zugriff auf die Meeting-URL einen Warteraum und bleiben dort, bis der Host sie genehmigt.
- Vom Host verwalteter Gastzugang: Hosts können Gäste innerhalb der Benutzeroberfläche genehmigen oder ablehnen und so die volle Kontrolle darüber haben, wer dem Meeting beitreten darf.
- Videokonferenz: Die Anwendung bietet Grundlegende Meeting-Funktionen, und der Host kann ein Meeting starten und beenden
- Aufzeichnungen mit Transkription und Zusammenfassung: Meetings werden automatisch aufgezeichnet und in AWS S3 gespeichert; GPT analysiert die Transkripte, um Meeting-Zusammenfassungen, wichtige Erkenntnisse und Handlungspunkte zu generieren



## Besprechungsprotokolle IOE-Videocall

Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz  
2024/25

Nr	DA Sprint 1
Typ	(Sprintplanning)
Datum	4.10.2024
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Tasks erstellen, besprechen und schätzen für den ersten Sprint</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Erste Tasks für den Sprint erstellt, besprochen und geschätzt</li> <li>• Tasks teilweise den Mitgliedern zugewiesen</li> <li>• Sprint gestartet</li> </ul>

Nr	DA Sprint 1
Typ	Standup
Datum	11.10.2024
Standort	HTL Spengergasse
Teilnehmer	Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Aufgabenverteilung, Priorisierung von Aufgaben</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• In der vergangenen Woche wurde nicht allzu viel fertiggestellt.</li> <li>• Die nächsten Wochen wird sich hauptsächlich auf PRE Aufgaben fokussiert, Sprint sollte trotzdem planmäßig durchgeführt werden.</li> </ul>

Nr	DA Sprint 1
Typ	Standup
Datum	18.10.2024
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was haben wir die Woche gemacht?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• In dieser Woche konnte aufgrund von schulischen Terminen nicht viel fertig gestellt werden</li> </ul>

Nr	DA Sprint 1
Typ	Meeting mit naos
Datum	24.10.2024
Standort	MS Teams
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz, Hannes Baumgartner, Armin Schleicher
Agenda	<ul style="list-style-type: none"> <li>Was haben wir die letzten 2 Wochen gemacht?</li> <li>Wo gab es Probleme?</li> <li>Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>In den letzten 2 Wochen wurde das Backend sowie das Frontend zum coden eingerichtet.</li> </ul>

Nr	DA Sprint 2
Typ	Sprintplanning, Retro
Datum	25.10.2024
Standort	Discord
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>Was haben wir die letzten 3 Wochen gemacht?</li> <li>Wo gab es Probleme?</li> <li>Was wird nächsten Sprint gemacht?</li> <li>Was können wir für den nächsten Sprint verbessern?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>Tasks den Mitgliedern zugewiesen</li> <li>Bessere Aufgabenverteilung</li> <li>Mehr Absprachen untereinander</li> <li>Sprint gestartet</li> </ul>

Nr	DA Sprint 2
Typ	Meeting mit naos
Datum	13.11.2024
Standort	MS Teams
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz, Hannes Baumgartner, Armin Schleicher
Agenda	<ul style="list-style-type: none"> <li>Was haben wir die letzte Woche gemacht?</li> <li>Wo gab es Probleme?</li> <li>Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>In der letzten Woche wurde das Backend für den ersten Prototypen fertiggestellt.</li> <li>Am Frontend wird noch bis Freitag (15.11.) dafür gearbeitet</li> </ul>

Nr	DA Sprint 3
Typ	Sprint Planning
Datum	15.11.2024
Standort	MS Teams
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was wurde diesen Sprint erledigt?</li> <li>• Was blieb offen und warum?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommenden Sprint gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• 1. Prototyp kommenden Sprint fertigstellen</li> </ul>

Nr	DA Sprint 3
Typ	Standup
Datum	22.11.2024
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was haben wir die letzte Woche gemacht?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Diese Woche wurde nur der 1. Prototype fertiggestellt.</li> <li>• In der folgenden Woche soll das Design optimiert werden und OpenAI Whisper erfolgreich ins Projekt integriert werden.</li> </ul>

Nr	DA Sprint 3
Typ	Standup
Datum	29.11.2024
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was haben wir die letzte Woche gemacht?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Durch schulischen Stress nicht viel weitergebracht</li> <li>• Nächste Woche wird nachgearbeitet</li> </ul>

Nr	DA Sprint 4
Typ	Sprint Planning / Retro
Datum	20.12.2024
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was wurde diesen Sprint erledigt?</li> <li>• Was blieb offen und warum?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommenden Sprint gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Userlist und Transkription wird am kommenden Sprint weitergearbeitet</li> <li>• Join Room Request wird überarbeitet</li> <li>• Änderung im Backend von JS zu TS</li> </ul>

Nr	DA Sprint 4
Typ	Standup
Datum	10.1.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was haben wir die letzte Woche gemacht?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird kommende Woche gemacht</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Prototyp Präsentation für PRE vorbereiten</li> <li>• Tests schreiben</li> <li>• Änderung von JS zu TS abgeschlossen</li> </ul>

Nr	DA Sprint 5
Typ	Sprintplanning/Retro
Datum	17.1.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was wurde diesen Sprint erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was lief gut?</li> <li>• Was könnte man verbessern</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Prototyp abgeschlossen</li> <li>• Sehr viel Fortschritt am Projekt</li> <li>• Warteraum Feature kommt</li> </ul>

Nr	DA Sprint 5
Typ	Meeting mit naos
Datum	22.1.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Sehr viel Fortschritt am Projekt</li> <li>• Warteraum Feature kommt</li> </ul>

Nr	DA Sprint 5
Typ	Sprintplanning/Retro
Datum	23.02.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Louis Muhr, Maximilian Schwarz
Agenda	<ul style="list-style-type: none"> <li>• Was wurde diesen Sprint erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was lief gut?</li> <li>• Was könnte man verbessern</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Wenig Fortschritte in letzter Zeit</li> <li>• Warteraum Feature besprochen</li> </ul>

Nr	DA Sprint 6
Typ	Meeting mit naos
Datum	26.02.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz, Louis Muhr
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• UI verschönert</li> <li>• InputSelections eingebaut</li> </ul>

Nr	DA Sprint 6
Typ	Meeting mit naos
Datum	05.03.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz, Louis Muhr
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>

Besprochene Themen	<ul style="list-style-type: none"> <li>• Warteraum eingebaut</li> <li>• Transkription vorgehensweise</li> <li>• </li> </ul>
--------------------	---

Nr	DA Sprint 6
Typ	Meeting mit naos
Datum	05.03.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz, Louis Muhr
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Warteraum erweitert</li> <li>• Projektabnahme festgelegt</li> <li>• Probleme mit einem AWS Chime Bug</li> </ul>

Nr	DA Sprint 6
Typ	Meeting mit naos
Datum	13.03.2025
Standort	HTL Spengergasse
Teilnehmer	Philip Schrenk
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Probleme mit demselben AWS Chime Bug wie letzte Woche</li> <li>• Wichtigkeit des Bugs diskutiert</li> </ul>

Nr	DA Sprint 6
Typ	Teambesprechung
Datum	16.03.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz, Louis Muhr
Agenda	<ul style="list-style-type: none"> <li>• Was wurde letzte Woche erledigt?</li> <li>• Wo gab es Probleme?</li> <li>• Was wird ab jetzt weiter gemacht?</li> </ul>
Besprochene Themen	<ul style="list-style-type: none"> <li>• Kamera Bug hat sich von selbst gelöst</li> <li>• Transkription fertig und gemerged mit main Branch</li> <li>• Transkription braucht noch kleine Tweaks</li> </ul>

Nr	DA Sprint 6
Typ	Projektabnahme
Datum	20.03.2025
Standort	HTL Spengergasse
Teilnehmer	Bastian Seidl, Philip Schrenk, Maximilian Schwarz, Louis Muhr
Agenda	<ul style="list-style-type: none"><li>• Was hat gut funktioniert?</li><li>• Wo gab es Probleme?</li><li>• Welche Features wurden umgesetzt?</li><li>• Wo gibt es Mängel?</li></ul>
Besprochene Themen	<ul style="list-style-type: none"><li>• Abnahme verlief im Großen und Ganzen gut.</li><li>• Transkription nur teilweise umgesetzt: Speicherung in S3 nicht durchgeführt</li></ul>

# Detailed report



28/09/2024 - 24/03/2025

Total: 573:54:02 Billable: 573:54:02 Amount: 0,00 USD

Date	Description	Duration	User
24/03/2025	Diploma / Formatting IOE-Videocall - Diploma (writing)	03:00:00 13:00:00 - 16:00:00	Schrenk Philip 0,00 USD
24/03/2025	Literaturverzeichnis IOE-Videocall - Diploma (writing)	01:20:00 12:00:00 - 13:20:00	Louismuhr8 0,00 USD
24/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	03:43:00 10:03:00 - 13:46:00	Scm120087 0,00 USD
23/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	06:00:00 20:00:00 - 02:00:00 <sup>+1</sup>	Scm120087 0,00 USD
23/03/2025	Korrekturlesen & Verbessern IOE-Videocall - Diploma (writing)	03:30:00 18:00:00 - 21:30:00	Louismuhr8 0,00 USD
23/03/2025	Letzte Korrekturen IOE-Videocall - Diploma (writing)	01:50:00 12:30:00 - 14:20:00	Louismuhr8 0,00 USD
22/03/2025	Fertigstellen IOE-Videocall - Diploma (writing)	02:20:00 17:00:00 - 19:20:00	Louismuhr8 0,00 USD
22/03/2025	Schreiben IOE-Videocall - Diploma (writing)	02:30:00 15:00:00 - 17:30:00	Louismuhr8 0,00 USD
22/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	05:00:00 14:00:00 - 19:00:00	Scm120087 0,00 USD
22/03/2025	Diploma / Design Choices IOE-Videocall - Diploma (writing)	03:00:00 12:00:00 - 15:00:00	Schrenk Philip 0,00 USD
21/03/2025	Diploma / Design Choices IOE-Videocall - Diploma (writing)	04:00:00 15:00:00 - 19:00:00	Schrenk Philip 0,00 USD
21/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	03:09:00 14:00:00 - 17:09:00	Scm120087 0,00 USD
21/03/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
20/03/2025	Deployment IOE-Videocall - Frontend	01:26:07 10:06:17 - 11:32:24	Schrenk Philip 0,00 USD

20/03/2025	<b>Deployment</b> IOE-Videocall - Backend	05:00:00 08:00:00 - 13:00:00	Bastian Seidl 0,00 USD
19/03/2025	<b>Diploma / Rewrite Warterau</b> IOE-Videocall - Diploma (writing)	02:01:23 20:20:51 - 22:22:14	Schrenk Philip 0,00 USD
19/03/2025	<b>Deployment</b> IOE-Videocall - Frontend	01:36:23 18:44:14 - 20:20:37	Schrenk Philip 0,00 USD
19/03/2025	<b>Deployment</b> IOE-Videocall - Backend	03:00:00 17:00:00 - 20:00:00	Bastian Seidl 0,00 USD
18/03/2025	<b>Transcription merge</b> IOE-Videocall - Frontend	01:40:09 11:00:00 - 12:40:09	Schrenk Philip 0,00 USD
17/03/2025	<b>(Without Description)</b> IOE-Videocall - Diploma (writing)	03:22:00 18:37:00 - 21:59:00	Scm120087 0,00 USD
17/03/2025	<b>Bug Fixing</b> IOE-Videocall - Backend	04:00:00 17:00:00 - 21:00:00	Bastian Seidl 0,00 USD
17/03/2025	<b>Finale Presentation</b> IOE-Videocall - PRE	00:00:02 12:08:08 - 12:08:10	Louismuhr8 0,00 USD
17/03/2025	<b>Diplom</b> IOE-Videocall - Diploma (writing)	04:22:56 12:01:04 - 16:24:00	Louismuhr8 0,00 USD
17/03/2025	<b>Fixed Awaiting Join Requests issue</b> IOE-Videocall - Frontend	01:30:00 11:00:00 - 12:30:00	Schrenk Philip 0,00 USD
17/03/2025	<b>Bug Fixing for waiting room</b> IOE-Videocall - Backend	05:15:00 10:45:00 - 16:00:00	Bastian Seidl 0,00 USD
17/03/2025	<b>(Without Description)</b> IOE-Videocall - PRE	02:09:00 10:14:00 - 12:23:00	Scm120087 0,00 USD
16/03/2025	<b>(Without Description)</b> IOE-Videocall - PRE	05:46:00 20:49:00 - 02:35:00 <sup>+1</sup>	Scm120087 0,00 USD
16/03/2025	<b>Finale Presentation</b> IOE-Videocall - PRE	00:19:00 20:45:00 - 21:04:00	Louismuhr8 0,00 USD
16/03/2025	<b>(Without Description)</b> IOE-Videocall - OpenAi Whisper	00:52:00 18:08:00 - 19:00:00	Scm120087 0,00 USD
16/03/2025	<b>Schreiben</b> IOE-Videocall - Diploma (writing)	02:40:00 15:30:00 - 18:10:00	Louismuhr8 0,00 USD
16/03/2025	<b>(Without Description)</b> IOE-Videocall - Diploma (writing)	02:00:00 15:00:00 - 17:00:00	Scm120087 0,00 USD

16/03/2025	Warteraum IOE-Videocall - Frontend	02:00:00 12:00:00 - 14:00:00	Schrenk Philip 0,00 USD
15/03/2025	(Without Description) IOE-Videocall - OpenAi Whisper	05:55:00 20:00:00 - 01:55:00 <sup>+1</sup>	Scm120087 0,00 USD
15/03/2025	(Without Description) IOE-Videocall - PRE	01:15:00 18:13:00 - 19:28:00	Scm120087 0,00 USD
15/03/2025	(Without Description) IOE-Videocall - PRE	03:12:00 13:37:00 - 16:49:00	Scm120087 0,00 USD
15/03/2025	Diplom IOE-Videocall - Diploma (writing)	01:39:58 13:30:00 - 15:09:58	Louismuhr8 0,00 USD
14/03/2025	Finale Presentation IOE-Videocall - PRE	00:45:02 16:37:06 - 17:22:08	Louismuhr8 0,00 USD
14/03/2025	Diplom IOE-Videocall - Diploma (writing)	04:45:00 12:45:00 - 17:30:00	Louismuhr8 0,00 USD
14/03/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
13/03/2025	Notifications pt2 IOE-Videocall - Frontend	00:45:00 14:30:00 - 15:15:00	Schrenk Philip 0,00 USD
13/03/2025	Weekly Call IOE-Videocall - Weekly call	00:45:00 13:45:00 - 14:30:00	Schrenk Philip 0,00 USD
13/03/2025	Notifications pt2 IOE-Videocall - Frontend	00:45:00 13:00:00 - 13:45:00	Schrenk Philip 0,00 USD
13/03/2025	Refactoring IOE-Videocall - Backend	02:15:00 12:45:00 - 15:00:00	Bastian Seidl 0,00 USD
13/03/2025	ReadMe Anleitung IOE-Videocall - Reviews	00:36:16 08:18:50 - 08:55:06	Louismuhr8 0,00 USD
13/03/2025	LeaveRoom Request & WaitingRoom Fix IOE-Videocall - Backend	04:45:00 08:00:00 - 12:45:00	Bastian Seidl 0,00 USD
12/03/2025	Bug Fixing IOE-Videocall - Backend	04:00:00 19:00:00 - 23:00:00	Bastian Seidl 0,00 USD
12/03/2025	Diploma / Warteraum IOE-Videocall - Diploma (writing)	04:00:00 18:00:00 - 22:00:00	Schrenk Philip 0,00 USD
12/03/2025	Sockets Bugfix IOE-Videocall - Backend	06:00:00 17:00:00 - 23:00:00	Bastian Seidl 0,00 USD

12/03/2025	<b>Weekly Call</b> IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
12/03/2025	<b>SettingsModal</b> IOE-Videocall - Frontend	02:00:00 08:00:00 - 10:00:00	Schrenk Philip 0,00 USD
11/03/2025	<b>Testing</b> IOE-Videocall - Backend	01:00:00 20:00:00 - 21:00:00	Bastian Seidl 0,00 USD
11/03/2025	<b>Diplom</b> IOE-Videocall - Diploma (writing)	02:42:00 18:04:00 - 20:46:00	Louismuhr8 0,00 USD
10/03/2025	<b>Warteraum Frontend (done)</b> IOE-Videocall - Frontend	01:11:26 18:48:34 - 20:00:00	Schrenk Philip 0,00 USD
10/03/2025	<b>WaitingRoom</b> IOE-Videocall - Backend	04:00:00 17:00:00 - 21:00:00	Bastian Seidl 0,00 USD
10/03/2025	<b>Diplom</b> IOE-Videocall - Diploma (writing)	00:41:00 16:25:00 - 17:06:00	Louismuhr8 0,00 USD
08/03/2025	<b>(Without Description)</b> IOE-Videocall - Diploma (writing)	02:42:00 14:15:00 - 16:57:00	Scm120087 0,00 USD
07/03/2025	<b>(Without Description)</b> IOE-Videocall - Diploma (writing)	02:29:00 15:55:00 - 18:24:00	Scm120087 0,00 USD
07/03/2025	<b>Bug Fixing</b> IOE-Videocall - Backend	05:00:00 14:00:00 - 19:00:00	Bastian Seidl 0,00 USD
07/03/2025	<b>Diplom</b> IOE-Videocall - Diploma (writing)	01:26:48 11:59:12 - 13:26:00	Louismuhr8 0,00 USD
07/03/2025	<b>(Without Description)</b> IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
06/03/2025	<b>ReadMe</b> IOE-Videocall - Helping Teammates	01:38:20 14:46:40 - 16:25:00	Louismuhr8 0,00 USD
06/03/2025	<b>Warteraum Frontend (done)</b> IOE-Videocall - Frontend	03:20:46 13:43:32 - 17:04:18	Schrenk Philip 0,00 USD
05/03/2025	<b>Diploma / TailwindCss</b> IOE-Videocall - Diploma (writing)	02:00:00 18:45:00 - 20:45:00	Schrenk Philip 0,00 USD
05/03/2025	<b>(Without Description)</b> IOE-Videocall - OpenAi Whisper	01:34:00 14:31:00 - 16:05:00	Scm120087 0,00 USD
05/03/2025	<b>Weekly Call</b> IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD

05/03/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
05/03/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
05/03/2025	Diploma IOE-Videocall - Diploma (writing)	01:55:00 12:45:00 - 14:40:00	Louismuhr8 0,00 USD
05/03/2025	Warteraum Frontend (done) IOE-Videocall - Frontend	04:19:41 08:30:34 - 12:50:15	Schrenk Philip 0,00 USD
05/03/2025	Warteraum Frontend (done) IOE-Videocall - Frontend	00:00:00 08:30:00 - 08:30:00	Schrenk Philip 0,00 USD
04/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	01:59:00 14:05:00 - 16:04:00	Louismuhr8 0,00 USD
02/03/2025	Diploma / Nextjs for AWS Chime IOE-Videocall - Frontend	02:00:00 18:00:00 - 20:00:00	Schrenk Philip 0,00 USD
02/03/2025	(Without Description) IOE-Videocall - Diploma (writing)	03:26:00 17:08:00 - 20:34:00	Louismuhr8 0,00 USD
28/02/2025	Warteraum Frontend (done) IOE-Videocall - Frontend	02:45:00 15:15:00 - 18:00:00	Schrenk Philip 0,00 USD
28/02/2025	Warteraum Frontend (done) IOE-Videocall - Frontend	01:19:52 10:45:00 - 12:04:52	Schrenk Philip 0,00 USD
28/02/2025	waitingroom IOE-Videocall - Reviews	02:33:00 09:50:00 - 12:23:00	Louismuhr8 0,00 USD
28/02/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
27/02/2025	Nextjs vs React IOE-Videocall - Diploma (writing)	03:00:00 17:00:00 - 20:00:00	Schrenk Philip 0,00 USD
27/02/2025	Warteraum accept IOE-Videocall - Frontend	00:41:59 10:00:00 - 10:41:59	Schrenk Philip 0,00 USD
26/02/2025	(Without Description) IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
26/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
26/02/2025	(Without Description) IOE-Videocall - OpenAi Whisper	02:40:00 13:25:00 - 16:05:00	Scm120087 0,00 USD

26/02/2025	Warterraum Frontend (done) IOE-Videocall - Frontend	08:56:39 08:03:21 - 17:00:00	Schrenk Philip 0,00 USD
26/02/2025	Waiting Room IOE-Videocall - Backend	08:00:00 08:00:00 - 16:00:00	Bastian Seidl 0,00 USD
25/02/2025	(Without Description) IOE-Videocall - OpenAi Whisper	01:12:00 22:03:00 - 23:15:00	Scm120087 0,00 USD
25/02/2025	Gathering Ideas / Bunch of Rewriting IOE-Videocall - Diploma (writing)	03:00:00 19:00:00 - 22:00:00	Schrenk Philip 0,00 USD
25/02/2025	(Without Description) IOE-Videocall - Testing	03:12:00 16:13:00 - 19:25:00	Scm120087 0,00 USD
25/02/2025	Documentation IOE-Videocall - Diploma (writing)	02:00:00 15:00:00 - 17:00:00	Bastian Seidl 0,00 USD
25/02/2025	Warterraum Frontend (bug) IOE-Videocall - Frontend	03:49:16 11:10:44 - 15:00:00	Schrenk Philip 0,00 USD
25/02/2025	Bug Fixing & Waiting Room IOE-Videocall - Backend	04:50:00 10:00:00 - 14:50:00	Bastian Seidl 0,00 USD
24/02/2025	Seite 1 / Changes IOE-Videocall - Diploma (writing)	02:30:00 13:30:00 - 16:00:00	Schrenk Philip 0,00 USD
24/02/2025	Warterraum IOE-Videocall - Reviews	00:30:00 11:30:00 - 12:00:00	Schrenk Philip 0,00 USD
24/02/2025	Bug Fixing IOE-Videocall - Backend	05:05:00 11:00:00 - 16:05:00	Bastian Seidl 0,00 USD
23/02/2025	(Without Description) IOE-Videocall - OpenAi Whisper	01:54:00 15:00:00 - 16:54:00	Scm120087 0,00 USD
23/02/2025	Testing IOE-Videocall - Backend	04:00:00 12:00:00 - 16:00:00	Bastian Seidl 0,00 USD
22/02/2025	Seite 1 / Einleitung IOE-Videocall - Diploma (writing)	02:00:00 14:00:00 - 16:00:00	Schrenk Philip 0,00 USD
21/02/2025	(Without Description) IOE-Videocall - OpenAi Whisper	02:21:00 16:27:00 - 18:48:00	Scm120087 0,00 USD
21/02/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
20/02/2025	Seite 1 / Einleitung IOE-Videocall - Diploma (writing)	01:40:00 13:35:00 - 15:15:00	Schrenk Philip 0,00 USD

19/02/2025	Settings and background IOE-Videocall - Frontend	02:15:33 13:44:27 - 16:00:00	Schrenk Philip 0,00 USD
19/02/2025	(Without Description) IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
19/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
19/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
19/02/2025	(Without Description) IOE-Videocall - Frontend	02:40:00 13:25:00 - 16:05:00	Scm120087 0,00 USD
19/02/2025	Settings and background IOE-Videocall - Frontend	01:00:00 12:30:00 - 13:30:00	Schrenk Philip 0,00 USD
19/02/2025	Bug Fixing IOE-Videocall - Backend	09:30:00 08:00:00 - 17:30:00	Bastian Seidl 0,00 USD
18/02/2025	Waiting Room IOE-Videocall - Backend	10:00:00 08:00:00 - 18:00:00	Bastian Seidl 0,00 USD
17/02/2025	Projektbesprechung mit Zöttl! IOE-Videocall - PRE	01:35:00 12:00:00 - 13:35:00	Schrenk Philip 0,00 USD
15/02/2025	Methodtesting IOE-Videocall - Testing	02:45:00 14:00:00 - 16:45:00	Louismuhr8 0,00 USD
15/02/2025	(Without Description) IOE-Videocall - Reviews	01:43:20 13:27:00 - 15:10:20	Louismuhr8 0,00 USD
14/02/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
13/02/2025	Inputsettings IOE-Videocall - Frontend	01:29:55 14:00:05 - 15:30:00	Schrenk Philip 0,00 USD
13/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:00:00 - 14:00:00	Schrenk Philip 0,00 USD
12/02/2025	(Without Description) IOE-Videocall - OpenAi Whisper	05:42:00 17:04:00 - 22:46:00	Scm120087 0,00 USD
12/02/2025	Naos metting IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Scm120087 0,00 USD
12/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD

12/02/2025	(Without Description) IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
12/02/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
12/02/2025	Transcription testing IOE-Videocall - OpenAi Whisper	00:30:00 13:00:00 - 13:30:00	Scm120087 0,00 USD
10/02/2025	Notifications pt2 IOE-Videocall - Frontend	02:10:30 12:58:29 - 15:08:59	Schrenk Philip 0,00 USD
07/02/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
06/02/2025	Background + Notifications IOE-Videocall - Frontend	03:25:00 16:35:00 - 20:00:00	Schrenk Philip 0,00 USD
06/02/2025	Site Design IOE-Videocall - Frontend	02:05:02 14:47:01 - 16:52:03	Louismuhr8 0,00 USD
05/02/2025	Background IOE-Videocall - Frontend	04:00:00 16:00:00 - 20:00:00	Schrenk Philip 0,00 USD
04/02/2025	Inform on Background Blurs IOE-Videocall - Frontend	03:00:00 17:00:00 - 20:00:00	Schrenk Philip 0,00 USD
31/01/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
30/01/2025	DA-35 IOE-Videocall - Reviews	02:33:00 10:52:00 - 13:25:00	Louismuhr8 0,00 USD
29/01/2025	(Without Description) IOE-Videocall - OpenAi Whisper	01:30:00 14:30:00 - 16:00:00	Scm120087 0,00 USD
29/01/2025	Naos meeting IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Scm120087 0,00 USD
29/01/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
27/01/2025	(Without Description) IOE-Videocall - OpenAi Whisper	02:16:00 16:39:00 - 18:55:00	Scm120087 0,00 USD
24/01/2025	404 Site neu gemacht IOE-Videocall - Frontend	02:59:00 12:47:00 - 15:46:00	Louismuhr8 0,00 USD
24/01/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD

22/01/2025	(Without Description) IOE-Videocall - Backend	04:14:00 15:34:00 - 19:48:00	Scm120087 0,00 USD
22/01/2025	Waiting Room IOE-Videocall - Backend	03:00:00 14:00:00 - 17:00:00	Bastian Seidl 0,00 USD
22/01/2025	Naos meeting IOE-Videocall - Weekly call	00:30:00 13:30:00 - 14:00:00	Bastian Seidl 0,00 USD
22/01/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
22/01/2025	Transcription testing IOE-Videocall - OpenAi Whisper	03:30:00 12:35:00 - 16:05:00	Scm120087 0,00 USD
22/01/2025	Waiting Room IOE-Videocall - Backend	00:55:00 12:35:00 - 13:30:00	Bastian Seidl 0,00 USD
20/01/2025	Max helfen bei transkription IOE-Videocall - Helping Teammates	02:19:00 10:45:00 - 13:04:00	Louismuhr8 0,00 USD
17/01/2025	small changes at login page IOE-Videocall - Frontend	00:50:00 14:07:00 - 14:57:00	Louismuhr8 0,00 USD
17/01/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:45:00 08:00:00 - 08:45:00	Louismuhr8 0,00 USD
17/01/2025	Sprint Planning IOE-Videocall - Sprint Meetings	00:50:00 08:00:00 - 08:50:00	Schrenk Philip 0,00 USD
15/01/2025	Reviewing DA-23 IOE-Videocall - Reviews	01:00:00 17:00:00 - 18:00:00	Schrenk Philip 0,00 USD
15/01/2025	Host-Attendee Feature Frontend IOE-Videocall - Frontend	00:11:04 14:00:00 - 14:11:04	Schrenk Philip 0,00 USD
15/01/2025	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
15/01/2025	Attendee update, Waiting room, refactoring, Reviewing IOE-Videocall - Backend	09:15:00 08:00:00 - 17:15:00	Bastian Seidl 0,00 USD
15/01/2025	attendeeelist request IOE-Videocall - Frontend	05:30:00 08:00:00 - 13:30:00	Schrenk Philip 0,00 USD
14/01/2025	Attendee List and refactoring IOE-Videocall - Backend	03:07:00 20:00:00 - 23:07:00	Bastian Seidl 0,00 USD
14/01/2025	DA-36 Review IOE-Videocall - Reviews	01:00:00 16:15:00 - 17:15:00	Bastian Seidl 0,00 USD

14/01/2025	Attendee update IOE-Videocall - Backend	02:00:00 12:00:00 - 14:00:00	Bastian Seidl 0,00 USD
14/01/2025	DA-33 IOE-Videocall - Reviews	01:20:00 11:45:00 - 13:05:00	Louismuhr8 0,00 USD
14/01/2025	Migrate to certain AWS components IOE-Videocall - Frontend	02:29:49 11:10:11 - 13:40:00	Schrenk Philip 0,00 USD
13/01/2025	Prototype Testing IOE-Videocall - Testing	02:50:00 09:40:00 - 12:30:00	Schrenk Philip 0,00 USD
12/01/2025	Prototyp IOE-Videocall - PRE	03:24:00 20:01:00 - 23:25:00	Bastian Seidl 0,00 USD
12/01/2025	Prototype IOE-Videocall - PRE	04:00:00 18:30:00 - 22:30:00	Scm120087 0,00 USD
12/01/2025	Prototype Meeting and Document IOE-Videocall - PRE	05:00:00 16:00:00 - 21:00:00	Schrenk Philip 0,00 USD
12/01/2025	Buttonclick IOE-Videocall - Backend	00:35:00 14:25:00 - 15:00:00	Louismuhr8 0,00 USD
12/01/2025	Prototype report IOE-Videocall - PRE	00:53:41 14:20:00 - 15:13:41	Louismuhr8 0,00 USD
11/01/2025	Prototype IOE-Videocall - PRE	02:24:00 17:34:00 - 19:58:00	Scm120087 0,00 USD
11/01/2025	Prototype meeting IOE-Videocall - PRE	03:00:33 14:02:30 - 17:03:03	Schrenk Philip 0,00 USD
11/01/2025	Prototype report IOE-Videocall - PRE	01:07:06 14:02:03 - 15:09:09	Louismuhr8 0,00 USD
11/01/2025	Besprechung und Prototyp IOE-Videocall - Meeting	02:52:38 14:01:30 - 16:54:08	Bastian Seidl 0,00 USD
10/01/2025	Prototype report IOE-Videocall - PRE	00:17:14 12:29:50 - 12:47:04	Louismuhr8 0,00 USD
10/01/2025	Prototype report IOE-Videocall - PRE	01:18:06 08:17:33 - 09:35:39	Louismuhr8 0,00 USD
10/01/2025	(Without Description) IOE-Videocall - Sprint Meetings	00:12:00 08:08:00 - 08:20:00	Louismuhr8 0,00 USD
10/01/2025	Standup IOE-Videocall - Sprint Meetings	00:50:00 08:00:00 - 08:50:00	Bastian Seidl 0,00 USD

09/01/2025	DA-50	00:35:00	Louismuhr8
	IOE-Videocall - Reviews	19:01:00 - 19:36:00	0,00 USD
09/01/2025	Prototype report	01:08:00	Louismuhr8
	IOE-Videocall - PRE	15:50:00 - 16:58:00	0,00 USD
09/01/2025	Prototype	00:39:16	Schrenk Philip
	IOE-Videocall - PRE	13:46:41 - 14:25:57	0,00 USD
09/01/2025	(Without Description)	00:15:00	Louismuhr8
	IOE-Videocall - Sprint Meetings	08:00:00 - 08:15:00	0,00 USD
08/01/2025	Join Room Request	03:30:00	Bastian Seidl
	IOE-Videocall - Backend	12:35:00 - 16:05:00	0,00 USD
03/01/2025	Join Room Request	02:30:00	Bastian Seidl
	IOE-Videocall - Backend	17:00:00 - 19:30:00	0,00 USD
02/01/2025	Refactoring	04:00:00	Bastian Seidl
	IOE-Videocall - Backend	14:00:00 - 18:00:00	0,00 USD
28/12/2024	(Without Description)	04:20:00	Scm120087
	IOE-Videocall - OpenAi Whisper	16:05:00 - 20:25:00	0,00 USD
28/12/2024	Change from JS to TS	01:45:00	Bastian Seidl
	IOE-Videocall - Backend	15:15:00 - 17:00:00	0,00 USD
24/12/2024	Changed JS to TS	01:00:00	Scm120087
	IOE-Videocall - Backend	03:38:00 - 04:38:00	0,00 USD
23/12/2024	(Without Description)	01:39:00	Scm120087
	IOE-Videocall - Backend	12:46:00 - 14:25:00	0,00 USD
21/12/2024	(Without Description)	03:55:00	Scm120087
	IOE-Videocall - Backend	17:48:00 - 21:43:00	0,00 USD
21/12/2024	Change from JS to TS	02:10:00	Bastian Seidl
	IOE-Videocall - Backend	12:00:00 - 14:10:00	0,00 USD
19/12/2024	DA-46	01:03:00	Louismuhr8
	IOE-Videocall - Reviews	18:02:00 - 19:05:00	0,00 USD
18/12/2024	Changed JS to TS	02:00:00	Scm120087
	IOE-Videocall - Backend	14:00:00 - 16:00:00	0,00 USD
18/12/2024	naos meeting	01:00:00	Scm120087
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD
18/12/2024	Naos meeting	00:30:00	Bastian Seidl
	IOE-Videocall - Weekly call	13:30:00 - 14:00:00	0,00 USD

18/12/2024	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
18/12/2024	Titelblatt IOE-Videocall - Diploma (writing)	00:26:37 09:14:00 - 09:40:37	Schrenk Philip 0,00 USD
18/12/2024	User List IOE-Videocall - Frontend	00:08:54 09:11:06 - 09:20:00	Schrenk Philip 0,00 USD
18/12/2024	User List IOE-Videocall - Frontend	00:38:47 08:10:00 - 08:48:47	Schrenk Philip 0,00 USD
14/12/2024	test methods IOE-Videocall - Backend	02:11:00 18:02:00 - 20:13:00	Louismuhr8 0,00 USD
12/12/2024	(Without Description) IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
09/12/2024	Google Meet interface IOE-Videocall - Frontend	02:00:00 08:00:00 - 10:00:00	Schrenk Philip 0,00 USD
08/12/2024	OpenAi Whisper Implementation IOE-Videocall - OpenAi Whisper	02:00:00 23:00:00 - 01:00:00 <sup>+1</sup>	Scm120087 0,00 USD
08/12/2024	404 site neu IOE-Videocall - Frontend	02:17:00 13:47:00 - 16:04:00	Louismuhr8 0,00 USD
07/12/2024	microphone button IOE-Videocall - Backend	02:52:27 13:54:33 - 16:47:00	Louismuhr8 0,00 USD
06/12/2024	Sprint Meeting IOE-Videocall - Sprint Meetings	00:50:00 08:00:00 - 08:50:00	Schrenk Philip 0,00 USD
04/12/2024	(Without Description) IOE-Videocall - OpenAi Whisper	03:12:00 18:22:00 - 21:34:00	Scm120087 0,00 USD
04/12/2024	(Without Description) IOE-Videocall - Weekly call	01:00:00 14:35:05 - 15:35:05	Louismuhr8 0,00 USD
04/12/2024	naos meeting IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Scm120087 0,00 USD
04/12/2024	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
04/12/2024	label component/waiting room IOE-Videocall - Frontend	02:15:47 12:44:13 - 15:00:00	Louismuhr8 0,00 USD
03/12/2024	URL Params, Content Share, Member List IOE-Videocall - Frontend	01:50:00 11:00:17 - 12:50:17	Schrenk Philip 0,00 USD

02/12/2024	Join Room Request IOE-Videocall - Backend	03:00:00 15:00:00 - 18:00:00	Bastian Seidl 0,00 USD
29/11/2024	test methods IOE-Videocall - Backend	02:06:00 12:56:00 - 15:02:00	Louismuhr8 0,00 USD
29/11/2024	backend methoden IOE-Videocall - Reviews	02:30:53 12:03:07 - 14:34:00	Louismuhr8 0,00 USD
28/11/2024	Changing Page Layout IOE-Videocall - Frontend	01:10:00 12:25:00 - 13:35:00	Schrenk Philip 0,00 USD
27/11/2024	(Without Description) IOE-Videocall - Weekly call	00:20:00 13:46:08 - 14:06:08	Louismuhr8 0,00 USD
27/11/2024	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
27/11/2024	AsciiDoc setup IOE-Videocall - Diploma (writing)	01:00:44 11:52:34 - 12:53:18	Schrenk Philip 0,00 USD
27/11/2024	OpenAi Whisper Implementation IOE-Videocall - OpenAi Whisper	03:30:00 10:45:00 - 14:15:00	Scm120087 0,00 USD
27/11/2024	First Review IOE-Videocall - Reviews	00:10:00 09:16:21 - 09:26:21	Louismuhr8 0,00 USD
27/11/2024	Adding Button Functionality IOE-Videocall - Frontend	00:25:00 09:00:00 - 09:25:00	Schrenk Philip 0,00 USD
27/11/2024	Reviewing DA-27 IOE-Videocall - Reviews	00:15:00 08:45:00 - 09:00:00	Schrenk Philip 0,00 USD
27/11/2024	waiting room IOE-Videocall - Frontend	00:32:00 08:34:35 - 09:06:35	Louismuhr8 0,00 USD
27/11/2024	Introduction to Tailwind for Louis IOE-Videocall - Helping Teammates	00:15:00 08:30:00 - 08:45:00	Schrenk Philip 0,00 USD
26/11/2024	OpenAi Whisper Implementation IOE-Videocall - OpenAi Whisper	02:00:00 18:00:00 - 20:00:00	Scm120087 0,00 USD
25/11/2024	Reviewing DA-27 IOE-Videocall - Reviews	00:22:00 19:50:00 - 20:12:00	Schrenk Philip 0,00 USD
25/11/2024	waiting room IOE-Videocall - Basic Site	01:03:35 16:37:25 - 17:41:00	Louismuhr8 0,00 USD
23/11/2024	(Without Description) IOE-Videocall - OpenAi Whisper	03:10:00 11:49:00 - 14:59:00	Scm120087 0,00 USD

22/11/2024	Standup	01:00:00	Scm120087
	IOE-Videocall - Sprint Meetings	08:00:00 - 09:00:00	0,00 USD
22/11/2024	Sprintplanning	00:50:00	Bastian Seidl
	IOE-Videocall - Sprint Meetings	08:00:00 - 08:50:00	0,00 USD
22/11/2024	Sprint Meeting	00:50:00	Schrenk Philip
	IOE-Videocall - Sprint Meetings	08:00:00 - 08:50:00	0,00 USD
21/11/2024	Added some Buttons	00:45:00	Schrenk Philip
	IOE-Videocall - Basic Site	13:25:00 - 14:10:00	0,00 USD
21/11/2024	Backend Refactoring	01:40:00	Bastian Seidl
	IOE-Videocall - Backend	11:45:00 - 13:25:00	0,00 USD
20/11/2024	OpenAi Whisper Implementation	01:00:00	Scm120087
	IOE-Videocall - OpenAi Whisper	16:00:00 - 17:00:00	0,00 USD
20/11/2024	Weekly Call	01:00:00	Schrenk Philip
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD
20/11/2024	Prettier Site	00:53:16	Schrenk Philip
	IOE-Videocall - Basic Site	12:51:44 - 13:45:00	0,00 USD
20/11/2024	impressum	04:55:00	Louismuhr8
	IOE-Videocall - Frontend	12:45:00 - 17:40:00	0,00 USD
20/11/2024	OpenAi Whisper Implementation	01:40:00	Scm120087
	IOE-Videocall - OpenAi Whisper	11:45:00 - 13:25:00	0,00 USD
19/11/2024	Waitingroom	05:17:00	Louismuhr8
	IOE-Videocall - Frontend	11:45:00 - 17:02:00	0,00 USD
19/11/2024	Chime Setup	01:05:53	Schrenk Philip
	IOE-Videocall - Basic Site	11:14:07 - 12:20:00	0,00 USD
15/11/2024	Small changes, login page	00:48:00	Louismuhr8
	IOE-Videocall - Frontend	15:04:00 - 15:52:00	0,00 USD
14/11/2024	Setting up chime	01:07:53	Schrenk Philip
	IOE-Videocall - Basic Site	11:52:07 - 13:00:00	0,00 USD
13/11/2024	(Without Description)	01:30:00	Scm120087
	IOE-Videocall - OpenAi Whisper	14:30:00 - 16:00:00	0,00 USD
13/11/2024	Installing Chime	02:05:37	Schrenk Philip
	IOE-Videocall - Basic Site	13:45:23 - 15:51:00	0,00 USD
13/11/2024	(Without Description)	01:00:00	Louismuhr8
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD

13/11/2024	naos meeting IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Scm120087 0,00 USD
13/11/2024	Weekly Call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
13/11/2024	Installing Chime IOE-Videocall - Basic Site	00:59:00 12:31:00 - 13:30:00	Schrenk Philip 0,00 USD
09/11/2024	Wireframes IOE-Videocall - PRE	01:54:00 12:59:00 - 14:53:00	Louismuhr8 0,00 USD
08/11/2024	(Without Description) IOE-Videocall - Testing	02:13:00 11:49:00 - 14:02:00	Scm120087 0,00 USD
07/11/2024	DA-15 IOE-Videocall - Backend	01:40:00 08:00:00 - 09:40:00	Bastian Seidl 0,00 USD
06/11/2024	naos meeting IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Scm120087 0,00 USD
06/11/2024	meeting mit naos IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Bastian Seidl 0,00 USD
06/11/2024	(Without Description) IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Louismuhr8 0,00 USD
06/11/2024	weekly call IOE-Videocall - Weekly call	01:00:00 13:30:00 - 14:30:00	Schrenk Philip 0,00 USD
06/11/2024	Created routes IOE-Videocall - Basic Site	00:59:16 13:00:44 - 14:00:00	Schrenk Philip 0,00 USD
06/11/2024	Install Amazon CLI IOE-Videocall - Setup Backend	00:27:12 12:52:48 - 13:20:00	Schrenk Philip 0,00 USD
06/11/2024	Install Amazon CLI IOE-Videocall - Setup Backend	00:12:00 12:40:00 - 12:52:00	Schrenk Philip 0,00 USD
06/11/2024	Install Amazon CLI IOE-Videocall - Setup Backend	00:13:08 12:27:00 - 12:40:08	Schrenk Philip 0,00 USD
06/11/2024	DA-15 IOE-Videocall - Backend	01:30:00 12:00:00 - 13:30:00	Bastian Seidl 0,00 USD
06/11/2024	Install Amazon CLI IOE-Videocall - Setup Backend	00:27:34 11:00:00 - 11:27:34	Schrenk Philip 0,00 USD
06/11/2024	Created routes IOE-Videocall - Basic Site	00:29:00 10:45:00 - 11:14:00	Schrenk Philip 0,00 USD

31/10/2024	Tests	02:57:00	Louismuhr8
	IOE-Videocall - Backend	17:04:00 - 20:01:00	0,00 USD
25/10/2024	Systemarchitektur	01:00:00	Scm120087
	IOE-Videocall - Post-Kickoff	13:00:00 - 14:00:00	0,00 USD
25/10/2024	(Without Description)	00:10:00	Louismuhr8
	IOE-Videocall - Post-Kickoff	12:44:56 - 12:54:56	0,00 USD
25/10/2024	Sprintplanning, Retro	00:29:00	Bastian Seidl
	IOE-Videocall - Sprint Meetings	12:04:00 - 12:33:00	0,00 USD
25/10/2024	(Without Description)	00:30:00	Louismuhr8
	IOE-Videocall - Sprint Meetings	12:00:00 - 12:30:00	0,00 USD
25/10/2024	Sprintplanning	00:30:00	Scm120087
	IOE-Videocall - Sprint Meetings	12:00:00 - 12:30:00	0,00 USD
25/10/2024	sprint meeting	00:21:00	Schrenk Philip
	IOE-Videocall - Sprint Meetings	12:00:00 - 12:21:00	0,00 USD
25/10/2024	Arbeit an Post Kickoff	01:00:00	Bastian Seidl
	IOE-Videocall - Post-Kickoff	08:00:00 - 09:00:00	0,00 USD
25/10/2024	Post-Kickoff	00:57:00	Schrenk Philip
	IOE-Videocall - Post-Kickoff	08:00:00 - 08:57:00	0,00 USD
24/10/2024	Arbeit an Post Kickoff	01:40:00	Bastian Seidl
	IOE-Videocall - Post-Kickoff	08:00:00 - 09:40:00	0,00 USD
23/10/2024	SWOT Analyse	01:00:00	Louismuhr8
	IOE-Videocall - Post-Kickoff	14:45:00 - 15:45:00	0,00 USD
23/10/2024	Systemarchitektur	01:30:00	Scm120087
	IOE-Videocall - Post-Kickoff	14:30:00 - 16:00:00	0,00 USD
23/10/2024	refine post-kickoff	01:53:29	Schrenk Philip
	IOE-Videocall - Post-Kickoff	13:52:51 - 15:46:20	0,00 USD
23/10/2024	Naos call	01:00:00	Bastian Seidl
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD
23/10/2024	naos meeting	01:00:00	Scm120087
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD
23/10/2024	Weekly Meeting	00:23:00	Louismuhr8
	IOE-Videocall - Weekly call	13:30:00 - 13:53:00	0,00 USD
23/10/2024	Weekly call	01:00:00	Schrenk Philip
	IOE-Videocall - Weekly call	13:30:00 - 14:30:00	0,00 USD

23/10/2024	fix imports IOE-Videocall - import component library	00:55:00 12:05:00 - 13:00:00	Schrenk Philip 0,00 USD
22/10/2024	Anforderungsdefinitionen & Zeitaufzeichnung IOE-Videocall - Post-Kickoff	01:20:00 11:40:00 - 13:00:00	Louismuhr8 0,00 USD
21/10/2024	started importing IOE-Videocall - import component library	01:29:45 09:04:15 - 10:34:00	Schrenk Philip 0,00 USD
20/10/2024	Post-Kickoff IOE-Videocall - PRE	01:16:00 17:48:00 - 19:04:00	Louismuhr8 0,00 USD
18/10/2024	(Without Description) IOE-Videocall - OpenAi Whisper	03:28:00 15:58:00 - 19:26:00	Scm120087 0,00 USD
18/10/2024	Meeting Protokoll IOE-Videocall	00:10:02 11:13:44 - 11:23:46	Louismuhr8 0,00 USD
18/10/2024	Standup IOE-Videocall - Sprint Meetings	00:10:00 08:05:00 - 08:15:00	Scm120087 0,00 USD
18/10/2024	Standup IOE-Videocall - Sprint Meetings	00:50:00 08:00:00 - 08:50:00	Bastian Seidl 0,00 USD
18/10/2024	sprint meeting IOE-Videocall - Sprint Meetings	00:50:00 08:00:00 - 08:50:00	Schrenk Philip 0,00 USD
16/10/2024	KPI's IOE-Videocall - Post-Kickoff	00:45:00 16:15:00 - 17:00:00	Scm120087 0,00 USD
16/10/2024	small methods IOE-Videocall - Backend	02:46:00 13:02:00 - 15:48:00	Louismuhr8 0,00 USD
14/10/2024	Post-Kickoff IOE-Videocall - Post-Kickoff	00:32:00 08:32:57 - 09:04:57	Schrenk Philip 0,00 USD
14/10/2024	Meeting-Protokolle, Kanban, Sprint Parameter IOE-Videocall - Post-Kickoff	00:50:00 08:00:00 - 08:50:00	Bastian Seidl 0,00 USD
11/10/2024	Small test IOE-Videocall - Backend	00:49:26 13:56:34 - 14:46:00	Louismuhr8 0,00 USD
11/10/2024	Meeting Protokoll IOE-Videocall	00:14:54 13:24:13 - 13:39:07	Louismuhr8 0,00 USD
11/10/2024	Wireframe 2 IOE-Videocall	00:18:00 13:05:00 - 13:23:00	Louismuhr8 0,00 USD
11/10/2024	Meeting Protokoll IOE-Videocall	00:10:02 11:13:44 - 11:23:46	Louismuhr8 0,00 USD

10/10/2024	Wireframe	00:30:00	Louismuhr8
	IOE-Videocall	16:00:00 - 16:30:00	0,00 USD
10/10/2024	404 Site	02:06:34	Louismuhr8
	IOE-Videocall - Frontend	13:00:00 - 15:06:34	0,00 USD
10/10/2024	Post-Kickoff	00:12:00	Schrenk Philip
	IOE-Videocall - Post-Kickoff	11:44:59 - 11:56:59	0,00 USD
10/10/2024	Weekly Meeting	00:25:00	Louismuhr8
	IOE-Videocall - Weekly call	08:46:18 - 09:11:18	0,00 USD
09/10/2024	switch to gitlab	00:23:29	Schrenk Philip
	IOE-Videocall - switch to gitlab	13:38:27 - 14:01:56	0,00 USD
09/10/2024	Weekly call	01:00:00	Schrenk Philip
	IOE-Videocall - Weekly call	13:20:00 - 14:20:00	0,00 USD
07/10/2024	Post-Kickoff	00:38:53	Schrenk Philip
	IOE-Videocall - Post-Kickoff	13:32:01 - 14:10:54	0,00 USD
07/10/2024	naos meeting	00:30:00	Scm120087
	IOE-Videocall - naos meeting #1	13:00:00 - 13:30:00	0,00 USD
03/10/2024	Weekly Meeting	00:45:00	Louismuhr8
	IOE-Videocall - Weekly call	08:46:49 - 09:31:49	0,00 USD
02/10/2024	Backend Setup	02:00:00	Bastian Seidl
	IOE-Videocall - Setup Backend	14:00:00 - 16:00:00	0,00 USD
02/10/2024	nextjs setup	01:30:05	Schrenk Philip
	IOE-Videocall - Setup nextjs on our github	13:37:55 - 15:08:00	0,00 USD
02/10/2024	Meeting with Naos	01:00:00	Bastian Seidl
	IOE-Videocall - naos meeting #1	13:00:00 - 14:00:00	0,00 USD
02/10/2024	IOE-Videocall Kickoff	01:00:00	Schrenk Philip
	IOE-Videocall - naos meeting #1	13:00:00 - 14:00:00	0,00 USD