Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Bachelorarbeit (Informatik)

Automatische Transkription von Interviews

Autoren	Nicolas Elvis Eckhart Mathias Richard Marxer
Hauptbetreuung	Mark Cieliebak

Datum	05.06.2019

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

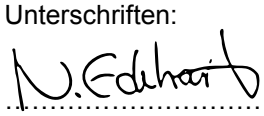
Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

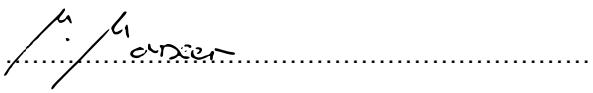
Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Winterthur, 06.06.2019

Unterschriften:





Zusammenfassung

Die Transkription gesprochener Konversationen ist ein Anwendungsfall, der in unzähligen Gebieten wie der Politik, der Medizin oder der Geschäftswelt vorhanden ist. Seien es Meetings, Interviews im Medienumfeld oder Bewerbungsgespräche, die meisten Unternehmen berufen sich auf menschliche Transkription, was nicht nur ein kosten-, sondern auch ein zeitaufwändiger Prozess ist. Mehr und mehr zeigt sich jedoch automatisierte Audio-Transkription als realistische Alternative. Die Auswahl an Programmen, die ohne technische Kenntnisse auf der Seite des Benutzers zur Audio-Transkription verwendet werden können, ist jedoch klein und viele wichtige Funktionen, wie Sprecher- oder Pausenerkennung, werden nicht, oder nur minimal unterstützt.

Ziel dieser Arbeit war es, basierend auf den Erkenntnissen und Ergebnissen aus einer vorangegangenen Projektarbeit, ein System für die automatische Transkription von Interviews zu entwickeln. Es sollen Audio-Dateien der Anwendung übergeben werden können und das Transkript zurückerhalten werden. Anschliessend soll der Benutzer in einem integrierten Editor Text-, sowie Spracherkennungsfehler korrigieren können. Weiter soll das Transkript im Editor durch hervorheben einzelner Wörter mit der Audio-Datei synchronisiert werden und es sollen dem Benutzer Möglichkeiten zum Export des Transkriptes bereitgestellt werden.

Die Software wurde mittels einem agilen Vorgehensmodell, unter der Verwendung von Elementen aus Scrum und Kanban, in einwöchigen Sprints entwickelt. Jede Woche wurden die im vorgehenden Sprint erarbeiteten Resultate präsentiert, wonach die verbleibenden Funktionen diskutiert und neu priorisiert wurden. Anhand dessen wurde bestimmt, welche User Stories als nächstes in Angriff zu nehmen waren.

Das bereits bestehende «Proof of Concept» wurde weitgehend überarbeitet und stark verbessert. Die Software «InterScriber» beinhaltet alle wichtigen Grundfunktionen, damit ein Arbeiten an einem transkribierten Interview möglich ist. Als alternative Transkriptions-Schnittstelle zur Google-Cloud-API wurde ausserdem die IBM-Cloud-API eingebunden, und Deutsch wurde als unterstützte Sprache hinzugefügt. Während die Funktionalität zum Transkribieren von Interviews nun gegeben ist, bleiben jedoch noch einige Funktionen zu implementieren, um aus dem Prototypen ein marktaugliches Produkt zu machen.

Abstract

The process of transcribing conversations is a use case relevant in dozens of fields from politics, through medicine as well as the business world. Most companies rely on manual transcription, a process both costly and time intensive. Recently, automatic transcription of audio has shown itself as a more and more viable alternative. While this is the case, the selection of software that can be used without deeper technical understanding is still limited and many key features are still not widely supported.

The aim of this project was to implement a system for the automatic transcription of interviews, based on the knowledge and results of a prior project. The software should take in audio files and provide the user with a transcript of the conversation. Following that, the user can correct mistakes both relating to the transcribed text as well as the recognized speakers in an integrated editor. Furthermore, the transcript within the editor must be synchronized with the audio by highlighting the words being played back. Lastly, the user should be provided with multiple export options for the transcript.

The program was implemented using an agile approach in single week sprints, using elements from both Scrum and Kanban. Each week the results of the previous sprint were presented, after which the remaining features were discussed and reprioritized. Based on this, the relevant user stories where selected for completion in the next sprint.

The existing proof of concept was extensively reworked and improved. The application «InterScriber» contains all the important features to allow for working on a transcribed interview. As an alternative transcription interface to the Google-Cloud-API, the IBM-Cloud-API was intergrated, and German interviews are now fully supported. While the functionality for transcribing interviews has been fully implemented, several features still need to be developed for full marketability.

Vorwort

Unsere Namen sind Nicolas Eckhart und Mathias Marxer. Während dem letzten Semester haben wir im Rahmen dieser Arbeit unsere im Studium gelernten Kenntnisse festigen und vieles an neuen Information in den Feldern der Künstlichen Intelligenz, der Linguistik, der Informatik und des UI-Designs sammeln können. In unserer Gruppe haben wir beide an allen Aspekten dieser Arbeit mitgewirkt, sei es Recherche, Programmieren, Testen oder Schreiben des Berichtes.

Vor einem Jahr, als das Thema für die Projektarbeit zu wählen war, hatten wir ursprünglich eine andere Arbeit im Bereich der Textanalyse und der künstlichen Intelligenz im Sinn, jedoch haben die Umstände dieses Projekt nicht zugelassen. Als wir gemeinsam die alternativen Ideen der Mitarbeiter des Instituts für angewandte Informationstechnologie besprachen, war es uns schnell klar, dass wir dieses Thema wählen wollten.

Wir wollen uns ganz herzlich bei Prof. Dr. Mark Cieliebak bedanken. Nicht nur für die Hilfe bei der Ideensuche, sondern selbstverständlich auch für die Betreuung der Bachelorarbeit, wie auch der Projektarbeit im vergangenen Jahr. Weiter wollen wir uns beim Institut für angewandte Informationstechnologie für die Deckung der entstandenen Kosten und bei der Zürcher Hochschule für angewandte Wissenschaften für die erlaubte Nutzung der technischen und räumlichen Infrastruktur bedanken. Und zuletzt bei Malgorzata Ulasik, die uns in der Projektarbeit tatkräftig unterstützt hat und uns half, den Grundstein für diese Bachelorarbeit zu legen.

Winterthur, 05. Juni 2019

INHALTSVERZEICHNIS

ZUSAMMENFASSUNG **1**

ABSTRACT **2**

VORWORT **3**

1 EINLEITUNG **6**

1.1 AUSGANGSLAGE 6
1.2 ZIELSETZUNG 6
1.2.1 Spezifikation *6*
1.2.2 Aufbau der Arbeit *7*
1.2.3 Terminologie *7*

2 THEORETISCHE GRUNDLAGEN **8**

2.1 SPRACHERKENNUNG 8
2.1.1 Linguistische Grundlage *8*
2.1.2 Merkmalgewinnung *8*
2.1.3 Worterkennung *8*
2.1.4 Herausforderungen *9*
2.2 DIARISIERUNG 9
2.2.1 Segmentierung *9*
2.2.2 Sprecher Clustering *9*

3 STAND DER TECHNIK **10**

3.1 KONKURRENZANALYSE 10
3.1.1 Wreally *10*
3.1.2 TranscribeMe *10*
3.1.3 Scribie *11*
3.1.4 Rev *11*
3.1.5 GoTranscript *11*
3.1.6 Sonix *11*
3.1.7 Temi *12*
3.1.8 Trint *13*
3.1.9 Fazit *13*
3.2 INDUSTRIESTANDARDS VON TRANSKRIPT FORMATEN 14
3.2.1 Google-Cloud-API *14*
3.2.2 IBM-Cloud-API *14*
3.2.3 Microsoft Azure *15*
3.2.4 Temi API *16*
3.2.5 SeerNet – Multi-Speaker-Diarization API *16*
3.2.6 remeeting ASR API *17*
3.2.7 Fazit *17*

4 VORGEHEN **18**

4.1 BESTEHENDES PRODUKT 18
4.2 ANFORDERUNGSANALYSE 19
4.2.1 Funktionale Anforderungen *20*
4.2.2 Nicht-funktionale Anforderungen *21*
4.3 DESIGN 22
4.3.1 Physische Architektur *22*
4.3.2 Logische Architektur *23*
4.4 IMPLEMENTATION 28
4.5 HERAUSFORDERUNGEN 34
4.5.1 Fehlerhafte deutsche Diarisierung *34*
4.5.2 Zeitstempel *34*
4.5.3 Externes Backend *35*
4.5.4 Performanz *35*

5 EVALUATION 36

5.1 TRANSKRIPTIONSAUFWAND 36
5.2 BENUTZER FEEDBACK 37

6 DISKUSSION UND AUSBLICK 38

7 VERZEICHNISSE 39

7.1 LITERATURVERZEICHNIS 39
7.2 BILDVERZEICHNIS 41
7.3 TABELLENVERZEICHNIS 42

8 ANHANG 43

8.1 OFFIZIELLE AUFGABENSTELLUNG, PROJEKTAUFTRAG 43
8.2 PROJEKTMANAGEMENT 44
8.2.1 Projektplan 44
8.2.2 Product Backlog 47
8.3 TECHNISCHE DOKUMENTATION 50
8.3.1 Installation und Entwicklung 50
8.3.2 Übersicht der Backend Dateien 52
8.3.3 Übersicht der Frontend Dateien 53
8.3.4 Vollständiges Klassendiagramm 54
8.3.5 Backend HTTP Schnittstelle 55
8.4 USB STICK 58

1 Einleitung

1.1 Ausgangslage

Die Transkription gesprochener Konversationen ist ein Anwendungsfall, der in unzähligen Gebieten wie der Politik, der Medizin oder der Geschäftswelt vorhanden ist. Seien es Meetings, Interviews im Medienumfeld oder Bewerbungsgespräche, die meisten Unternehmen berufen sich auf menschliche Transkription, was nicht nur ein kosten-, sondern auch ein zeitaufwändiger Prozess ist. Mehr und mehr zeigt sich jedoch automatisierte Audio-Transkription als realistische Alternative. Mit Weiterentwicklungen in Feldern wie Natural Language Processing, Audio-Analyse und dem maschinellen Lernen, konnten Tech-Konzerne wie IBM, Google, Amazon und Microsoft Speech-to-Text Dienste mit hoher Genauigkeit und tiefen Preisen der Öffentlichkeit zugänglich machen.

Während sich diese Dienste gut für Enterprise-Lösungen eignen, sind sie für Endbenutzer nicht optimal. Die Auswahl an Programmen, die ohne technische Kenntnisse auf der Seite des Benutzers zur Audio-Transkription verwendet werden können, ist klein und viele wichtige und nötige Funktionen zur automatischen Transkription, wie Sprecher- oder Pausenerkennung, werden nicht oder noch nicht in allen Sprachen unterstützt.

Diese Arbeit baut auf einer vorangegangenen Arbeit auf [1] und die Theorie hinter der automatischen Interview Transkription basiert auf der Literatur von Pfister et al. [2], Kolossa [3], Schukat-Talamazzini [4], Hallerbach [5], Schulz et al. [6], Kemp et al. [7] und Anguera [8].

1.2 Zielsetzung

Ziel dieser Arbeit ist es, basierend auf den Erkenntnissen und Ergebnissen aus einer vorangegangenen Projektarbeit, ein System für die automatische Transkription von Interviews zu entwickeln. Dazu gehört das Hochladen der Audio-Datei, das automatische Transkribieren von englischen und deutschen Aufnahmen, das Korrigieren des erzeugten Transkripts und dessen Export. Dabei wird die bestehende Anwendung der vorangegangenen Projektarbeit weiterentwickelt und mit neuen Funktionen erweitert. Die komplette Aufgabenstellung, die dieser Arbeit zu Grunde liegt, kann dem Anhang (vgl. Abschnitt 8.1) entnommen werden.

1.2.1 Spezifikation

Das Resultat dieser Arbeit soll sein, dass die Anwendung alle nötigen Funktionen beinhaltet, welche für ein Arbeiten mit dem Transkript wichtig sind. Es soll darauf geachtet werden, dass die Software einfach zu installieren und zu benutzen ist. Dazu gehört auch, dass es einfach sein soll, eine Audio-Datei hochzuladen und diese zu transkribieren. Anschliessend soll der Text korrigiert, neue Sprecher hinzugefügt und bestehende geändert werden können. Zusätzlich soll die Audio-Datei angehört und direkt im Transkript ersichtlich sein, bei welcher Stelle im Text man sich befindet. Das Transkript soll in ein Word- oder Excel Dokument exportiert werden können und der Anwender soll auch die Möglichkeit haben, die Arbeit zu speichern.

1.2.2 Aufbau der Arbeit

Zu Beginn werden die theoretischen Grundlagen von Spracherkennung und Diarisierung erklärt (vgl. Kapitel 2). Nachfolgend wird in Kapitel 3 eine Konkurrenzanalyse durchgeführt, um bereits bestehende Produkte zu vergleichen und aufzuzeigen, was der aktuelle Stand der Technik ist und was bei diesen Produkten fehlt. Zudem werden in diesem Kapitel die typischen Transkript Formate der verfügbaren APIs detailliert. Nachfolgend wird das Vorgehen zur Entwicklung der Software beschrieben (vgl. Kapitel 4), das Design erklärt und das Resultat der Arbeit ausführlich erläutert. Schlussendlich wird im Kapitel 5 noch eine kurze Evaluation mit Feedback von Testern durchgeführt, bevor im Kapitel 6 das Resultat und Vorgehen diskutiert und einen Ausblick auf mögliche Weiterentwicklungen gegeben wird.

1.2.3 Terminologie

Diarisierung, Diarization, Sprecher Diarization, Speaker Diarization, Sprechertrennung	Erkennen und unterscheiden von verschiedenen, unbekannten Sprechern in einer Audio-Datei.
Spracherkennung, Speech-to-Text (STT), (Automatic) Speech Recognition (ASR)	Die gesprochene Sprache in einer Audio-Datei zu Text transkribieren.
«Full verbatim» / «strict verbatim» Transkript	Wortwörtliches Transkript inklusive Füllwörter, Wiederholungen, etc.
Sprechertrennung	Überbegriff für Sprecher-Verifikation, Sprecher-Identifikation und Sprecher Diarization.
Sprachmodell	Regelsystem welches die Struktur einer Sprache beschreibt.
Spektralbereich / Frequenzspektrum	Kontinuierliches Spektrum das mittels einer Fourier-Transformation auf einem Audio-Signal berechnet wird.
Merkmal, Feature (in Bezug auf Audio)	Verschiedene Eigenschaften, die dazu dienen, Audio zu charakterisieren.
Nulldurchgangsrate, Zero Crossing Rate	Feature, welches wenn tief auf Sprachsegmente und wenn hoch auf Ruhe hinweist.
Intersprechervariabilität	Unterschiede in der Aussprache (Dialekt, Betonung, etc.) zwischen verschiedenen Sprechern.
Intrasprechervariabilität	Unterschiede in der Aussprache eines Wortes desselben Sprechers.
Utterance, Äusserung	Ist ein Teil einer Unterhaltung, wenn ein Sprecher etwas sagt.
parallel Utterance, parallele Äusserung	Eine Äusserung mit dem Unterschied, dass mehrere Sprecher gleichzeitig sprechen
Zuversicht, Confidence (in Bezug auf ein transkribiertes Wort)	Ein Wert, der einem Wort zugewiesen wird und aussagt, wie sicher sich ein System ist, dass das Wort richtig erkannt wurde.

2 Theoretische Grundlagen

In dem dieser Arbeit vorgegangenen Bestreben wurden die theoretischen Grundlagen der maschinellen Transkription von Gesprächen weitgehend recherchiert und beschrieben [1]. Während der Fokus dieser Arbeit nicht primär auf den theoretischen Hintergründen liegt, werden der Vollständigkeit halber dennoch die zentralen Konzepte noch einmal zusammenfassend aufgeführt.

Die automatische Transkription einer Konversation hat zwei zugrundeliegende Aspekte: Die Spracherkennung, oder Umwandlung des gesprochenen Wortes in die textliche Form und die Sprecher Diarisierung, wodurch zwischen den Gesprächsteilnehmern differenziert wird. Im Folgenden werden diese beiden Themen ausführlich dargestellt.

2.1 Spracherkennung

Der Prozess der Spracherkennung ist der des Umwandelns von Sprache in Form eines Audiosignals in eine textuelle Form. Wie Pfister und Kaufmann [2] aufweisen, ist Spracherkennung ein interdisziplinäres Gebiet, basierend auf der Akustik, Informatik, Linguistik, Signalverarbeitung und Statistik.

2.1.1 Linguistische Grundlage

Die Struktur der gesprochenen Sprache wird auf der phonemischen Ebene als Abfolge von Phonemen oder Lauten, den kleinsten Einheiten der Sprache, definiert. Da sich die Laute in einem physikalischen Sprachsignal jedoch weder in zeitlicher Hinsicht noch bezüglich deren Charakteristiken klar voneinander abgrenzen lassen, ist eine direkte Abbildung der Laute einer Aufnahme auf gespeicherte Signale unmöglich [2]. Abhilfe schaffen hier statistische Modelle, mit welchen anhand spezifischer Merkmale eines digitalen Signales eine Worterkennung trotzdem durchgeführt werden kann.

2.1.2 Merkmalgewinnung

Die Merkmale, auch Features genannt, welche ein Sprachsignal charakterisieren, werden aus dem digitalisierten Signal extrahiert. In einem ersten Schritt wird das Signal gefiltert, um zwischen aktiver Sprache und Hintergrundgeräuschen zu unterscheiden. Beispielsweise werden für das Erkennen von Stille und Geräuschen die Features der Nulldurchgangsrate und der Signal-Energie analysiert [2]. Für die eigentliche Spracherkennung wird das Signal in den Spektralbereich transformiert und in überlappende Segmente (Frames) von 5-30 Millisekunden aufgeteilt, worauf auf den einzelnen Frames unter anderem die Cepstrum-Koeffizienten, Energiewerte und Frequenzbereichsstärken untersucht werden [2] – [5].

2.1.3 Worterkennung

Anhand der berechneten Feature-Vektoren lässt sich das jeweils wahrscheinlichste Wort nun erschliessen. Hidden Markov Modelle (HMM) sind die weitverbreitetste Technik für die Worterkennung, jedoch lässt sich die Problemstellung auch mit Deep Neural Networks oder Hybridmodellen angehen [2]. Ein wichtiges Mittel zur Erhöhung der Genauigkeit der Worterkennung sind Sprachmodelle. Indem ein linguistisches Regelsystem einführt wird, kann die Wahrscheinlichkeit gewisser Wortkombinationen berechnet, und somit der Such-Raum eingeschränkt werden [6].

2.1.4 Herausforderungen

Bei der maschinellen Spracherkennung müssen verschiedene Faktoren berücksichtig werden, welche sich auf die Qualität der Resultate auswirken können. Technische Faktoren wie die Raumakustik, die Einstellung des Aufnahmesystems oder die elektrische Übertragung des Signals können das Audiosignal verändern indem Frequenzbereiche gedämpft oder verstärkt werden, was wiederum die Qualität der Spracherkennung beeinträchtigen kann. Ein weiteres Hindernis sind die menschlichen Faktoren der Inter- und Intrasprechervariabilität. Nicht nur unterscheiden sich Sprecher bezüglich der Physiologie des Vokaltraktes oder Dialekts, sondern die Aussprache eines einzelnen Sprechers kann sich aufgrund dessen Sprechgewohnheiten oder seines emotionalen Zustands verändern. Zuletzt ist ebenfalls die Struktur der Sprache zu beachten. So lassen sich zum Beispiel das Sprachsignal eines Homophons nicht eindeutig auf das korrekte Wort zuweisen. Für solche Probleme ist zusätzliches Wissen über die Sprache notwendig (vgl. Abschnitt 2.1.3).

2.2 Diarisierung

Die Diarisierung (Diarization) ist ein Teil des Problems der Sprechertrennung. Dabei sollen die dem System unbekannten Sprecher eines Gesprächs im Audiosignal erkannt und unterschieden werden. Im Folgenden wird der Diarisierungs-Prozess genauer beschrieben.

2.2.1 Segmentierung

Im ersten Schritt wird versucht die Sprecherwechsel im Audiosignal zu identifizieren und diese in akustisch gleichwertige Segmente aufzuteilen. Dafür wird analog zur Spracherkennung eine Merkmalgewinnung durchgeführt (vgl. Abschnitt 2.1.2), wonach mit den erarbeiteten Features das Signal segmentiert werden kann. Es bestehen verschiedene Ansätze für die Segmentierung, jedoch weisen laut einem Vergleich von Kemp et al. [7] modell- und metrikbasierte Entscheidungen die besten Resultate auf. Während modellbasierte Algorithmen exakter sind, haben metrikbasierte Algorithmen den Vorteil, dass keine Modelle im Voraus vorhanden sein müssen, da mittels einer Distanz Funktion bestimmt wird, ob zwischen zwei Frames segmentiert werden muss oder nicht [8].

2.2.2 Sprecher Clustering

Nachdem das Signal aufgeteilt wurde, werden die einzelnen Segmente mittels einer probabilistischen Clustering-Methode entsprechend ihrer Ähnlichkeit in bestimmten Metriken gruppiert. Hierbei können entweder Gaussian Mixture Modelle (GMM) oder Hidden Markov Modelle (HMM) verwendet werden. Schlussendlich sollten alle Sprach-Segmente derselben Sprecher zusammen gruppiert sein, sodass die Anzahl Sprecher und deren Segmente gegeben ist.

3 Stand der Technik

Im Folgenden werden als erstes die bestehenden Transkriptions-Dienste und Lösungen für Endbenutzer untersucht. Darauf werden die in der Industrie verbreiteten Formate für Transkripte beschrieben.

3.1 Konkurrenzanalyse

Der Fokus bei der Analyse der einzelnen Anbieter liegt auf den angebotenen Leistungen, und spezifisch auf ASR. Dabei werden die Preise, die unterstützten Sprachen, die angepriesene Genauigkeit und weitere Features aufgeführt. Menschliche und hybride Transkriptions-Dienste und Tools werden der Vollständigkeit halber ebenfalls erwähnt.

3.1.1 Wreally

Die Plattform bietet zwei Dienste an: Automatische Transkription und unterstützte menschliche Transkription [9]. Für klare Aufnahmen bietet Wreally maschinelle Transkription mit einer Korrektheit von 90% an. Unterstützt werden eine grosse Menge an Sprachen [10], obwohl viele noch als «experimentell» markiert sind. Alternativ wird ein Editor angeboten, um die Transkription von Hand zu vereinfachen. Dabei werden Audio und Editor nebeneinander dargestellt und die Geschwindigkeit kann reguliert werden, allerdings ist das Ganze vom ASR Service losgekoppelt.

Genauigkeit	90% oder mehr
Sprachen	Englisch, Deutsch, Hebräisch, Italienisch, Portugiesisch, Spanisch und weitere experimentelle Sprachen
Preise	0.1 USD pro Minute Audio
Weitere Features	Diarization

Tabelle 1: ASR Zusammenfassung - Wreally

3.1.2 TranscribeMe

TranscribeMe ist ein Transkriptions-Provider für Grosskunden wie auch Kleinkunden. Angeboten wird menschliche Transkription mit verschiedenen Genauigkeits-Garantien und Lieferzeiten, aber auch automatische Transkription. Die menschliche Transkription beginnt bei $0.76 / Minute mit einer Genauigkeit von 98% und einer Lieferzeit von 48h und geht bis zu «strict verbatim» für $2.00 / Minute und einer garantierten Korrektheit von 100% [11].

Genauigkeit	90 - 95% bei guter Audio-Qualität
Sprachen	Englisch
Preise	0.1 USD pro Minute Audio
Weitere Features	Diarization (+ $0.15 / Minute), Zeitstempel (+ $0.15 / Minute)

Tabelle 2: ASR Zusammenfassung - TranscribeMe

3.1.3 Scribie

Scribie ist ein Online-Dienst der die manuelle Transkription von Audio- und Videodateien, wie auch ASR anbietet. Die menschliche Transkription kostet $0.80 / Minute und es wird eine Genauigkeit von 99% oder mehr garantiert. Nicht-amerikanische Akzente, tiefe Audioqualität und auch optionale Zusatzleistungen wie «strict verbatim» oder SRT/VTT Dateien erhöhen den Minutenpreis um je $0.50 [12].

Genauigkeit	80 – 95%
Sprachen	Englisch
Preise	0.1 USD pro Minute Audio
Weitere Features	Keine

Tabelle 3: ASR Zusammenfassung - Scribie

3.1.4 Rev

Rev ist ein Dienst der ausschliesslich die manuelle Transkription von Englischen Audio- und Videodateien anbietet. Der Preis ist $1.00 pro Minute und die Genauigkeit 99%, bei einer Lieferzeit von ca. 20 Stunden [13].

3.1.5 GoTranscript

GoTranscript bietet nur manuelle Transkriptionen an. Die Genauigkeits-Garantie ist 99% und die Lieferzeit ca. 6 Stunden nachdem die Datei hochgeladen wurde. GoTranscript bietet ebenfalls ein Treue-Programm an, wobei nach einer gewissen Anzahl transkribierter Minuten der Preis neuer Transkriptionen sinkt. Der Preis pro Minute setzt sich der Audio-Länge, den gewählten Zusatz-Leistungen und der Sprache zusammen. Als Beispiel würde ein klares englisches Gespräch von 60 Minuten mit zwei Sprechern und ohne Zusatz-Leistungen $1.15 / Minute kosten [14].

3.1.6 Sonix

Sonix ist ein, auf maschinelle Transkription fokussierter Dienstleister. ASR von Sonix erlaubt es Audio- und Videodateien zu transkribieren und danach in einem Online-Editor die notwendigen Korrekturen durchzuführen. Bei einer Dateilänge von 30 Minuten soll der Text bereits nach drei bis vier Minuten geliefert werden. Der Editor (vgl. Abbildung 1) synchronisiert die Audio Datei mit dem Text, sodass die gesprochenen Wörter im Text hervorgehoben werden, und der Benutzer über den Text beliebige Stellen abspielen kann.

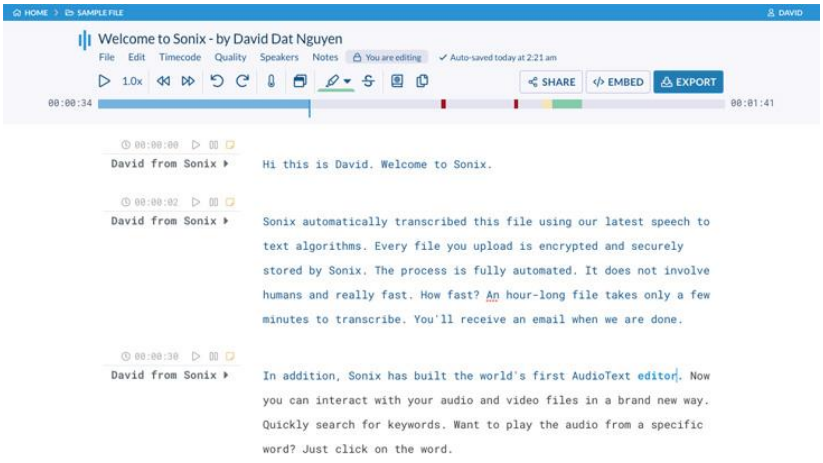


Abbildung 1: „AudioText Editor" von Sonix zur nachträglichen Korrektur des ASR Transkriptes [15]

Weitere Eigenschaften wie Suchen und Ersetzen, Wort-Zuversichten und Kollaborations-Möglichkeiten für mehrere Benutzer erleichtern laut Sonix das schnelle Korrigieren des Transkriptes [15].

Genauigkeit	Keine Angabe
Sprachen	26 Sprachen und Dialekte inklusive Englisch, Russisch, Chinesisch, Arabisch, Spanisch und Deutsch
Preise	$11.25 / Monat für die Nutzung des Editors und $6.00 / Stunde zu transkribierendes Audio (weitere Angebote für Firmen und Grosskunden)
Weitere Features	Diarization, «AudioText Editor», Export in verschiedene Formate

Tabelle 4: ASR Zusammenfassung - Sonix

3.1.7 Temi

Temi ist ein Dienstleister, der ASR für sämtliche Audio- und Videoformate anbietet. Bei guter Audio-Qualität, klaren Sprechern und geringen Akzenten verspricht der Anbieter eine Genauigkeit von 90 - 95%. Das Transkribieren eines halbstündigen Gespräches erfolgt innert einigen Minuten [16]. Der Editor (vgl. Abbildung 2) synchronisiert wieder den Text des Transkriptes mit der Audio-Datei (vgl. Abschnitt 3.1.6) und bietet nebst Textbearbeitung auch Suchen und Ersetzen, Ändern der Sprechernamen und Anpassen der Audio-Geschwindigkeit an [17].

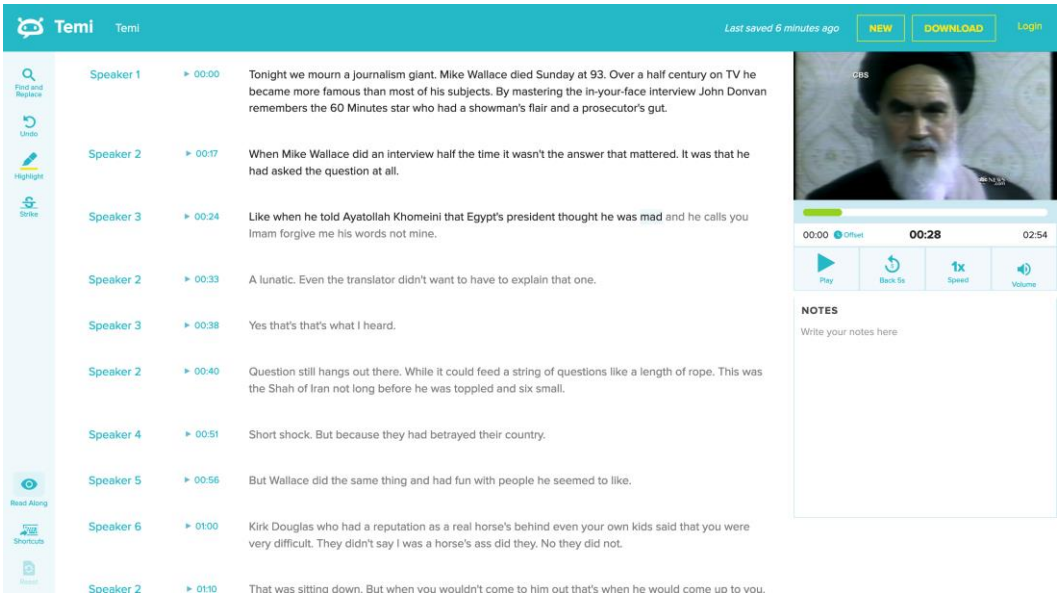


Abbildung 2: Editor von Temi zur nachträglichen Korrektur des ASR Transkriptes [17]

Genauigkeit	90 – 95%
Sprachen	Englisch
Preise	$0.10 / Minute
Weitere Features	Diarization, Online-Editor, Export als MS Word, PDF, SRT, VTT

Tabelle 5: ASR Zusammenfassung - Temi

3.1.8 Trint

Trint fokussiert sich ebenfalls auf automatische Transkription von Audio- und Videodateien. Wie die beiden zuvor genannten Dienste, stellt Trint ebenfalls einen Online-Editor (vgl. Abbildung 3) mit ähnlichen Features bereit (vgl. Abschnitte 3.1.6 und 3.1.7). Die Transkriptions-Dauer ist laut Trint etwa dieselbe wie die Länge der hochgeladenen Datei selbst [18].

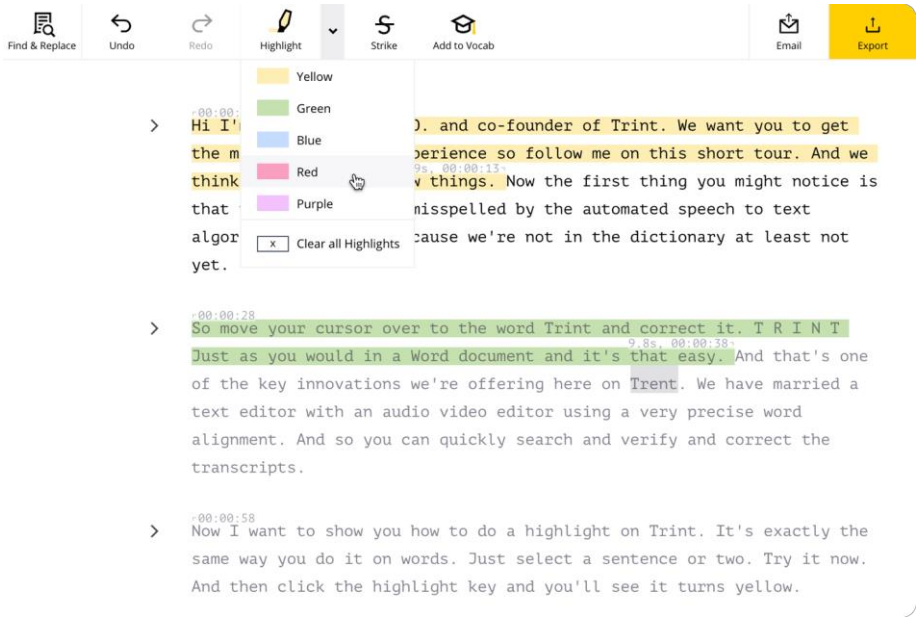


Abbildung 3: Editor von Trint zur nachträglichen Korrektur des ASR Transkriptes [18]

Genauigkeit	Keine Angabe
Sprachen	Englisch (US, UK, AUS), Französisch, Italienisch, Spanisch, Portugiesisch, Deutsch, Russisch, Polnisch, Finnisch, Ungarisch, Holländisch, Dänisch, Schwedisch
Preise	$0.25 / Minute
Weitere Features	Diarization, Online-Editor, Zapier-Integration für automatisierte Arbeitsabläufe, verschiedene Exportmöglichkeiten, iOS Applikation, um Aufnahmen direkt zu importieren

Tabelle 6: ASR Zusammenfassung - Trint

3.1.9 Fazit

Die Preise bewegen sich bei allen oben genannten ASR Dienstleister zwischen $0.1 und $0.25 pro Minute Audio. Für dieses Projekt interessant sind insbesondere die Anbieter Sonix, Temi und Trint. Alle drei bieten einen ähnlichen Editor an, um das maschinell erzeugte Transkript im Nachhinein zu korrigieren. Die Editoren von Sonix und Temi erlauben dem Benutzer tiefgreifende Änderungen an den Sprechern, dem Transkript und den Zeitstempeln, unterstützen aber nur die englische Sprache. Im Gegensatz dazu limitiert Trint die Editor-Funktionen auf Textänderungen, unterstützt dafür ein breites Angebot an Sprachen. Da sich die Zielgruppe dieser Arbeit auf den deutschsprachigen Raum begrenzt, wäre es interessant die Funktionalität von Sonix oder Temi mit deutschen ASR System zu kombinieren, und darauf aufzubauen.

3.2 Industriestandards von Transkript Formaten

Alle Anbieter von Transkriptions-Schnittstellen, wie Google, IBM, Microsoft, etc. haben unterschiedliche Transkript Formate für ihre Speech-to-Text und Diarization APIs. Dabei gibt es Unterschiede zwischen den Formaten, wenn ein Anbieter STT und Diarisierung anbietet oder wenn zusätzliche Funktionen, wie Zeitstempel oder Zuversicht (Confidence) eingestellt werden. Im Folgenden werden diese Formate, im Hinblick auf Design Entscheidungen für das Abspeichern der Transkripte, erläutert.

3.2.1 Google-Cloud-API

Die Google-Cloud-API bietet STT und Diarisierung in einer API mit unterschiedlichen Einstellungen an [19]. Das Format für einen Beispielstext mit aktivierter Diarisierung, Zeitstempel und Zuversichtlichkeit sieht folgendermassen aus:

```
{
    "results": [
        {
            "alternatives": [
                {
                    "transcript": "hi I'd like to buy a …",
                    "confidence": 0.87264615,
                    "words": [
                        {
                            "startTime": "0.500s",
                            "endTime": "1.100s",
                            "word": "hi",
                            "confidence": 0. 92142606,
                            "speakerTag": 2
                        }
                    ],
                    "languageCode": "en-us"
                }
            ]
        }
    ]
}
```

3.2.2 IBM-Cloud-API

Die IBM-Cloud-API bietet ebenfalls STT und Diarisierung an [20]. Dabei ist das Format ein wenig klarer aufgeteilt, welches der transkribierte Text und welches die gefundenen Sprecher sind, als bei Google:

```
{
    "results": [
        {
            "alternatives": [
                {
                    "timestamps": [
                        [
                            "hello",
                            0.68,
                            1.19
```

```
                        ]
                    ],
                    "confidence": 0.821,
                    "transcript": "hello yeah yeah how's Billy"
                }
            ],
            "final": true
        }
    ],
    "result_index": 0,
    "speaker_labels": [
        {
            "from": 0.68,
            "to": 1.19,
            "speaker": 2,
            "confidence": 0.418,
            "final": false
        }
    ]
}
```


3.2.3 Microsoft Azure

Microsoft Azure bietet sehr unterschiedliche APIs für verschiedenste Verwendungszwecke an [21]. Für das Transkribieren von Unterhaltungen konnten keine Formate gefunden werden, allerdings sieht das detaillierte Format für die Speech-to-Text API folgendermassen aus:


```
{
    "RecognitionStatus": "Success",
    "Offset": "1236645672289",
    "Duration": "1236645672289",
    "NBest": [
        {
            "Confidence": "0.87",
            "Lexical": "remind me to buy five pencils",
            "ITN": "remind me to buy 5 pencils",
            "MaskedITN": "remind me to buy 5 pencils",
            "Display": "Remind me to buy 5 pencils.",
        },
        {
            "Confidence": "0.54",
            "Lexical": "rewind me to buy five pencils",
            "ITN": "rewind me to buy 5 pencils",
            "MaskedITN": "rewind me to buy 5 pencils",
            "Display": "Rewind me to buy 5 pencils.",
        }
    ]
}
```


Auffallend hier ist, dass die API verschiedene Interpretationen zu den Phrasen angibt. Dies sind folgende Interpretationen:

- lexikalisch (transkribierter Text)
- invertierte Text Normalisierung (ITN; mit Zahlen, Telefonnummern, etc.)

- maskierte ITN (mit zusätzlichem Fluchwort Filter)
- anzuzeigender Text (mit zusätzlicher Zeichensetzung und Gross-/Kleinschreibung)

3.2.4 Temi API

Die Temi API ist ähnlich zu Google und IBM und unterstützt auch STT und Diarisierung, baut das Format allerdings logischer auf [22]. Es werden STT und Diarisierung unterstützt:

```
transcript {
    speakers: [
        speaker {
            id //speaker Id,
            name //the name of the speaker as 16abelled in the transcript
        }
    ],
    monologues: [
        monologue {
            id //monologue Id,
            speaker //id of the speaker of the monologue,
            speaker_name //name of the speaker
            elements: [
                element {
                    type //"text", "tag",
                    value //text value of the element
                    ts //the timestamp of the beginning of the element
                         relative to the beginning of the audio
                    end_ts // the timestamp of the end of the element
                         relative to the beginning of the audio
                }
            ]
        }
    ]
}
```

Besonderheit hier ist, dass eine Sprecheräusserung (Utterance) in Monologe aufgeilt wird und jedes Element davon noch einen Typ besitzt, was das Element ist. Dies kann ein Wort, eine Pause, einen Ausruf, etc. sein.

3.2.5 SeerNet – Multi-Speaker-Diarization API

Die Multi-Speaker-Diarization API von SeerNet ist, wie der Name schon sagt, nur für Diarisierung zu verwenden [23] und baut auf der DeepAffects Speaker Diarization API auf [24]. Das Format teilt ein Audio Inhalt in Segmente auf:

```
{
    "num_speakers": 2,
    "segments": [
        {
            "speaker_id": "speaker1",
            "start": 0,
            "end": 1
        }
    ]
}
```

3.2.6 remeeting ASR API

Die remeeting ASR API wurde so entworfen, dass sie mit der IBM-Cloud-API kompatibel ist [25]. Daher ist das Transkript Format auch sehr ähnlich zu IBM. Es sieht folgendermassen aus:

```
{
    "created": "1997-08-29T03:14:15.926535-04:00",
    "id": "42",
    "updated": "2011-04-19T20:11:04.192011-04:00",
    "results": [
        {
            "result_index": 0,
            "results": [
                {
                    "final": true,
                    "alternatives": [
                        {
                            "transcript": "Hello, World. ",
                            "transcript_raw": "hello world ",
                            "confidence": 0.902
                        }
                    ]
                }
            ]
        }
    ],
    "status": "completed"
}
```

Zusätzliche Funktionen, wie Zeitstempel oder Diarisierung können separat aktiviert werden.

3.2.7 Fazit

Wie man den Format Beispielen oben entnehmen kann, sind die meisten APIs, welche STT und Diarisierung unterstützen, sehr ähnlich aufgebaut. Grössere Unterschiede gibt es nur bei der Temi API, das Format ist allerdings lesbarer. Wenn eine API nur STT oder Diarisierung unterstützt, sind die Formate einfacher, da sie weniger Informationen speichern müssen. Ein Beispiel dafür ist die Multi-Speaker-Diarization API in Abschnitt 3.2.5.

Im Kapitel 4.3 wird detailliert auf die Format Design Entscheidungen für die «InterScriber» Anwendung eingegangen und erklärt, wieso eine Mischung aus den APIs verwendet wurde.

4 Vorgehen

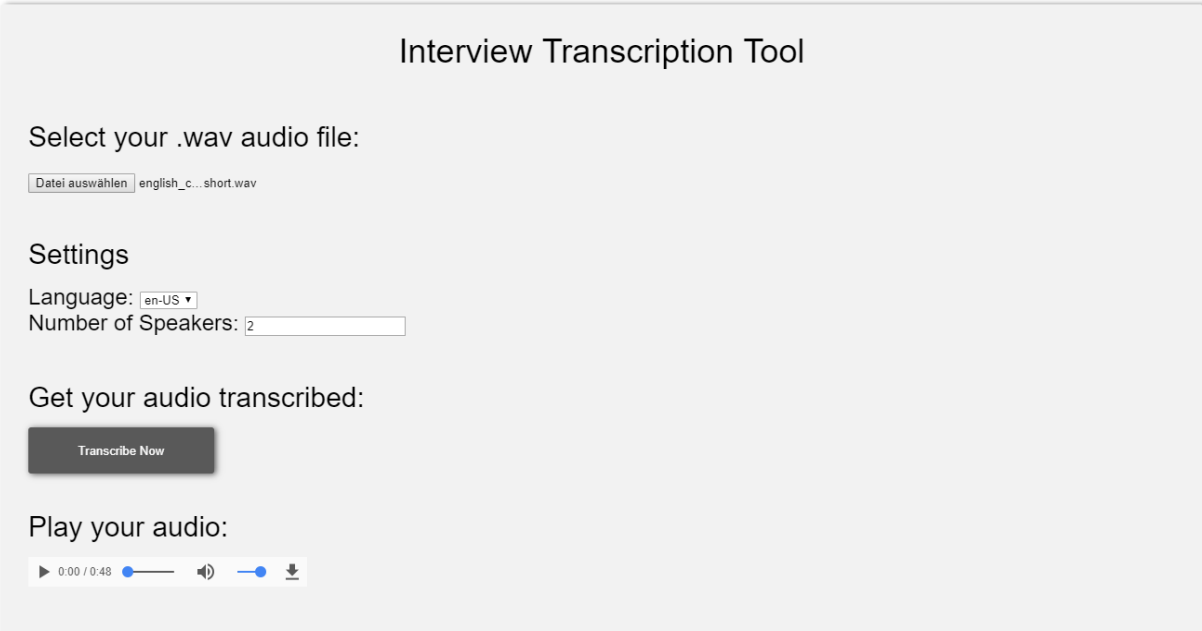
Die Software «InterScriber» wurde mittels einem agilen Vorgehensmodell, unter der Verwendung von Elementen aus Scrum und Kanban, in einwöchigen Sprints entwickelt. Jede Woche wurden die im vorgehenden Sprint erarbeiteten Resultate präsentiert, wonach die verbleibenden Funktionen diskutiert und neu priorisiert wurden. Anhand dessen wurde bestimmt, welche User Stories als nächstes in Angriff zu nehmen waren.

Im Folgenden wird zuerst das bestehende Produkt aus der vorangegangenen Arbeit erklärt, anschliessend werden Anforderungen und Funktionen an die Software definiert, danach wird das Design mit physischer- und logischer Architektur erläutert, bevor die Implementation gezeigt wird. Zum Abschluss dieses Kapitels werden noch Probleme und Herausforderungen während der Entwicklung aufgezählt und erklärt, wie damit umgegangen wurde.

4.1 Bestehendes Produkt

Wie erwähnt baut diese Arbeit auf den Erkenntnissen und dem Quellcode eines vorgegangenen Projekts auf. In diesem Abschnitt werden die Elemente dieser Vorarbeit beschrieben.

Der während der 14-wöchigen Projektarbeit erarbeitete Prototyp erlaubt die Transkription von englischen Konversationen mittels der Google Cloud API über ein einfach gehaltenes User Interface. Der Benutzer kann Audio-Dateien hochladen, welche transkribiert und ihm in textlicher Form dargestellt werden (vgl. Abbildung 4). Die Aussagen der einzelnen Sprecher werden klar getrennt angezeigt und Pausen im Sprachfluss werden ebenfalls dargestellt. Die Audio-Quelle kann mittels eines Players durch den Benutzer abgespielt werden und die dabei wiedergegebenen Wörter werden farblich hervorgehoben. Zusätzliche Funktionen sind das Umbenennen der vom System erkannten Sprecher, sowie das Speichern und Öffnen eines Transkriptes im Programm.



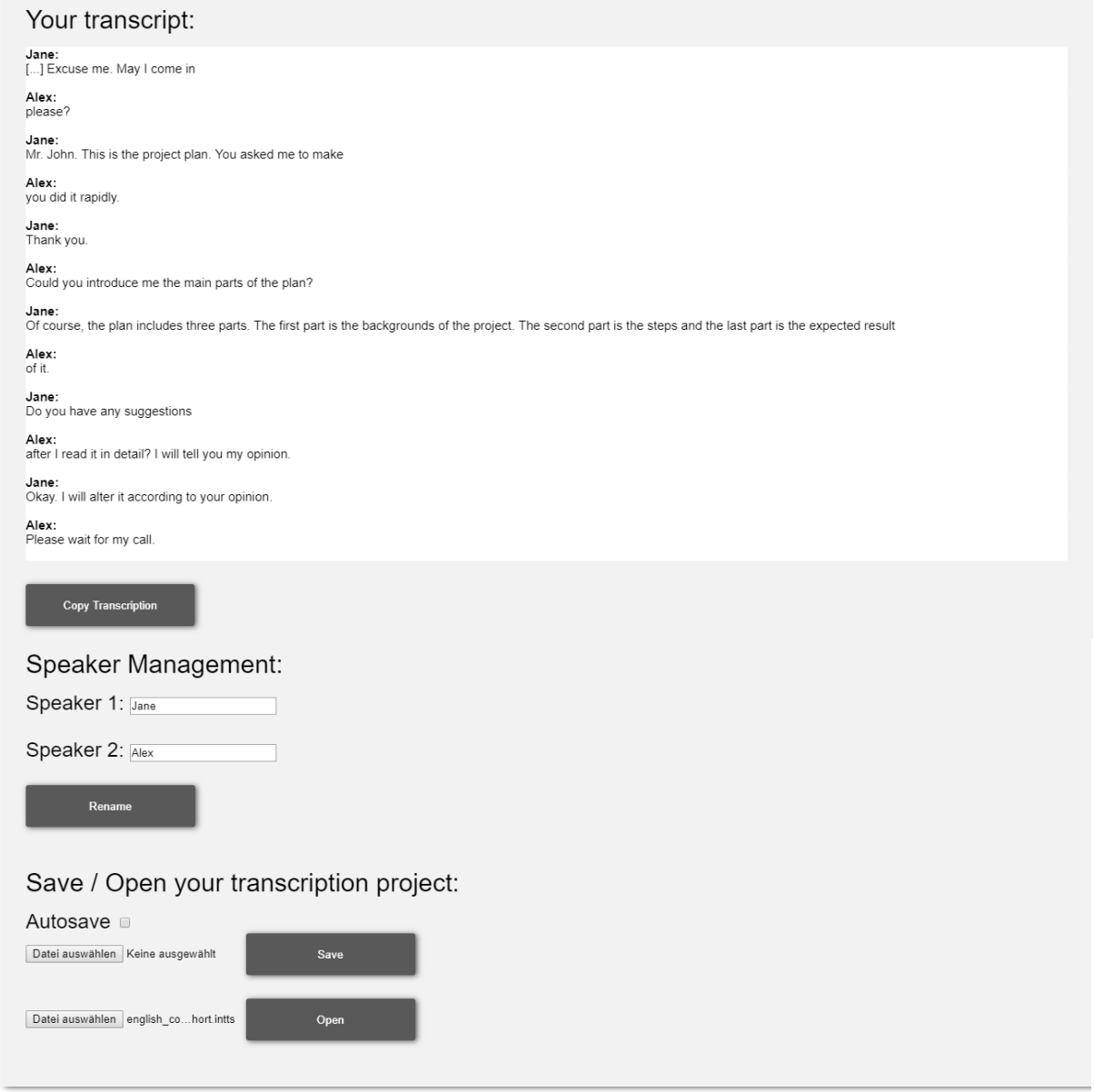


Abbildung 4: Benutzeroberfläche des bestehenden „Interview Transcription Tool“ [1]

Die Architektur des Programmes wurde so gewählt, dass möglichst einfach neue Systeme oder Dienste für STT und Diarisierung ohne grossen Programmieraufwand eingebaut werden können. Es soll ebenfalls die Möglichkeit bestehen, die Anwendung komplett offline zu verwenden, falls entsprechenden Systeme vorhanden sind. Um diese Ziele zu erreichen, werden alle Teile dieser Lösung, abgesehen von externen ASR Diensten (z.B. Google Cloud), direkt auf dem Client ausgeführt.

4.2 Anforderungsanalyse

Das Kapitel Anforderungen teilt sich auf in funktionale und nicht-funktionale Anforderungen. Zuerst werden die zentralen fünfzehn User Stories, wie sie zu Beginn der Arbeit festgelegt wurden, erklärt. Darauf werden die nicht-funktionalen Anforderungen detailliert.

4.2.1 Funktionale Anforderungen

Die ersten drei User Stories (US1-US3) beziehen sich direkt auf das Korrigieren von falsch transkribierten Teilen. Der Benutzer muss Fehler im Text, wie auch in Bezug auf die Sprecher berichtigen können.

US1	Fehler im Transkript korrigieren
Als Benutzer möchte ich Stellen im Text, die vom System falsch erkannt wurden, manuell im Editor korrigieren können.	

US2	Fehler der Sprechererkennung korrigieren
Als Benutzer möchte ich Stellen im Transkript, die vom System dem falschen Sprecher zugeordnet wurden, manuell dem richtigen Sprecher zuweisen können.	

US3	Parallele Sprecher markieren
Da das System von «InterScriber» Probleme hat, gleichzeitige Sprecher zu transkribieren, möchte ich als Benutzer im Editor zwei Aussagen als «parallel» markieren können. Diese Aussagen müssen wie alle Anderen editierbar sein.	

US4 bis und mit US10 beschreiben Funktionen, die dem Benutzer die Korrektur erleichtern sollen, während die letzten drei spezifisch Anforderungen an die Audio-Player stellen.

US4	Problemstellen hervorheben
Um im Transkript potenzielle Inkorrektheiten zu finden, möchte ich als Benutzer alle Stellen mit tiefer Zuversicht anzeigen lassen können. Ich will ebenfalls die Granularität einstellen können, ab welcher Zuversicht (Prozentzahl) etwas hervorgehoben werden soll.	

US5	Pausen anzeigen
Als Benutzer möchte ich nicht, dass mir jede noch so kleine Pause, die im Gespräch erkannt wurde, im Transkript angezeigt wird. Entsprechend will ich die Pausenanzeige aktivieren und deaktivieren können und ich möchte wählen, ab welcher Pausenlänge sie angezeigt werden sollen.	

US6	Transkript Export
Nach Fertigstellen der Korrekturen im Editor möchte ich als Benutzer mein Transkript als MS Word Dokument, als MS Excel Liste und als Fliesstext in die Zwischenablage, exportieren können.	

US7	Suchen und Ersetzen
Als Benutzer möchte ich nach spezifischen Wörtern im Transkript suchen und mir die Resultate anzeigen lassen. Nachdem ich nach einem Term gesucht habe, möchte ich als Benutzer die Vorkommen durch ein anderes Wort ersetzen können. Ich will dabei die Wahl haben, eines nach dem anderen oder alle auf einmal zu ersetzen.	

US8	Wiederholendes Abspielen
Als Benutzer möchte ich einen Teil im Transkript markieren und wiederholend so lange abspielen, bis ich den Audio-Player pausiere, oder etwas anderes abspiele.	

US9	Segment abspielen
Als Benutzer möchte ich eine Aussage eines Sprechers durch klicken auf den Sprechernamen an dieser Stelle abspielen.	

US10	Wiedergabe-Geschwindigkeit anpassen
Um spezifische Stellen genau korrigieren zu können, möchte ich die Geschwindigkeit des Audio-Players einstellen können.	

Die letzten fünf User Stories (US12 – US15) beziehen auf allgemeine Aspekte der Applikation.

US11	Electron Applikations-Menü
Als Benutzer möchte ich über das Applikations-Menü auf die wichtigsten Operationen direkt zugreifen können: Interview öffnen, Interview speichern, Interview speichern unter, neue Konversation transkribieren und Sprache wechseln.	

US12	Sprache wechseln (GUI)
Als Benutzer möchte ich in der Applikation zwischen deutscher und englischer Übersetzung wählen und jederzeit die Sprache wechseln können.	

US13	Informationen zur Audio-Quelle
Als Benutzer möchte ich nach Hochladen einer neuen Audio-Datei, dass mir relevante Informationen dazu angezeigt werden (Länge, Grösse, Transkriptionskosten, ungefähre Transkriptionsdauer, etc.).	

US14	Deutsche Interviews transkribieren
Als Benutzer möchte ich nebst englischen Interviews, auch deutsche Konversationen transkribieren lassen können.	

US15	IBM-Cloud-API unterstützen
Als Herausgeber der Applikation, möchte ich nebst Google Cloud auch den Dienst von IBM integrieren, um die Infrastruktur und Modularität von «InterScriber» zu testen und um dem Benutzer die Wahl zwischen den beiden zu bieten.	

4.2.2 Nicht-funktionale Anforderungen

Einer der zentralsten, nicht-funktionalen Anforderungen ist ein modernes, selbst-erklärendes User Interface, das auf die wichtigsten Elemente für ein schnelles Transkribieren begrenzt ist. Der Prozess vom initialen Hochladen der Audio-Datei bis zum Editor und schlussendlich zum Export soll so aufgebaut sein, dass sich ein Benutzer ohne Anleitung durcharbeiten kann. Dieser Ansatz soll auch für die Installation der Software gelten; so soll ein Anwender keine komplizierten Installationen oder Einstellungen vornehmen müssen.

Abgesehen von der Nutzererfahrung soll die Architektur der Applikation weiterhin so konzipiert werden, dass die Erweiterung auf weitere ASR Dienste und Sprachen möglichst wenig Aufwand bedingt.

4.3 Design

In den nachfolgenden Unterkapiteln wird zuerst die physische Architektur mit den eingesetzten Technologien, welche schon im bestehenden Produkt verwendet wurden, erklärt. Anschliessend wird ausführlich auf die logische Architektur der Anwendung eingegangen.

4.3.1 Physische Architektur

Die Software ist in zwei separierte Anwendungen, dem Backend und dem Frontend unterteilt. Das Backend beinhaltet alle Aufgaben für die Transkription, während das Frontend vor allem für die Darstellung und Bearbeitung des Transkriptes zuständig ist.

Für das Backend wurde eine Python Applikation verwendet, welche mit Hilfe des Frameworks Flask mit dem Frontend kommuniziert. Sämtliche Transkriptions-Dienste und APIs sind über eine Schnittstelle an dieser Python Applikation angehängt. In dieser Arbeit sind es die Google-Cloud- und IBM-Cloud-API. Im nachfolgenden Unterkapitel wird detailliert auf diese Schnittstellen eingegangen.
Das Frontend ist eine Electron Anwendung, ein Framework, dass das Erstellen von plattformunabhängigen Desktop-Applikationen mittels Webtechnologien ermöglicht. Für die Gestaltung der Single-Page Applikation wurde das JavaScript-Framework Vue.js verwendet.

Die physische Architektur wurde so gewählt, dass der Benutzer die Möglichkeit hat, die komplette Anwendung offline zu betreiben, falls entsprechende offline APIs vorhanden sind. Es ist aber auch möglich, das Backend auf einem Server zu hosten und von einem lokalen Computer durch das Electron Frontend auf dieses zuzugreifen.
In Abbildung 5 ist die Situation dargestellt, in der das Backend auf einem Server deployed wurde und das Frontend auf dem Anwender PC läuft. Wenn offline APIs vorhanden sind, werden diese direkt über die Interface Schnittstelle angesprochen, laufen aber auf demselben Server, ansonsten wird über HTTP mit der API kommuniziert.

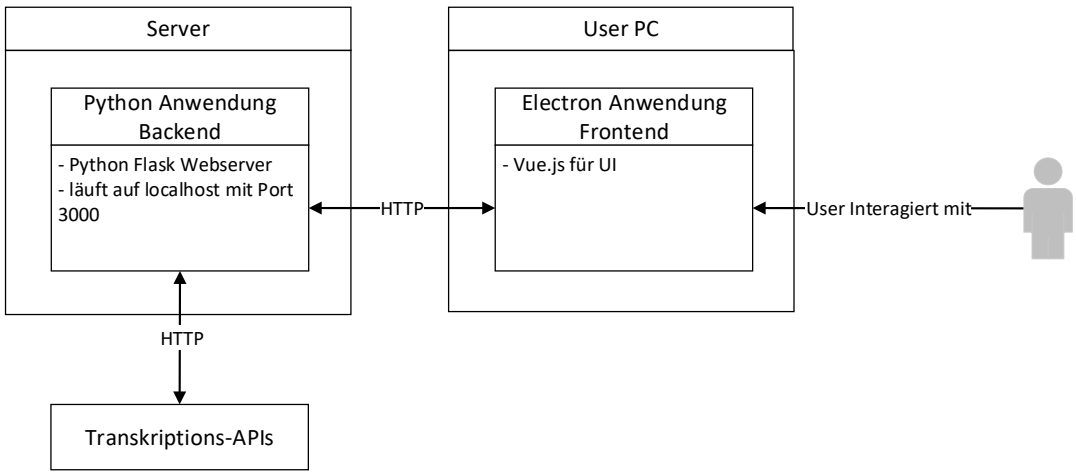


Abbildung 5: InterScriber - Physische Architektur

4.3.2 Logische Architektur

Backend

Wie weiter oben erklärt, wurde für das Backend System eine Python Anwendung verwendet. Python eignet sich optimal für Machine Learning, Data Science und Natural Language Processing (NLP) Probleme. Da die Möglichkeit bestehen soll, dass lokale Transkriptionssysteme angehängt werden können, ist die Programmiersprache eine gute Wahl, da die meisten der bestehenden Lösungen in diesem Bereich in Python geschrieben sind.

Nachfolgend wird das Klassendiagramm beschrieben. Der Lesbarkeit halber wurde es in drei Teile unterteilt. In Abbildung 6 werden die «app» und Service Klassen mit den Schnittstellen angezeigt, in Abbildung 7 die APIs mit Schnittstellen und Implementationen und in Abbildung 8 wird die ganze Interview Klasse, inklusive Subklassen dargestellt. Das komplette Klassendiagramm ist im Anhang zu finden (vgl. Abschnitt 8.3.4).

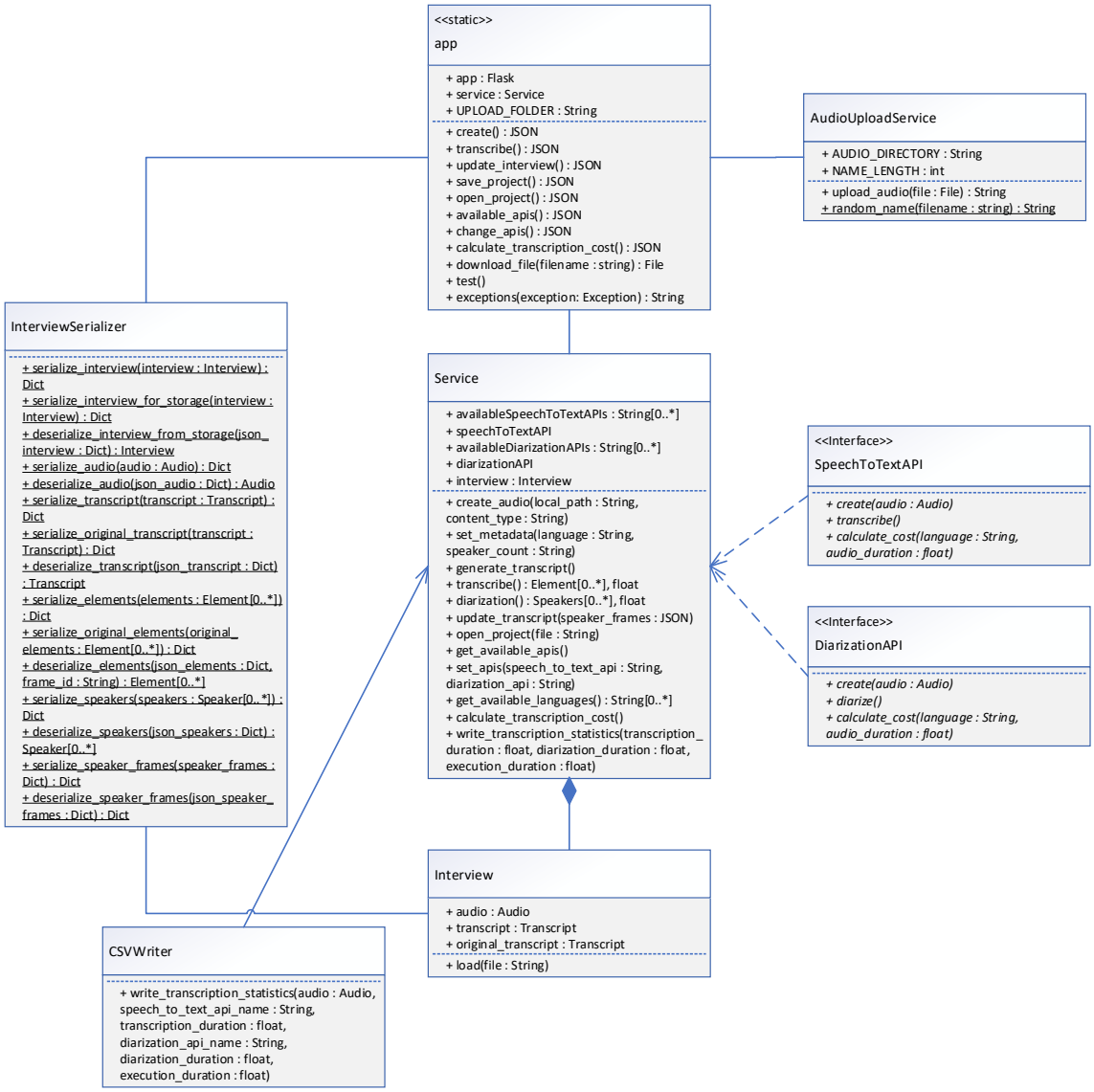


Abbildung 6: Backend-Klassendiagramm – App und Service Klassen

Die zentrale Schnittstelle zum Frontend ist die statische Klasse «app». Diese beinhaltet die Flask Anwendung, welche über eine REST (Representational State Transfer) Schnittstelle mit dem Frontend kommuniziert. Dabei werden JSON-Objekte über das HTTP Protokoll gesendet und empfangen. Ebenfalls Teil davon sind alle Methoden, welche für die Transkription und Bearbeitung eines Transkripts nötig sind. Eine Auflistung aller HTTP-Schnittstellen ist im Anhang (vgl. Abschnitt 8.3.5) zu finden. Die «app» wird als Webserver gestartet.

Die «app» Klasse leitet alle Anfragen an die zuständigen Service Klassen weiter. Dies ist bei der Erstellung der Audio-Datei der «AudioUploadService», welcher dafür zuständig ist, die Audio-Datei umzubenennen, damit bestehende Dateien nicht zufällig überschrieben werden, und in den Upload-Ordner zu kopieren. Dieser Vorgang ist vor allem wichtig, wenn das Backend auf einem Server läuft und somit keinen Zugriff auf lokale Dateien hat.

Die Klasse «Service» ist die Hauptklasse der Anwendung, wird über «app» angesprochen und sorgt dafür, dass alle benötigten Aufgaben ausgeführt werden. Dies beinhaltet das Analysieren von Audio-Dateien, Berechnen von Transkriptions-Kosten, Transkribieren und Diarisieren mittels der implementierten Schnittstellen-Klassen, Generieren und Aktualisieren des Transkriptes bei Benutzer Änderungen, Schreiben der Transkriptions-Statistiken mittels «CSVWriter» und Speichern und Laden der Projekte (vgl. Abbildung 6). «Service» beinhaltet unter anderem auch alle verfügbaren Speech-to-Text- und Diarization-APIs, welche dem Frontend geschickt werden. Ausserdem werden in den Variablen «speechToTextAPI» und «diarizationAPI» jeweils ein Objekt der implementierten Schnittstellen gespeichert. Diese werden weiter unten noch genauer beschrieben. Die «Service» Klasse beinhaltet ausserdem das Interview-Objekt, welches später erläutert wird.

Damit ein Interview an das Frontend geschickt und zurückerhalten werden kann, wird die «InterviewSerializer» Klasse benötigt. Diese kann Interviews serialisieren, damit diese über «app» als JSON-Objekt an das Frontend geschickt werden können und deserialisieren, damit ein Interview Objekt vom Frontend wieder in die Interview Struktur des Backend gebracht werden kann.

In Abbildung 7 wird die Aufteilung der Speech-to-Text- und Diarization APIs genauer dargestellt. «SpeechToTextAPI» und «DiarizationAPI» bilden die Schnittstelle zu der «Service» Klasse. Jede Speech-to-Text API muss diese Schnittstelle implementieren, so auch bei den Diarization APIs. Wie bereits erwähnt, sind dies die «GoogleCloudAPI» und «IBMCloudAPI» in dieser Arbeit. Weil diese Dienste STT, sowie Diarisierung unterstützen, müssen sie beide Schnittstellen implementieren. Zusätzlich braucht jede der beiden Klassen noch einen Konverter, um die Elemente und Sprecher der Transkription zu extrahieren. Dafür gibt es auch für STT und Diarisierung eine eigene Schnittstelle, welche die betreffenden Klassen implementieren müssen. Beide Dienste speichern auch die Audio-Datei auf ihren eigenen Cloud-Lösungen ab, um während der Transkription schnell darauf zugreifen zu können.

Jeder der Klassen liest bei der Instanziierung die Konfigurations-Datei, welche angibt, welche Funktionen eingeschaltet sind, welche Sprachen unterstützt sind oder wie gross das Transkriptions-Timeout ist. Dafür wird die «ConfigParser» Klasse verwendet. Diese kommt auch in der «Service» Klasse zum Einsatz, um die verfügbaren Transkriptions-APIs zu lesen.

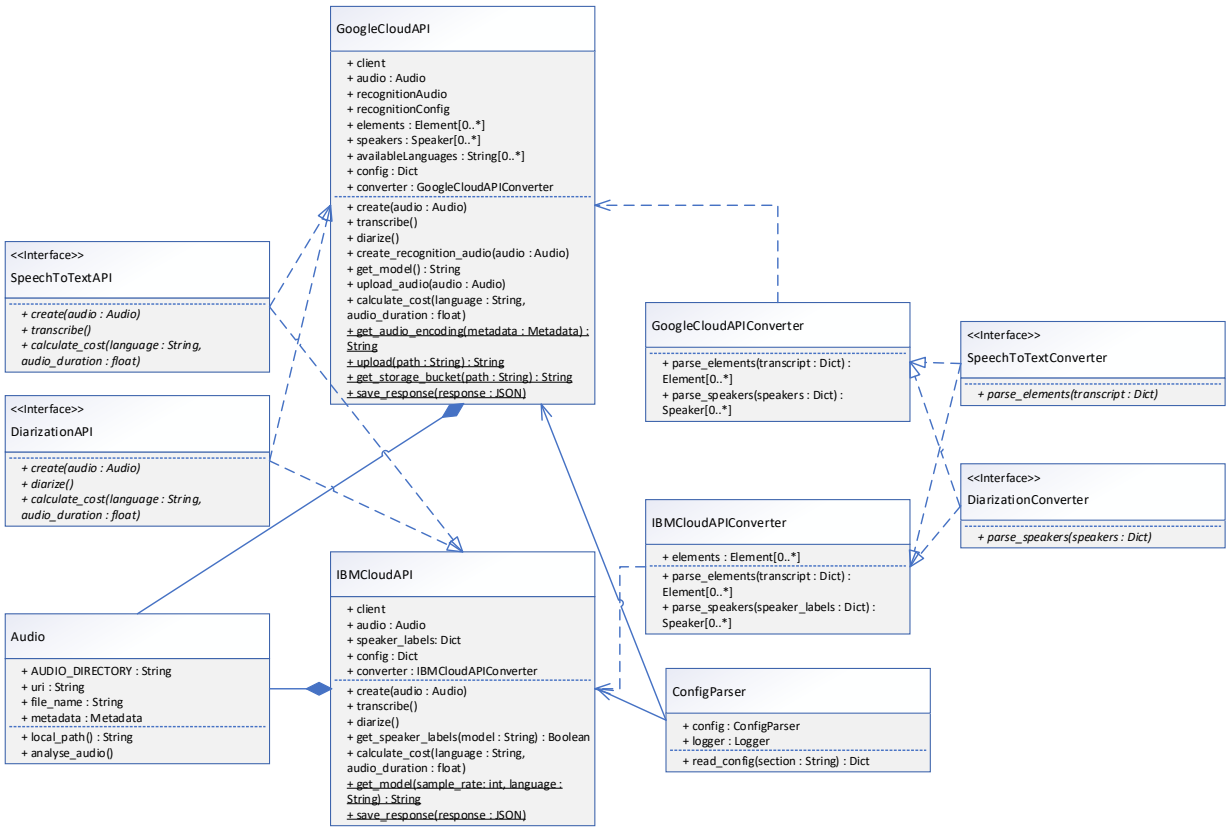


Abbildung 7: Backend-Klassendiagramm – SpeechToText und Diarization Klassen

Die «Interview» Klasse setzt sich aus Folgenden Unterklassen zusammen (vgl. Abbildung 8): Jedes Interview beinhaltet das aktuelle Audio, das Original Transkript und das aktuelle Transkript, damit Benutzer-Änderungen festgehalten werden können. Ausserdem beinhaltet diese Klasse auch das Laden eines Interview Projektes.

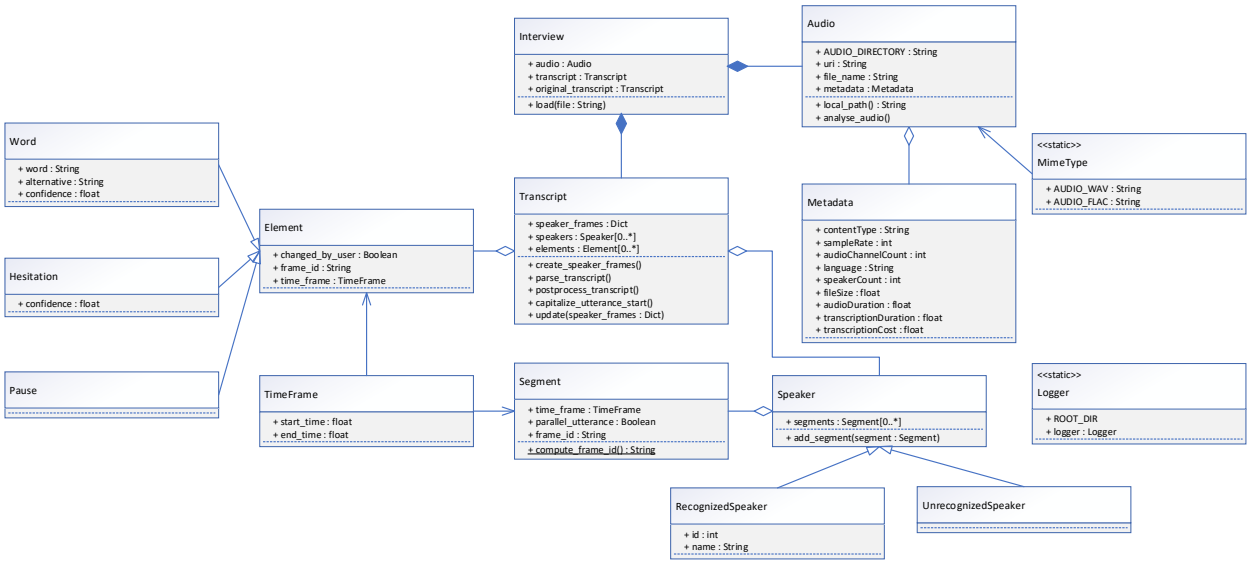


Abbildung 8: Backend-Klassendiagramm – Interview- und Unterklassen

Für das Speichern wurde ein Format gewählt, welches eine Mischung aus den grossen API-Anbieter, wie Google und IBM, und Temi ist (vgl. Abschnitt 3.2.7). Grund dafür waren die Entscheidungen, so nahe wie möglich am Stand der Technik zu sein und dass das Format lesbar sein sollte. Da das «Audio» Objekt auch abgespeichert werden muss, bot sich die Struktur der «Interview» Klasse an. So wird das «Interview» Objekt, inklusive «Audio», «transcript» und «original_transcript» serialisiert und an das Frontend schickt, welches das Interview in einer gewählten Datei abspeichert. Dies muss über das Frontend geschehen, da es wiederum möglich ist, dass sich das Backend auf einem Server befindet und keinen Zugriff auf das lokale Datei-System hat. Wie das Format genau aussieht, kann dem Anhang (vgl. Abschnitt 8.3.5) unter '/interview', entnommen werden.

Die «Audio» Klasse beinhaltet eine URI, wenn die Audio-Datei für die Transkription auf einen Server hochgeladen wird. Ausserdem einen Datei-Namen, welcher bei der Kopierung in den Upload Ordner generiert wird. Bei der Analyse der Audio-Datei wird das Metadaten-Objekt mit Audio-Dauer, Sampling Rate, Datei-Grösse, Anzahl Audio Kanäle und Audio-Datei Format gefüllt. Die unterstützen Audio-Formate werden in der statischen «MimeType» Klasse aufgeführt. In dieser Arbeit sind dies das WAV und FLAC Format, andere Formate werden von der Google-Cloud-API nur mit viel Aufwand unterstützt.

Ein «Transkript» Objekt setzt sich aus den Elementen und Sprechern, welche bei der Transkription in dieses Format gebracht wurden, und den Sprecher-Rahmen zusammen. Die Sprecher-Rahmen werden nach der Transkription anhand der Sprecher erzeugt und mit Elementen gefüllt. Anschliessend hat man das eigentliche Transkript mit Sprecher Äusserungen und den dazugehörenden Elementen, wie Worten, Pausen oder Zögern (Hesitation).
Jedes dieser Elemente erbt von der «Element» Klasse, welche gemeinsame Datenfelder, wie ein «changed_by_user» Kennzeichen, die «frame_id» und dem «time_frame», welcher sich aus Start- und Endzeit zusammensetzt, enthält. Bei einem «Wort» Objekt kommen die Datenfelder «word», «alternative» und «confidence» hinzu, während es bei einem «Zögern» Objekt lediglich die «confidence» ist und bei einer Pause keine zusätzlichen Datenfelder benötigt werden.
Ein «Sprecher» Objekt, von welchem die «RecognizedSpeaker» und «UnrecognizedSpeaker» erben, beinhaltet «Segment» Objekte. Diese beinhalten einen «time_frame», ein «parallel_utterance» Kennzeichen und eine «frame_id». Diese wird bei Erstellung des Objektes erzeugt und alle dazugehörigen Elemente müssen dieselbe «frame_id» besitzen. Falls der Sprecher bereits bei der Transkription bekannt ist, wird er zu einem «RecognizedSpeaker» und erhält die Datenfelder «id» und «name».

Die statische Klasse «Logger», welche rechts neben der «Speaker» Klasse gezeichnet ist, wird in fast allen Klassen des Backend verwendet, um Ereignisse zu protokollieren. Daher wurde diese Klasse der Lesbarkeit halber so dargestellt, anstatt sie mit jeder anderen Klasse in Beziehung zu bringen.

Zusätzlich kann gesagt werden, dass sich die Backend-Struktur während der Entwicklung immer wieder ein wenig angepasst hat, um neuen Bedürfnissen, wie parallele Sprecheräusserungen oder das Abspeichern der Audio-Datei in einem bestimmten Ordner bei der Erstellung des «Audio» Objektes, gerecht zu werden.

Frontend

Wie bereits bei der physischen Architektur beschrieben, wurde für das Frontend eine Electron Anwendung mit Vue.js gewählt. Ein Vorteil von Electron ist, dass die Anwendung auf jedem Betriebssystem laufen kann und bei Bedarf auch offline einsetzbar ist. Der zweite Vorteil ist, dass Electron Funktionen anbietet, welche das Nutzererfahrung stark verbessern können. So kann zum Beispiel ein eigenes Anwendungsmenü eingesetzt oder spezielle Dialoge für die Anwenderinteraktion angezeigt werden.

Vue.js ist ein neues Framework, wurde erst 2014 ins Leben gerufen und wird stetig weiter-entwickelt. Es ist komponentenbasiert und erlaubt, Webseiten als Single-Page-Applikationen zu bauen. Ein Vorteil ist, dass jede Komponente sowohl HTML in Form eines «template», als auch JavaScript als «script» und CSS als «style» enthält. Im «script» Teil werden benötigte Komponenten importiert und Variablen und Methoden definiert. Somit kann jede Komponente klar von den anderen abgegrenzt und individuell gestaltet werden.

Jede Vue.js Anwendung hat eine zentrale Instanz, welche Abhängigkeiten, wie das i18n Plugin für Übersetzungen, definiert. Für das Frontend wurde kein Klassendiagramm angefertigt, da es wegen den Komponenten keine Klassen gibt. Daher werden nachfolgend alle verwendeten Komponenten aufgezählt und kurz erklärt, was ihre Aufgabe ist.

LandingPage	Zentrale Komponente, beinhaltet alle anderen und sorgt für den Transkriptions-Ablauf
Header	Ist für die Darstellung der Kopfzeile verantwortlich
Alert	Zeigt Fehler (z.B.: beim Laden von Audio-Dateien) an
LoadAudio	Beinhaltet einen File-Input für das Auswählen von Audio-Dateien und Erstellen im Backend
AudioStatistics	Zeigt Audio-Statistiken nach dem Erstellen der Audio-Datei an
GeneralSettings	Für die Auswahl der STT und Diarization APIs, nur für Entwicklung gedacht, ist dafür zuständig, dass die aktuellen APIs und Sprachen im Frontend verfügbar sind.
AudioSettings	Für das Einstellen der Sprache und Anzahl Benutzer und für das Berechnen des Transkriptions-Preises
LoadInterview	Ist für das Laden eines bestehenden Interviews (Transkript + Audio) verantwortlich
ProgressBar	Zeigt einen Fortschrittsbalken während des Transkribierens
SaveInterview	Speichert geöffnete Interviews. Falls es noch nicht gespeichert wurde, wird der Anwender gefragt, wo im Datei-System das Interview mit Datei-Endung «.intts» gespeichert werden soll. Anschliessend wird das Interview vom Backend geholt und auf dem lokalen Computer abgespeichert. Der Benutzer hat auch die Möglichkeit ein geöffnetes Interview unter einem neuen Namen neu abzuspeichern.
Editor	Ist für das Bearbeiten des Transkripts zuständig
EditorSettings	Einstellungen für den Editor, wie Wiedergabegeschwindigkeit der Audio-Datei, Änderung und Darstellung der Zuverssichts-Schwelle (confidence threshold), Änderung und Darstellung der Pausen und Export

	Möglichkeiten, wie Kopieren in die Zwischenablage und Konvertieren zu Word oder CSV
AssignSpeaker	Dialog für das Zuweisen eines Sprechers einer Äusserung
NewUtterance	Dialog für das Erstellen einer neuen Äusserung
DeleteUtterance	Dialog für das Löschen einer Äusserung
ChangeLanguage	Für das Wechseln der Anzeigesprache verantwortlich (In dieser Arbeit wird Englisch und Deutsch unterstützt)
Snackbar	Verantwortlich für das Anzeigen von Toasts am unteren Bildschirmrand bei Erfolgen (z.B.: Interview gespeichert) und Fehlern (z.B.: Input darf nicht leer sein)

Tabelle 7: Frontend-Komponenten von InterScriber

Zusätzlich zu den Komponenten gibt es noch sogenannte «Mixins». Diese beinhalten Methoden, welche von mehreren Komponenten verwendet werden und zur Vereinfachung extrahiert wurden. Ein Beispiel hierfür ist die Umrechnung von Zeiteinheiten.

Für die Darstellung von Pausen und Zögern im Transkript wurden diese an die allgemeinen Transkript-Konventionen angelehnt [27].
Eine Pause wird folgendermassen gekennzeichnet: (1.2 s) – Eine Pause für 1.2 Sekunden. Ein Zögern wird als (hesitation) dargestellt, wird jedoch nur vom Dienst von IBM, nicht aber von Google Cloud, erkannt.

4.4 Implementation

Die Software «InterScriber» wurde auf Basis des bestehenden Produkts während 14 Wochen weiterentwickelt und verbessert. Dabei arbeiteten alle Teammitglieder sowohl am Front-, als auch am Backend. Es wurden alle benötigten Grundfunktionen implementiert, das Userinterface stark vereinfacht und der Ablauf einer Transkription klarer aufgeteilt. Die User Stories 1 – 6 und 8 – 15, inklusive der Nicht-funktionalen Anforderungen wurden umgesetzt. Für die User Story 7 (Suchen und Ersetzen) blieb am Schluss zu wenig Zeit, da noch Probleme evaluiert werden mussten.
Der genaue Ablauf der Implementierung, was Woche für Woche erledigt wurde, kann dem Projektplan im Anhang entnommen werden (vgl. Abschnitt 8.2.1). Mögliche neue Funktionen und bekannte, noch nicht behobene Fehler können im Product Backlog in Anhang (vgl. Abschnitt 8.2.2) nachgeschlagen werden.

Nachfolgend wird ein typischer Arbeitsablauf in «InterScriber» dargestellt und ausführlich erklärt. Das Öffnen der Anwendung zeigt immer die Startseite an (vgl. Abbildung 9).
Im ersten Schritt kann entweder eine Audio-Datei, welche transkribiert werden soll, gewählt werden oder man öffnet ein bereits transkribiertes Interview, welches zuvor als «.intts» Datei abgespeichert wurde. Dabei wird ein Fenster geöffnet, wo die Dateien ausgewählt werden können.

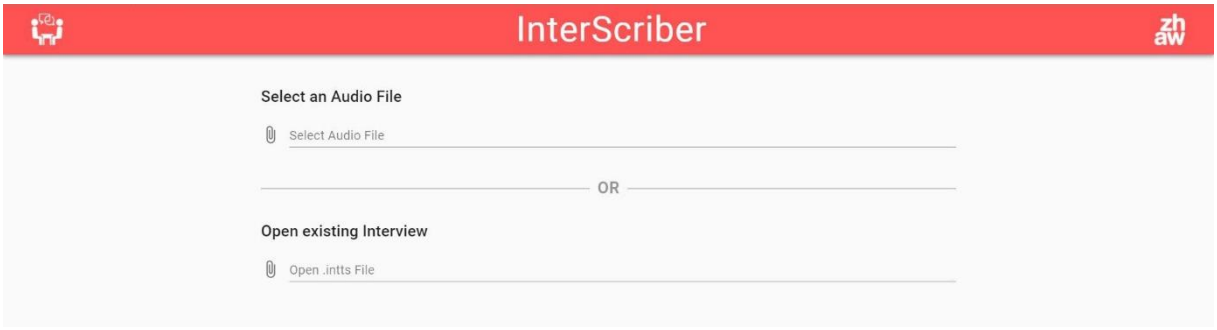


Abbildung 9: InterScriber - Startseite

Wird eine neue Audio-Datei ausgewählt, wird diese zuerst im Backend in den spezifizierten Upload Ordner kopiert, anschliessend wird die Audio-Datei auf die Metadaten analysiert, die standardmässig ausgewählten STT und Diarization APIs erstellt (welche Funktionen sind aktiviert, welche Sprachen unterstützt, etc.), bevor das Resultat wieder zurück an das Frontend gegeben wird. Anschliessend können die Statistiken (Metadaten und zusätzliche Felder, wie Transkriptions-Preis, etc.) bei Bedarf eingeblendet werden und es können die Audio Einstellungen, wie Interview Sprache oder Anzahl Sprecher gewählt werden (vgl. Abbildung 10).
Wird bei der Interview Sprache eine Auswahl geändert, wird im Backend der neue Transkriptions-Preis berechnet. Dieser hängt vor allem von den gewählten APIs, der gewählten Sprache, dem damit verfügbaren Sprachmodell und der Audio-Dauer ab. Die API Auswahl ist für Benutzer der Einfachheit halber deaktiviert. Mit einem Druck auf «Transcribe!» wird die Audio-Datei transkribiert.

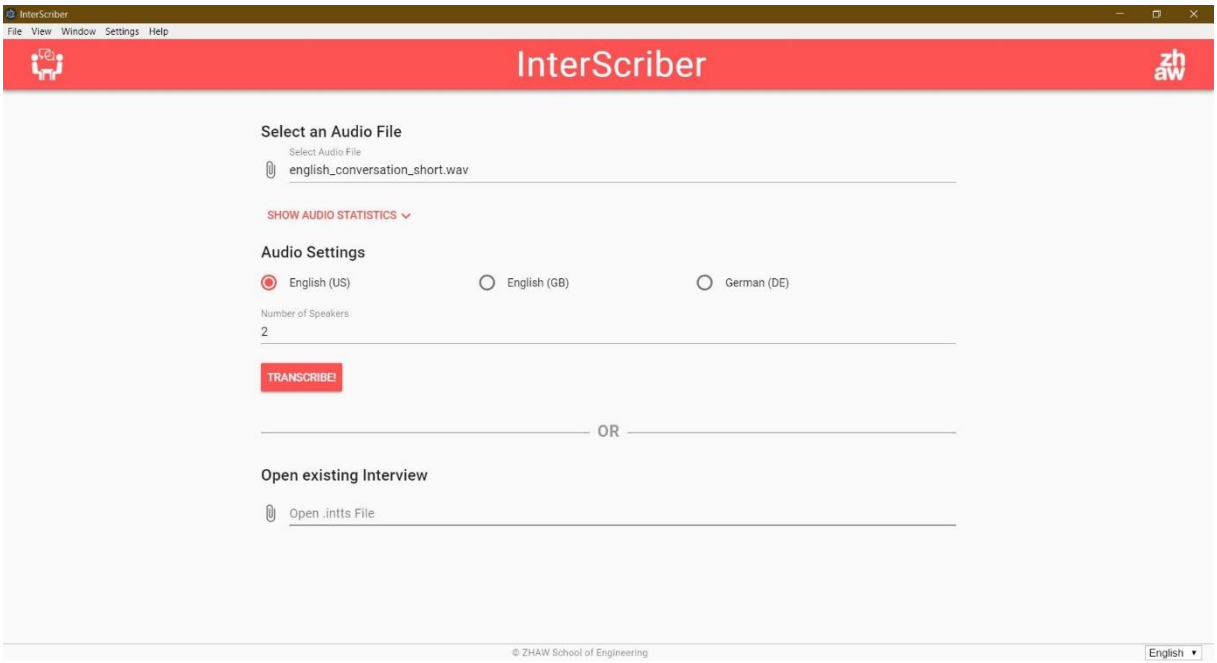


Abbildung 10: InterScriber - Audio Einstellungen

Der zweite Schritt ist die eigentliche Transkription (vgl. Abbildung 11). Es wird ein Fortschrittsbalken, die vergangene Zeit und die geschätzte Zeitdauer angezeigt.

Abbildung 11: InterScriber – Transkription

Das Frontend übermittelt die Einstellungen an das Backend. Dort werden zuerst die APIs konfiguriert, anschliessend wird die Audio-Datei an die Speech-to-Text API geschickt. Wenn das Resultat erhalten wurde, werden die Elemente für das Transkript erstellt, anschliessend wird die Audio-Datei an die Diarization API geschickt und nach der Antwort werden die erkannten Sprecher erstellt. Danach werden die Sprecher-Rahmen (speaker frames) aufgebaut und an das Frontend weitergegeben. Weil in dieser Arbeit die Google-Cloud- und IBM-Cloud-API STT und Diarisierung unterstützen, muss die Audio-Datei nur einmal an die API übertragen werden. Aus der Antwort werden zuerst die Elemente berechnet, anschliessend werden bei der Diarisierung die Sprecher aus der gleichen Antwort extrahiert.

Im dritten Schritt kann das Transkript bearbeitet und Fehler bei der Transkription verbessert werden (vgl. Abbildung 12). Als Hilfestellung kann das Audio immer wieder angehört werden. Dabei werden gesprochene Worte im Transkript rot hervorgehoben (in Abbildung 12 ist dies „please“ bei der fünften Äusserung). Zusätzlich kann die Wiedergabegeschwindigkeit angepasst werden (bei Playback Speed: 1x, 0.5x, 1.5x, etc.). Das Abspielen und Pausieren der Audio-Datei ist ausserdem über die Leertaste möglich.

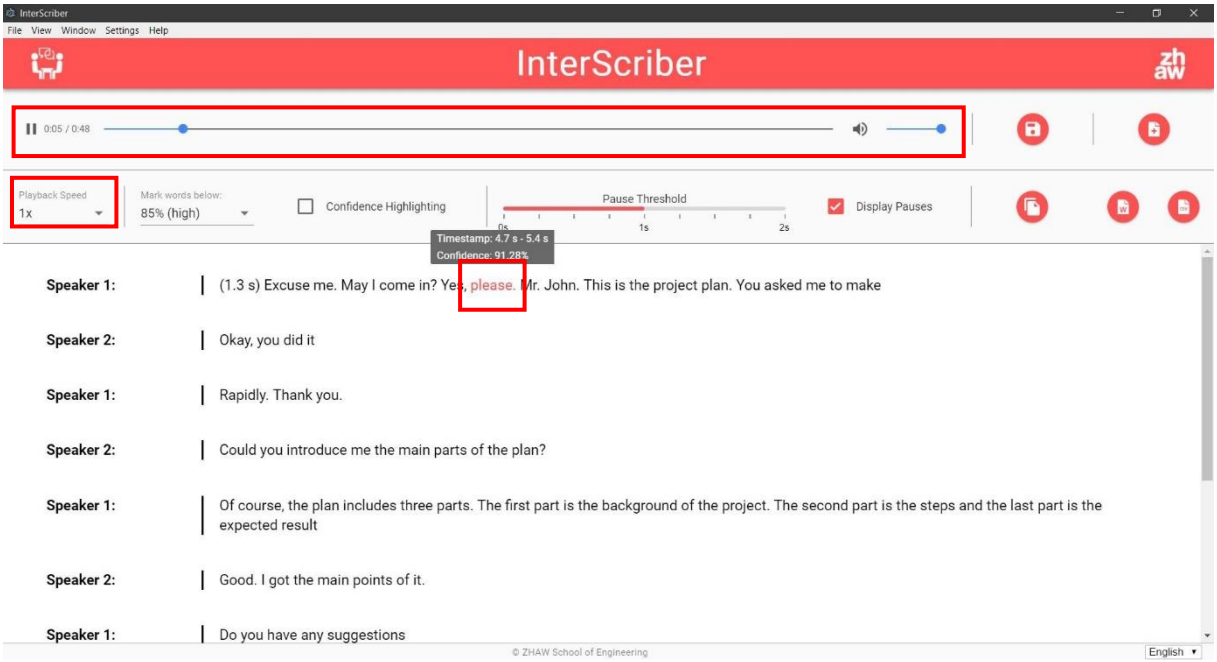


Abbildung 12: InterScriber – Editor

Eine weitere Hilfe ist es, die Zuversicht-Werte farbig anzeigen zu lassen, um Textstellen zu finden, bei der die API nicht sicher war, ob diese richtig transkribiert wurden. In Abbildung 13 werden alle Worte mit einer Zuversicht von weniger als 85% gelb markiert. Bei unter 60% werden die Worte rot markiert. Zusätzlich kann die Länge der angezeigten Sprecherpausen eingestellt oder die Pausen, je nach Bedarf, sogar komplett ausgeblendet werden.

Auf der rechten Seite der Editor-Einstellungen findet man verschiedene Schaltflächen. Derjenige oben links handhabt das Speichern von Interviews. Dabei kann gewählt werden, ob eine bestehende Datei überschrieben werden soll, falls das Interview bereits gespeichert wurde (Ctrl+S) oder ob eine neue Datei erstellt werden soll (Ctrl+Shift+S). Bei einer neuen Datei geht ein Fenster auf, worin gewählt werden kann, wo und unter welchem Namen das Interview gespeichert werden soll. Direkt daneben befindet sich die «Home» Schaltfläche. Durch Klick auf diesen Button gelangt man zur Startseite. Falls noch ungespeicherte Änderungen im Transkript vorhanden sind, erscheint eine Sicherheitsabfrage, damit keine Änderungen unabsichtlich verloren gehen.
In der unteren Leiste befinden sich die Export Möglichkeiten. Zum einen kann das Transkript in die Zwischenablage kopiert und nach Belieben weiterverwendet werden (Button ganz links in der unteren Leiste). Zum anderen kann das Transkript nach MS-Word (in der Mitte) oder CSV (ganz rechts) exportiert werden. Dabei öffnet sich, wie bei der Speicherung von Interviews, ein Fenster, wo der Ort und Datei-Name ausgewählt werden können.

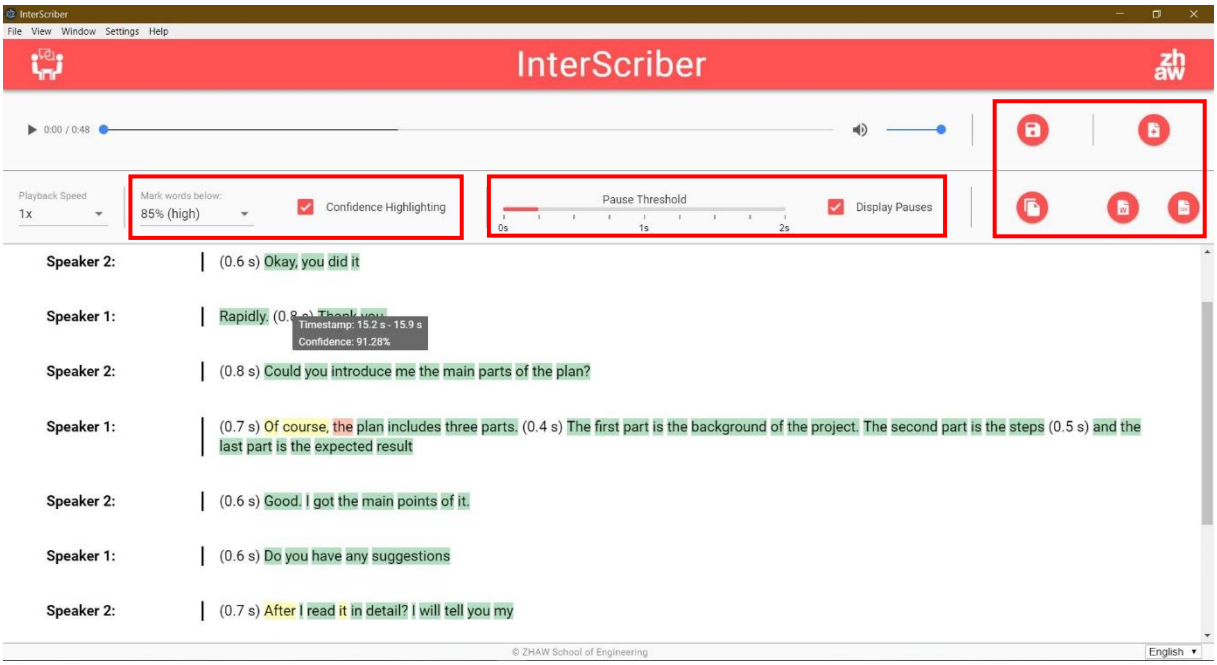


Abbildung 13: InterScriber - Editor mit Confidence-Highlighting

Zum Korrigieren und Bearbeiten des Transkriptes gibt es viele verschiedene Funktionen, welche nun genauer erläutert werden. Bei jeder Änderung wird das Backend aktualisiert, damit das «Interview» Objekt immer auf dem aktuellen Stand ist.
In Abbildung 14 ist ein Ausschnitt aus der Benutzeroberfläche dargestellt. Man sieht, dass die letzte Äusserung anders dargestellt wird, weil es eine parallele Äusserung ist. Dabei reden zwei oder mehr Sprecher gleichzeitigt. Wie diese generiert werden können, wird nachfolgend erklärt.

Ausserdem sieht man einen Tooltip, welcher beim Überschweben mit der Maus auf Elementen, wie Worte, Pausen und Zögern, und Sprechern erscheint. Darin wird die Zeit angezeigt, von wann bis wann dieses Element im Audio auftritt und bei Worten und Zögern zusätzlich auch die Zuversicht.

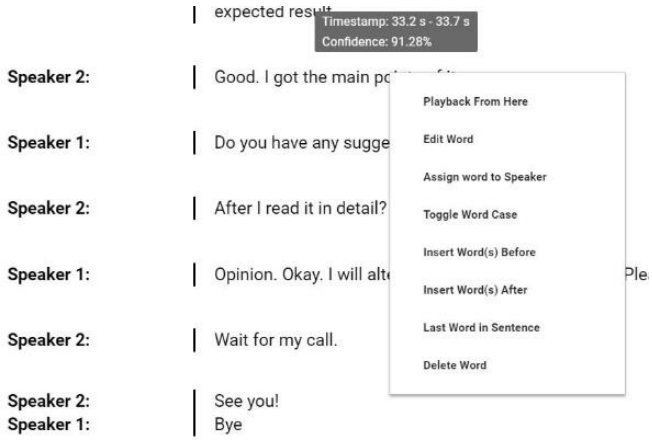


Abbildung 14: InterScriber - Wort Kontextmenü

Macht man einen Rechtsklick auf ein Wort oder Zögern, öffnet sich ein Kontextmenü. Dieses beinhaltet das Abspielen des Audios ab dem ausgewählten Element (bzw. mit Linksklick) und das Editieren, bei dem ein Input-Feld generiert wird (Doppelklick auf ein Element). Die nächste Option ist das Zuweisen des Elements einem Sprecher. Dies wird über einen speziellen Dialog gemacht (vgl. Abbildung 15).

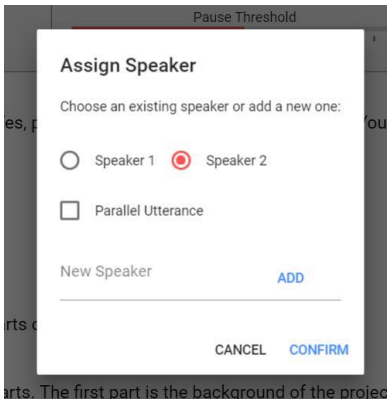


Abbildung 15: InterScriber – Sprecherzuweisungs-Dialog

In diesem Dialog kann der richtige Sprecher ausgewählt, bei Bedarf ein komplett neuer Sprecher hinzugefügt werden (oberhalb von «Cancel» und «Confirm») oder die Äusserung als parallele Äusserung klassifizieren. Dadurch wird diese Äusserung zusammen mit anderen, umliegenden, parallelen Äusserungen dargestellt. Durch einen Klick auf «Cancel» wird der Vorgang abgebrochen, bei «Confirm» wird die Änderung bestätigt.
Je nach gewähltem Wort kann es vorkommen, dass die Äusserung in drei einzelne aufgeteilt werden müssen, falls das Wort in der Mitte einer Äusserung ist. Falls das Wort am Anfang oder am Ende einer Äusserung ausgewählt wurde, wird es der vorherigen, respektive der nachfolgenden Äusserung zugewiesen, sofern die Sprecher und der Äusserungstyp (parallel oder normal) übereinstimmen. Ansonsten wird eine neue Äusserung mit diesem Sprecher erstellt.

Im vorher gezeigten Kontextmenü gibt es ausserdem noch folgende Funktionen: Die Gross- / Kleinschreibung ändern, Worte vor- oder nach dem ausgewählten Element einfügen, einen neuen Satz nach diesem Wort beginnen und das ausgewählte Element löschen. Dabei wird die Zeitdauer in eine Pause umgewandelt, damit die Zeit nicht verloren geht. Das Einfügen von neuen Worten ist auch durch einen Doppelklick auf die Leerzeichen zwischen den Elementen möglich.

Durch einen Rechtsklick auf eine Selektion von Wörtern, kann das Selektions-Kontextmenü aufgerufen werden (vgl. Abbildung 16). In diesem Menü kann ausgewählt werden, dass die markierten Wörter nacheinander immer wieder abgespielt oder dass sie einem anderen Sprecher zugewiesen werden sollen. Dabei wird der Sprecherzuweisungs-Dialog verwendet.

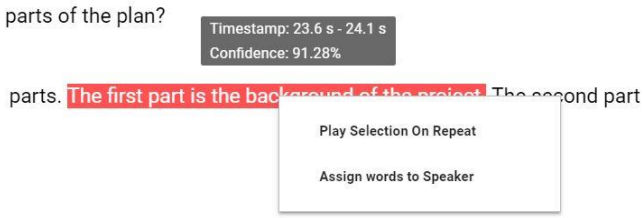


Abbildung 16: InterScriber – Selektion Kontextmenü

Werden alle Worte einer Äusserung markiert oder wird ein Rechtsklick auf einen Sprecher ausgeführt, öffnet sich das Sprecher Kontextmenü (vgl. Abbildung 17).

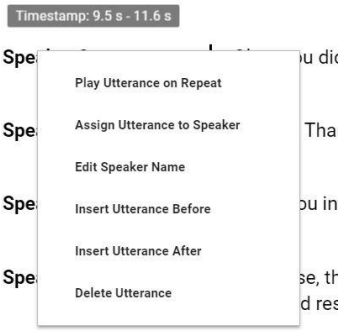


Abbildung 17: InterScriber - Sprecher Kontextmenü

In diesem Menü hat man die Möglichkeit, die komplette Äusserung immer wieder abzuspielen, wie es auch bei markierten Worten der Fall ist. Man kann auch die komplette Äusserung einem anderen Sprecher zuweisen, dazu wird wieder der Sprecherzuweisungs-Dialog verwendet. Die dritte Option ist, den Sprechernamen zu ändern, was auch mit Doppelklick auf den Namen möglich ist. Standardmässig werden Sprecher immer im Format «Sprecher 1:» angezeigt, es sei denn ein Name wurde vergeben. Das Standardformat wurde so gewählt, dass bei verschiedenen UI-Sprachen auch die Sprecher richtig übersetzt werden. Die nächsten beiden Optionen sind, eine neue Äusserung vor oder nach der gewählten einzufügen. Dafür wird der Dialog in Abbildung 18 verwendet.

Dieser Dialog ist ähnlich wie der Sprecherzuweisungs-Dialog aufgebaut, mit der Besonderheit, dass mehrere Sprecher mit zugehörendem Text erstellt werden können. Wie der Abbildung entnommen

werden kann, können weitere Sprecher über das «+» hinzugefügt und mit «-» wieder entfernt werden, sofern es eine parallele Äusserung sein soll. Falls dieses Kontrollkästchen deselektiert ist, wird lediglich ein Sprecher angezeigt. Durch einen Klick auf «Create» wird die Äusserung erstellt.

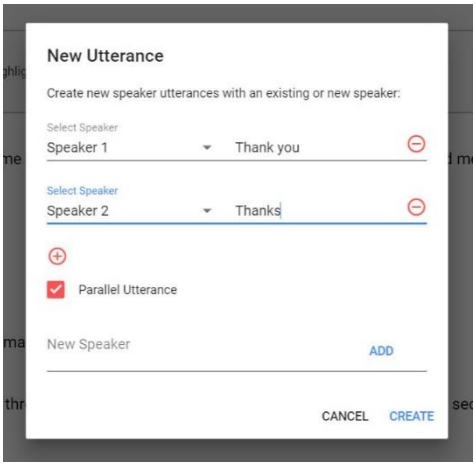


Abbildung 18: InterScriber – Neue Äusserung Dialog

Die letzte Aktion im Sprecher Kontextmenü ist das Löschen einer Äusserung. Wird diese Option gewählt, wird ein Bestätigungsdialog angezeigt, um zu bestätigen, dass die Äusserung auch wirklich gelöscht werden soll.

4.5 Herausforderungen

Im Folgenden werden die Herausforderungen und Problemstellen, die im Verlauf dieser Arbeit aufgetreten sind, detailliert und erklärt, wie diese gelöst wurden.

4.5.1 Fehlerhafte deutsche Diarisierung

Gegen Ende der Arbeit wurde bemerkt, dass bei manchen deutschen Interviews die Sprechererkennung fehlschlägt, und die gesamte Konversation als eine Aussage interpretiert wird. Das Problem liess sich reproduzieren, und es wird vermutet, dass es durch eine Kombination von ähnlich klingenden Sprechern und dem deutschen Sprechermodell von Google, welches sich in einem experimentellen Zustand befindet, verursacht wird.
Da kein Einfluss auf das Sprechermodell genommen werden kann, wurde eine Zwischenlösung implementiert, wobei beim Auftreten dieses Problems die Konversation zusätzlich als englisch diarisiert wird. Anhand der englischen Sprecher-Aufteilung und dem deutschen Transkript, wird schlussendlich ein korrektes Transkript erstellt.

4.5.2 Zeitstempel

Die Dienste von Google und IBM liefern jeweils die Zeitfenster in der Audio-Datei, wo sich ein gewisses erkanntes Wort befindet. Diese Information macht sich «InterScriber» zu nutzen, indem es das Transkript mit dem Audio-Player verknüpft (vgl. Abschnitt 4.1).
Da der Benutzer jedoch die Möglichkeit hat, das Transkript frei zu bearbeiten, müssen wird drauf achten, dass die Zeitstempel jedes Wortes so akkurat wie möglich bleiben. Der erste Ansatz bestand daraus, eine Sprechrate für jeden Sprecher zu berechnen. Die Idee war, dass das Zeitfenster einer Äusserung immer gleichbleiben soll, und diese Zeit bei Änderungen mittels der Sprechrate auf die

Wörter neuverteilt wird. Obwohl diese Methode initiale Erfolge aufwies, gerieten Audio und Text bei besonders langen Äusserungen oftmals komplett aus der Synchronisation.
Mit dem zweiten Ansatz sollte die durch Änderungen entstehende zeitliche Ungenauigkeit auf einen möglichst kleinen Teil des Transkriptes reduziert werden. Erreicht wurde das, indem neu eingefügte Wörter einen Teil der Zeit von den Umliegenden erhalten. Von den direkten Nachbarn wird am meisten Zeit genommen und von den weiter Entfernten immer weniger.

4.5.3 Externes Backend

In der vorangegangenen Arbeit sollten sich sämtliche Teilsysteme auf dem PC des Endbenutzers befinden (vgl. Abschnitt 4.1). Es wurde jedoch bereits damals erkannt, dass die Installation eines solchen Systems impraktikabel war. Entsprechend wurde in dieser Arbeit die Entscheidung getroffen, das Python Programm (Backend) auf einen zentralen Webserver auszulagern.
Erste Schwierigkeiten ergaben sich beim Hochladen und Streamen der Audio-Datei, wie auch dem Öffnen und Speichern eines Interviews. Anstatt dass das Backend direkt auf das Datei-System Zugriff hat, musste die HTTP Schnittstelle erweitert werden, sodass die Dateien korrekt kommuniziert werden können.
Eine weitere Problematik, die sich offenbarte, war, dass das Backend nun für mehr als einen Benutzer zuständig ist. Da das Programm ursprünglich nicht für diesen Zweck gedacht war, waren keine getrennten Sessions oder ähnliche Funktionalität vorhanden. Aus zeitlichen Gründen war keine saubere Lösung im Rahmen dieser Arbeit mehr möglich, da andere, höher priorisierte Funktionen, zu machen waren. Als vorläufige Lösung wurden mehrere Instanzen des Webservers erzeugt, wodurch mehrere Benutzer zeitgleich an «InterScriber» arbeiten können.
Das Backend auf einem externen Server zu starten gestaltete sich anfangs als anspruchsvoll. Mithilfe eines Artikels von R. Nayak [27] konnte die Migration erfolgreich abgewickelt werden. Der anschliessende Aufwand für neue Versionen war sehr klein, da lediglich die Dateien ersetzt und der «supervisor» Service neugestartet werden musste.

4.5.4 Performanz

Eine Herausforderung, die bis zum Ende der Arbeit teilweise ungelöst blieb, bezieht sich auf die Performanz der Desktop Applikation bei längeren Interviews. Ab einer Länge von einer Stunde machen sich Verzögerungen zwischen der Benutzereingabe und der Rückmeldung bemerkbar. Das Problem liegt daran, dass bei einer solchen Länge in der Editor Komponente zu viele Elemente dargestellt werden müssen. Nach mehreren Anpassungen an dieser Komponente wurde das Problem für gewisse Funktionen bereits gelöst, aber es ist noch immer präsent.
Zwei mögliche Lösungen, mit welchen sich das Problem beseitigen lassen könnte, sind zum einen das Verwenden einer passenderen Datenstruktur für die Transkript-Elemente und zum anderen das Herunterbrechen der Editor Komponente auf kleinere Vue.js Komponenten. Durch den zweiten Ansatz würde die Rendering-Methode von Vue.js besser ausgenützt, was wiederum die Reaktionszeit verbessern könnte.

5 Evaluation

In der vorangegangenen Arbeit wurde die Transkriptions-Qualität der APIs evaluiert. Da in dieser Arbeit der Fokus auf der Bedien- und Benutzbarkeit der Anwendung und nicht auf der Qualität der APIs lag, konzentriert sich die Evaluation auf zwei verschiedene Benutzer-Aspekte. Zum einen wurde der Transkriptionsaufwand evaluiert. Die Frage war, wie lange es dauert, bis ein Interview mit «InterScriber» transkribiert und korrigiert wurde. Zum anderen wurde das Programm während der Entwicklung mehreren Testern vorgelegt, um Rückmeldungen über die Benutzererfahrung zu sammeln.

5.1 Transkriptionsaufwand

Für die Evaluation des Transkriptionsaufwandes wurden mehrere Interviews von ungefähr 20 bis 30 Minuten transkribiert und anschliessend korrigiert. Dabei konnten Verbesserungen und Fehler entdeckt und teilweise bereits korrigiert werden.

Eine Transkription eines Interviews mit einer Dauer von 20 Minuten dauert im Schnitt etwa gleich lange, wie die Audio-Dauer an sich. Diese Faustregel gilt für beide Dienste von Google und IBM. Für die Transkription spielt auch die Bandbreite des Benutzers eine Rolle, weil die Audio-Datei zu Beginn hochgeladen werden muss. Wie zu erwarten kann dies bei grösseren Daten und niedrigen Upload-Raten einige Zeit kosten.

Wenn das Transkript im Editor verfügbar ist, gilt es die Text- und Diarisierungsfehler zu korrigieren. Dabei müssen vor allem Worte am Ende einer Äusserung einem anderen Sprecher zugewiesen werden, Satzzeichen gesetzt und Äusserungen aufgeteilt werden. Das Problem der fehlenden Satzzeichen ist primär bei deutschen Transkripten präsent.

Während des Arbeitens wurde gemerkt, dass das Zuweisen von Worten an einen Sprecher zu viele Klicks braucht. Anfänglich waren es sechs Klicks vom Auswählen eines Wortes, über Selektieren der Funktion, Auswählen des Sprechers bis zum Bestätigen. Diese Anzahl konnte bereits auf vier Klicks reduziert werden, in dem standardmässig ein anderer Sprecher beim Zuweisen der Worte an eine Äusserung ausgewählt ist und mittels Radiobuttons dargestellt wird und nicht noch ein Dropdown-Menü geöffnet werden muss.
Es wurde auch vermerkt werden, dass es praktikabler wäre, wenn mehrere Worte oder sogar die ganze Äusserung in einem bearbeitet werden könnte, anstatt nur ein einzelnes Wort. Das Problem dabei ist, dass die Zeitdauer der Elemente dabei nicht verloren gehen und wie Pausen gehandhabt werden müssen.

Generell kann gesagt werden, dass das Korrigieren von deutschen Interviews mehr Zeit in Anspruch nimmt, da mehr Worte und Sprecher korrigiert werden müssen, weil die Sprachmodelle für Deutsch noch nicht so ausgereift sind wie für amerikanisches Englisch. Bei solchen Interviews muss weniger korrigiert werden.
Bei deutschen Interviews kommt noch dazu, dass die Satzzeichen oftmals fehlen und diese manuell ergänzt werden müssen. Dafür wurde bereits die Funktion hinzugefügt, um ein Satzende zu markieren. Dann wird nach dem gewählten Wort ein Punkt gesetzt und das nächste Wort wird grossgeschrieben.

Eine nützliche Funktion wäre, Sätze bereits nach der Transkription automatisch zu erkennen und mit Satzzeichen zu versehen. Eine weitere Funktion, welche dem Benutzer viel Arbeit ersparen würde, wäre es, wenn korrigierte Wörter festgehalten werden und gelernt wird, welche Wörter meist falsch transkribiert wurden. Diese Wörter könnten automatisch korrigiert werden. Zusätzlich wäre es für den Benutzer praktisch, er könnte Eigennamen angeben, welche im Interview vorkommen, denn meist werden Namen in Interviews falsch und unterschiedlich transkribiert, da diese dem Sprachmodell nicht bekannt sind. Durch Verwendung einer solchen Liste könnten häufige Fehler maschinell korrigiert werden.

Im Schnitt wurde für das Bearbeiten eines Transkripts ca. 45 Minuten für etwa 20 bis 30 Minuten Audio aufgewendet. Bei englischen Interviews etwas weniger, bei deutschen dafür etwas mehr.

5.2 Benutzer Feedback

Die Applikation wurde von mehreren Benutzern im Parallelen zur Entwicklung getestet, um Verbesserungsmöglichkeiten und Probleme zu identifizieren. Grössenteils haben sich die Rückmeldungen auf das Benutzeroberfläche konzentriert, in Form vieler kleiner Anpassungsmöglichkeiten. Diese wurden jeweils in den Product Backlog aufgenommen und in der nächsten Iteration diskutiert und priorisiert (vgl. Kapitel 8).
Die Transkriptionsqualität hing laut Aussagen der Tester stark von den Akzenten ab, war jedoch bei klaren, akzentlosen Sprechern «gut». Bei solchen mit starken Akzenten war es laut einem Tester einfacher die Transkription von Hand zu machen. Diese Rückmeldungen decken sich mit den Erwartungen dieser Arbeit.
Ein weiterer Kritik-Punkt war die lange Wartezeit bei grossen Dateien. Diese entsteht zum einen durch das doppelte Hochladen zum Webserver und dann zur externen API, und durch die eigentliche Transkription der Aufnahme.

6 Diskussion und Ausblick

Das Ziel dieser Arbeit war es, in «InterScriber» ein vollständiges System für die Transkription von Interviews zu entwickeln. Der Arbeitsablauf, vom Hochladen einer Aufnahme, über das transkribieren und korrigieren, bis hin zum Export wurde implementiert. Während eine geplante User Story (vgl. US7 in Abschnitt 4.2.1) und viele im Laufe der Arbeit identifizierte Ideen (vgl. Product Backlog in Abschnitt 8.2.2) offenbleiben, wurden die zu Beginn der Arbeit festgelegten Ziele erreicht.

Aus Hinsicht des Projektmanagements verlief die Arbeit grundsätzlich reibungslos. Von der wöchentlichen Planung, zur Arbeitsteilung in der Gruppe, bis zur Zusammenarbeit mit der Betreuungsperson sind keine nennenswerte Schwierigkeiten aufgetreten. Aus den Erfahrungen der vorgegangenen Projektarbeit haben wir ebenfalls gelernt und haben mit mehr Audio-Dateien selbst getestet, sowie die Meinung von Dritten eingeholt.

In kurzfristiger Hinsicht bestehen noch eine Handvoll Aufgaben die gemacht werden müssen, welche bereits in Abschnitt 4.5 bei den Problemen und Herausforderungen erwähnt wurden. Die dringlichsten davon sind: Das Adressieren von Performanz-Problemen im Editor bei sehr langen Interviews und das Fehlen von Session-Management beim Flask Server.
Abgesehen von diesen Schwierigkeiten würden sich zudem kleinere Verbesserungen am Editor und der Benutzeroberfläche anbieten. Während beispielsweise alle nötigen Funktionen im Editor vorhanden sind, gäbe es verschiedene Wege, um die Nutzererfahrung zu steigern. Möglichkeiten dafür wären: Wörter oder Satzteile mit «Ziehen und Ablegen» anderen Sprechern zuweisen zu können, das Transkript als Fliesstext und nicht nur einzelne Wörter bearbeiten zu lassen oder dem Benutzer die Option zu geben, die Zeitstempel der Wörter und Segmente anzupassen.
Ein weiterer Diskussionspunkt ist das Portieren der Electron Applikation zu einer Web-Applikation. Ein Machbarkeitsnachweis wurde im Rahmen dieser Arbeit bereits gemacht und es würde den Installationsaufwand für Endbenutzer eliminieren.

Sobald diese Fragen geklärt sind, kann man sich im langfristen Zeitraum grösseren Erweiterungen widmen. Da diese Applikation in Zukunft kommerziell verfügbar sein soll, bieten sich mehrere Features an:

- Da es oft nicht im Sinne der Konversationsteilnehmer ist, die Aufnahme auf Servern im Ausland (unter tieferen Datenschutzstandards) bearbeiten zu lassen, wäre es wünschenswert, eine schweizerische Lösung anzubieten. Dafür könnten Open Source oder eigene Lösungen für ASR und Diarization an «InterScriber» angebunden werden.
- Für die kommerzielle Funktionsfähigkeit müsste ein Benutzer Management-System implementiert werden, sodass Daten, Zugriffe, Nutzung und Kosten klar separiert werden können.
- Entsprechend dem obigen Punkt, müsste ein Zahlungsverfahren implementiert werden, sodass die Nutzung der Applikation verrechnet werden kann.
- Eine potenziell erstrebenswerte Erweiterung wäre eine Lösung, mit der auch schweizerdeutsche Interviews transkribiert werden könnten, da es zurzeit noch keine Anbieter für diesen Anwendungsfall gibt.

7 Verzeichnisse

7.1 Literaturverzeichnis

[1] N. Eckhart, M. Marxer, M. Ulasik, "Projektarbeit (Informatik). Automatische Transkription von Interviews", School of Engineering, Zurich University of Applied Sciences, Winterthur, 2019

[2] B. Pfister, T. Kaufmann, "Sprachverarbeitung, Grundlagen und Methoden der Sprachsynthese und Spracherkennung", Springer Vieweg, 2017, S. 22-28

[3] D. Kolossa, "Grundlagen der automatischen Spracherkennung", Institut für Kommunikationsakustik, Ruhr-Universität Bochum, 21. April 2017, S. 73-86

[4] E.G. Schukat-Talamazzini, "Automatische Spracherkennung. Statistische Verfahren der Musteranalyse", Vieweg Verlag, 1995, S. 45-74

[5] A. Hallerbach, "HMM/KNN zur Spracherkennung", Universität Ulm, 2005, S. 3-6

[6] K. Schulz, Ch. Ringlstetter, F. Schiel, "Sprachmodelle", Universität München, 2006

[7] T. Kemp, M. Schmidt, M. Westphal, A. Waibel. *Strategies for automatic segmentation of audio data* [Online].
URL:
https://www.ee.columbia.edu/~dpwe/papers/KempSWW00-audseg.pdf
[Stand: 15.12.2018]

[8] X. Anguera, *Metric-Based Segmentation* [Online].
URL:
http://www.xavieranguera.com/phdthesis/node12.html [Stand: 15.12.2018]

[9] Wreally LLC. *Transcribe audio to text with minimal effort* [Online].
URL:
https://transcribe.wreally.com/ [Stand: 12.05.2019]

[10] Wreally LLC. *Frequently asked questions* [Online].
URL:
https://transcribe.wreally.com/faq#auto-transcribe-questions-1 [Stand: 12.05.2019]

[11] TranscribeMe Inc. *Flexible, On Demand Pricing* [Online].
URL:
https://www.transcribeme.com/pricing/ [Stand: 17.05.2019]

[12] CGBiz Corporation. *Flat Transcription Rates, Charged Per Minute of Audio* [Online].
URL:
https://scribie.com/transcription/pricing/ [Stand: 17.05.2019]

[13] Rev.com Inc. *Services* [Online].
URL:
https://www.rev.com/services/ [Stand: 17.05.2019]

[14] GoTranscript.com. *Pricing Calculator* [Online].
URL:
https://gotranscript.com/transcription-cost-estimate/ [Stand: 17.05.2019]

[15] Sonix, Inc. *The best automated transcription software powered by cutting-edge AI* [Online].
URL:
https://sonix.ai/features/ [Stand: 17.05.2019]

[16] Temi.com. *How long does Temi take to transcribe?* [Online].
URL:
https://help.temi.com/about-temi/how-long-does-temi-take-to-transcribe/
[Stand: 17.05.2019]

[17] Temi.com. *Editor* [Online].
URL:
https://www.temi.com/editor/ [Stand: 17.05.2019]

[18] Trint Ltd. *How does Trint's automated transcription work?* [Online].
URL:
https://trint.com/how-it-works/ [Stand: 17.05.2019]

[19] Google Cloud. *Cloud Speech-to-Text* [Online].
URL:
https://cloud.google.com/speech-to-text/ [Stand 29.05.2019]

[20] IBM Watson. *Speech to Text* [Online].
URL:
https://www.ibm.com/watson/services/speech-to-text/ [Stand 29.05.2019]

[21] Microsoft Azure. *Speech to Text* [Online].
URL:
https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
[Stand 29.05.2019]

[22] Temi.com. *Temi API* [Online].
URL:
https://www.temi.com/api [Stand 29.05.2019]

[23] SeerNet. *Multi-Speaker-Diarization* [Online]
URL:
https://github.com/SEERNET/Multi-Speaker-Diarization [Stand 29.05.2019]

[24] DeepAffects. *Speech analysis APIs for developers* [Online]
URL:
https://www.deepaffects.com/apis/ [Stand 29.05.2019]

[25] Remeeting.com. *ASR API* [Online]
URL:
https://remeeting.com/console/docs [Stand 29.05.2019]

[26] Medium.com. *Deploy flask app with nginx using gunicorn and supervisor* [Online].
URL:
https://medium.com/ymedialabs-innovation/deploy-flask-app-with-nginx-using-gunicorn-and-supervisor-d7a93aa07c18 [Stand 30.05.2019]

[27] A. Duranti. *Transcript* [Online]
URL:
http://www.sscnet.ucla.edu/anthro/faculty/duranti/audvis/annotate.htm
[Stand 02.06.2019]

7.2 Bildverzeichnis

Abbildung 1	„AudioText Editor“ von Sonix zur nachträglichen Korrektur des ASR Transkriptes
Abbildung 2	Editor von Temi zur nachträglichen Korrektur des ASR Transkriptes
Abbildung 3	Editor von Trint zur nachträglichen Korrektur des ASR Transkriptes
Abbildung 4	Benutzeroberfläche des bestehenden „Interview Transcription Tool“
Abbildung 5	InterScriber - Physische Architektur
Abbildung 6	Backend-Klassendiagramm – App und Service Klassen
Abbildung 7	Backend-Klassendiagramm – SpeechToText und Diarization Klassen
Abbildung 8	Backend-Klassendiagramm – Interview- und Unterklassen
Abbildung 9	InterScriber - Startseite
Abbildung 10	InterScriber - Audio Einstellungen
Abbildung 11	InterScriber - Transkription
Abbildung 12	InterScriber - Editor
Abbildung 13	InterScriber - Editor mit Confidence-Highlighting
Abbildung 14	InterScriber - Wort Kontextmenü
Abbildung 15	InterScriber - Sprecherzuweisungs Dialog
Abbildung 16	InterScriber - Selektion Kontextmenü
Abbildung 17	InterScriber - Sprecher Kontextmenü
Abbildung 18	InterScriber - Neue Äusserung Dialog

7.3 Tabellenverzeichnis

Tabelle 1	ASR Zusammenfassung - Wreally
Tabelle 2	ASR Zusammenfassung - TranscribeMe
Tabelle 3	ASR Zusammenfassung - Scribbie
Tabelle 4	ASR Zusammenfassung - Sonix
Tabelle 5	ASR Zusammenfassung - Temi
Tabelle 6	ASR Zusammenfassung - Trint
Tabelle 7	Frontend-Komponenten von InterScriber

8 Anhang

8.1 Offizielle Aufgabenstellung, Projektauftrag

Die Bachelorarbeit «Automatische Transkription von Interviews» wurde für das Frühlingssemester 2019 von Prof. Dr. Mark Cieliebak folgendermassen ausgeschrieben:

Beschreibung:
Wie kann man ein Gespräch, z.B. ein Zeitungs-Interview oder ein Bewerbungsgespräch, automatisch in geschriebenen Text transkribieren? Dazu braucht es zwei wesentliche Komponenten:

- Speech2Text, um die gesprochene Sprach in Text umzuwandeln
- Speaker Identification, um festzustellen wer zu welchem Zeitpunkt redet

Für beide Komponenten gibt es heute Lösungen: Speech2Text wird z.B. von Google als API angeboten, und für die Speaker Identification wird hier an der ZHAW zurzeit ein System mit Deep Learning entwickelt (siehe PA stdm_2).

Ziel der Arbeit:
In dieser Projektarbeit soll ein "komplettes" System für Interview-Transkription auf Deutsch entwickelt werden. Das Ziel ist ein vollständiger Workflow von der Audio-Aufnahme zum Dialog-Transkript.

Teilaufgaben:

- Evaluation von bestehenden Lösungen für Englisch und (falls vorhanden) Deutsch für Interview-Transkription und die wesentliche Teilkomponenten
- Modulare Architektur für das Gesamtsystem festlegen
- Implementierung/Integration der einzelnen Teilkomponenten
- Kombination der Teilkomponenten zu einem "fertigen" Transkript
- Evaluation der Ergebnisqualität auf realen Interview-Daten
- Vorschläge, wie das System verbessert/optimiert werden könnte

8.2 Projektmanagement

8.2.1 Projektplan

W	Von	Bis	Aufgaben
23	4.6	7.6	BA Abgabe am 7. Juni
22	28.5	3.6	BA Bericht
21	21.5	27.5	BA Bericht
			Frontend als Webseite (Machbarkeitsnachweis)
			Sprecher Eingabefeld soll nicht leer sein, wenn man doppelklickt
			In den Dialogfenstern anstatt Dropdowns für die Sprecher Radio Buttons verwenden.
20	14.5	20.5	Speichern unter soll mehrmals möglich sein (anstatt nur einmal)
			Bug: Deutsche Diarization - Problem mit zwei Sprechern (Beispiele sammeln)
			Szenario behandeln: Parallele Sprecher (von der Transkription bis zum Editor)
			«Einem neuen Sprecher zuweisen» Option im Kontextmenu ist zurzeit zu langsam und braucht zu viele Klicks > Vereinfachung: Im Kontextmenü einen anderen Sprecher in der Selektion anzeigen
			Mit Evaluations-Gruppe kurzschliessen ob es möglich ist Mozilla zu integrieren
			Bug: Bei Bestätigungsmeldungen (Toasts) wird kein Text angezeigt
			Styling: Eingabefeld beim Wort-Editieren ist zu lange (Länge am besten dynamisch berechnen)
			Ein “s” hinter die Pausenlänge in den Klammern
19	7.5	13.5	Bug: Wenn bei der Selektion auch der Sprecher markiert wird, ist die Auswahl manchmal nicht gültig
			«Confidence Threshold» Einstellung ist unverständlich und muss vereinfacht werden
			Die Schriftgrösse sämtlicher Tooltips soll vergrössert werden
			Beim Scrollen sollen sämtliche offenen Kontextmenü geschlossen werden
			Das Hervorheben der Wörter soll deaktiviert werden, wenn das Audio pausiert wird (zurzeit bleibt das momentane Wort nach Pausieren einfach rot)
			Es soll ein beschreibender Titel über den «Pausenlänge» Regler gemacht werden
			Doppelklick auf einen Sprecher soll ebenfalls das Umbenennen aufrufen (wie bei einem Wort im Transkript)
			Doppelklick zwischen zwei Wörter soll ein leeres Feld für ein neues Wort dazwischen aufmachen

			Farbliches Hervorheben der Segmente soll wieder entfernt werden
18	30.4	6.5	Hervorheben muss exakter gemacht werden, sodass nicht umliegende Wörter (besonders die von anderen Segmenten) ebenfalls rot sind
			Die Segmente sollen farblich voneinander unterschieden werden
			Diarization Bug (Sprecheränderung)
			Zeitberechnung bei Wortänderungen
			Den Ladebalken beim Transkribieren verbessern
			API und Audio Einstellungen auf einen «Step» reduzieren (die API Auswahl soll versteckt werden als Entwickler-Option), die Audio Statistiken ausklappbar machen (mehr anzeigen) > Schritt 1 und 2 kombinieren und alles ein bisschen schmaler und zentrierter
			Sprachauswahl als Radio-Buttons
			Sprachen lesbarer darstellen (z.B. «Englisch (US)», «Deutsch (DE)»)
17	23.4	29.4	Fehler bei der Diarisierung korrigieren
			Ermöglichen, dass ein Dateiname bei "Speichern Unter" gewählt werden kann (+ Toolbar)
			Neben dem Export zum Clipboard soll das Transkript ebenfalls als Word und CSV exportierbar machen
16	16.4	22.4	~~Mehrere Audioformate unterstützen (mp3, mp4, m4a)~~ Erkenntnis: Nur WAV und FLAC werden bei Google und IBM (+ MP3) unterstützt, konvertieren nur bei mp3 möglich
			Einfache Anleitung für Benutzung der Software schreiben (für SW-Tester)
			Eigenes Menü für Anwendungs-Einstellungen mit den wichtigsten Operationen erstellen (Speichern, Speichern unter, Interview öffnen, Audio hochladen, etc.)
			Deutsche Übersetzungen machen & Sprachauswahl in der App ermöglichen
			Mehrere Wörter sollen nach markieren wiederholend abgespielt werden können
			Wort an Segmentanfang immer gross
15	9.4	15.4	"Speichern Unter" von Interviews überarbeiten (für das spätere migrieren des Python-Teils auf einen externen Webserver)
			i18n Lokalisierung im User Interface unterstützen
			Backend Server aufsetzen (auf externem ZHAW Webserver)
			Development und Production Modi für das Frontend erstellen, mit eigenen Konfigurationen (damit z.B. nach Kompilieren automatisch der ZHAW Webserver verwendet wird)
			Laden von Audio-Daten für externes Backend überarbeiten

			Öffnen von Interviews für externes Backend überarbeiten (siehe oberster Punkt)
			Toolbar (Audio + Einstellungen) sollen während dem Scrollen sichtbar bleiben
			Snackbar für Erfolg-Meldungen
			Fehler im Transkript korrigieren (ändern, hinzufügen, löschen)
			Das Theme der Applikation auf Light ändern (zurzeit sehr dunkel)
			Utterance, Markierung: Kleine Pause beim Wiederholen (1s)
			Confidence Highlight Threshold (Dropdown)
			Musik-Playback von der Leertaste ignorieren, sodass Leerzeichen geschrieben werden können
14	2.4	8.4	Klick auf einen Sprecher spielt nur dieses Segment ab
			Wörter nach Gewissheit in verschiedenen Farben markieren
			Tiefe Gewissheit im Transkript farblich hervorheben (ein- und ausschaltbar)
			InterScriber Branding (Icons, App Config, package.json, Titel, etc.)
			BA Titel finden
			Die "Hesitation" welche von der IBM API gefunden wird direkt im Backend behandeln
			Speichern mit Ctrl-S
			Das Audio soll mit der Leertaste pausiert und abgespielt werden können
			~~Input Felder anpassen, damit Unterschied zwischen Text und Bearbeitung nicht auffällt~~ (die Bedienung wird unklar, wenn kein Rahmen vorhanden ist)
			Wortänderungen sollen im Hintergrund vermerkt werden und die Zuversicht nach Änderung auf 100% gesetzt werden
			en-GB als Audio-Sprache unterstützen
			~~Checkbox für automatische Anzahl Sprecher erkennen~~ (Wenn nicht anders angegeben, setzt Google die Anzahl auf 2, IBM wird Anzahl nicht angegeben (ist aber auf 2 Sprecher optimiert))
13	26.3	1.4	Editor überarbeiten / Darstellung optimieren * Default Editor Text wird nicht mehr benötigt * Tooltips für Zuversichtswerte * Kontextmenü für Wörter und Sprecher
			Langsames Anhören der Audio-Datei, um direkt mitschreiben zu können
			Original Transkript speichern für langfristiges Lernen von verbesserten Wörtern (nur aktuelles Transkript überschreiben, Audio / Original nicht nötig)
			Ladebalken und Spinner bei Operationen wie: Audio hochladen, Interview laden, Audio transkribieren
			Weitere kleine Features: Backend Logger & Button für Neues Projekt, Rückkehr zum ersten Schritt: Info für Speichern, wenn ein neues Projekt gestartet wird

			Audio als Datei speichern, nicht als Base 64 (führt zu Problemen bei grossen Dateien)
12	19.3	25.3	Dateiformate umsetzen (+ Konverter-Klassen für beide APIs), Serializer-Klasse
			Mögliche Features als Issues erfassen und priorisieren
			Bug: Die Dauer des ersten Wortes eines Segmentes ist zu lange (untersuchen)
			Neugestaltung des UI gemäss Besprechung
			Vorbereitung für Audio Statistiken, Transkriptions-Dauer erfassen, um zukünftig Dauer vorherzusagen, erste Statistiken anzeigen
11	12.3	18.3	Komplettes Interview transkribieren, um gute Features und Probleme zu finden
			Konfigurations-Dateien für jede API / Metadaten standardisieren (API spezifische Werte in Konfigurations-File; Sprachmodelle)
			IBM Authentifizierung flicken (zurzeit hartcodiert) & Google Credentials ohne ENV variable laden
			Deutsche Sprache unterstützen
10	5.3	11.3	Bugfixing für bisher bekannte Bugs (Google API)
9	26.2	4.3	Issues erfassen + labeln
			IBM Watson zum Laufen bringen + evtl. bereits integrieren
			Name für Anwendung finden
			Überarbeiten der PA Software (& FE / BE trennen)
			Bericht Grundgerüst
			Interview Transkription Tools genauer anschauen
			UI-Cleanup + Vorbereitung für neue Features
8	19.2	25.2	Zeitplan
			Dateiformate für STT + DIR recherchieren
			Mit Evaluation-Gruppe kurzschliessen wegen Output-Format (ulasimal@students.zhaw.ch)
			Interview Transkriptions Tools (Konkurrenz) recherchieren
			Architektur definieren (Aufteilung von Frontend, Backend, STT, DIR + eigene Repository für Zusammenarbeit)

8.2.2 Product Backlog

Feature	**Bereich**
Änderungen rückgängig machen (Undo / Redo)	Editor

Alle gleichen Wortvorkommen anpassen (Liste mit allen Vorkommen und Kontext, welche einzeln gewählt werden können)	Editor
Alternativen anzeigen	Editor
Sprache und Anzahl Sprecher automatisch erkennen	Transkriptionsprozess
Video Transkription	Transkriptionsprozess
Dialekte (Schweiz) zu Hochdeutsch transkribieren	Transkriptionsprozess
Diarization abkoppeln: Identification Prozess mit Sprecher Modellen (Speaker Identification) und anschliessender Diarization zur Verfeinerung	Transkriptionsprozess
Transkription Memory Datenbank: Häufige, themenspezifische Wortlisten hochladen und nach dem transkribieren Worte mit tiefer Gewissheit versuchen zu ersetzen (DB mit allen gemachten Änderungen am Text)	Transkriptionsprozess
Den Weg einer Audio-Aufnahme auf dem Smartphone zur PC Applikation vereinfachen	Transkriptionsprozess
Transkript mit einem bestehenden vergleichen, um die Fehler zu erkennen und für Evaluationszwecke: Goldstandard Transkript hochladen, und dann werden in Realtime das Transkript mit dem Goldstandard verglichen (Statistik, WER, Hervorhebungen im Transkript selbst wo es Abweichungen gibt, etc.)	Evaluierung
Smartphone App (iOS, Android)	Plattform
Automatisches Speichern für jegliche Änderungen	Editor
Wörterbuch für falsch geschriebene Wörter	Editor
Grammatikregeln für Sätze	Editor
Transkript Resultate automatisch verbessern: - Automatische Interpunktion nach der Transkription	Transkriptionsprozess
Frontend als Webseite anstatt als Electron Desktop Applikation	Anwendung
Audio Player mit Visualisierung der Stimmen	Audioplayer
Die Key Bindings durch den User anpassbar machen (Ein Klick, Doppelklick, Tastatur, ...)	Anwendung
Alle Buttons sollten ein Tooltip mit der Funktion und der Tastenkombination (falls vorhanden) haben	Anwendung
Wortsuche mit Ctrl-F mit einzelnem Durchspringen oder «Highlight all»	Editor
User Management für persönliche Einstellungen, Kostenkontrolle	Anwendung
Hilfe, Demo, Erklärung für Nutzung der Anwendung	Anwendung
Ein Segment sollte als Ganzes in einem Textfeld bearbeitet werden können (nicht nur einzelne Wörter)	Editor
Eine Offline API für ASR sollte integriert werden (z.B. Mozilla DeepSpeech)	Transkriptionsprozess
Transkript Conventions umsetzen (Pausen, Sprecher, etc.)	Transkriptionsprozess
Aufgrund des «unbekannten Herausgebers» wird die Desktop Applikation auf manchen PCs als potenzieller Virus klassifiziert	Sicherheit
Wenn im Player reingeklickt wird, soll das Transkript zu dieser Stelle im Text springen (scrollen)	Editor

Das Verzeichnis zum Speichern sollte standardmässig dasselbe der Audio Datei sein	Anwendung
Speichern Unter sollte immer im vorherigen Verzeichnis starten	Anwendung
Der Player sollte die aktuelle Zeit als Tooltip darstellen	Anwendung
Rechtsklick auf einen Leerschlag zeigt zurzeit: "Wort einem Sprecher zuweisen", was nicht hilfreich ist. Besser wäre: "Segment hier trennen" und der Rest wird dem folgenden Sprecher zugewiesen	Editor
Unnötige Electron Menu-Einträge entfernen (Hilfe etc.)	Anwendung
Auswahl zwischen "Schweizer Server" und "Externem Dienst" anbieten. Beim externen Dienst werden die Daten ausserhalb der Schweiz bearbeitet, dafür ist es genauer. Eine Hilfeseite wäre nötig, um den Unterschied zu beschreiben	Plattform
Für Tester: Eine Schaltfläche mit Formular wo Feedback an die Entwickler gesendet werden kann	Anwendung
Für Tester: Die Nutzung ist auf 5h Transkribieren und bis zu einem Datum zu begrenzen. Danach kann der Tester nur noch Feedback absenden. Mittels eines Codes soll die Probezeit verlängert werden können	Anwendung
Der Abstand zwischen den Wörtern soll vergrössert werden, um gezieltes klicken zu vereinfachen	Editor
Es soll eine Option geben, wo gewählt werden kann ob während dem Bearbeiten eines Wortes: 1. Das Audio weiterläuft, 2. Das Audio pausiert wird, 3. Das Audio diesen Teil repetiert	Anwendung
Eine selektierte Phrase oder auch mehrere Wörter sollen gemeinsam bearbeitet werden können	Editor
Die Grösse des "Play Button" im Audioplayer soll hochskaliert werden. Es soll ebenfalls ein Tooltip angehängt werden, welches sagt, dass auch die Leertaste dazu verwendet werden kann	Audioplayer
Mit ESC sollte das Editieren immer abgebrochen werden können	Editor
Wiederholende Wiedergabe soll auch für mehrere selektierte Phrasen möglich sein	Anwendung
Idee: Anstatt ein Wort über Selektion und Kontextmenu einem anderen Sprecher zuzuweisen, könnte man auch Drag'n'Drop verwenden	Editor
Bug: Wenn Musik zu Beginn der Konversation spielt, wird der Anfang oft fehlerhaft oder unvollständig transkribiert	Transkriptionsprozess
Audio: Mp3 bei Google konvertieren, IBM wird unterstützt	Transkriptionsprozess
Der Transkriptionsprozess soll abgebrochen werden können	Transkriptionsprozess
Datenstruktur der Speaker Frames optimieren für Suche, Hervorheben, etc. (Binärbaum?)	Anwendung
Sortierung von parallelen Aussagen anhand der Sprecher-Namen oder deren ID	Anwendung
Refactoring: Backend Klassen abkapseln durch private Variabeln und private Methoden	Anwendung
Flag für Sprecher, die durch User eingefügt wurden (-> UnrecognizedSpeaker in backend nutzen oder refactoren)	Anwendung

8.3 Technische Dokumentation

8.3.1 Installation und Entwicklung

The Python Flask server for external API communication is located in `backend/`, while the main app is located in `frontend/`.

Installation

Setup Credentials

- Follow the steps listed here:
  - https://cloud.google.com/speech-to-text/docs/quickstart-client-libraries
  - https://cloud.ibm.com/docs/services/watson?topic=watson-creating-credentials
- Add the credential-files (google_cloud_credentials.json, ibm-credentials.env) to `backend/` folder

Backend Setup

1. Install Python 2 & 3
2. Install python virtualenv: `pip install –upgrade virtualenv`
3. Create and activate the virtual environment in the `backend` folder:

```
# *NIX:
cd backend
virtualenv –python python3 env
source venv/Scripts/activate # OSX: source env/bin/activate

# Windows:
cd backend
virtualenv –python "c:\python36\python.exe" env
.\env\Scripts\activate
```

4. Install the requirements:

```
# *NIX:
pip install -r requirements.txt

# Windows:
pip install pypiwin32
pip install -r requirements.txt
```

5. Copy `backend/example_config.ini` and rename it to `backend/config.ini`

Frontend Setup

Install the required dependencies:

```
cd frontend
npm install
```

Known Issues

If the error `grpc module not found` or similar occurs, reinstall the google cloud speech package: `pip install google-cloud-speech`

Run in Development Mode

1. Execute start script

```
cd backend

# *NIX:
bin/start

# Windows:
bin/start.sh
```

2. Start the Electron application:

```
cd frontend && npm run dev
```

Further Information

Translations

Localization JSON files are located in `frontend/static/locales/*.json`. To add a new language, follow the steps below:

1. Copy one of the existing locale files and rename it (e.g. `cn.json`)
2. Open `frontend/src/renderer/i18n.js`
3. Import the new locale file: `import cn from '../../static/locales/cn.json'`
4. Add the imported object to the messages variable:
5. `const messages = { ..., cn: cn }`

Deactivate Virtual Environment

```
deactivate
```

Package and Run Electron App

1. Package the frontend

```
cd frontend

# build windows executable electron app
npm run build:win32

# build osx electron app
npm run build:Darwin
```

2. Make sure the virtual environment is started in backend (see backend setup)
3. Start the backend flask server:

```
cd backend
python api.py
```

1. Open interscriber.exe in `frontend\build\interscriber-*`

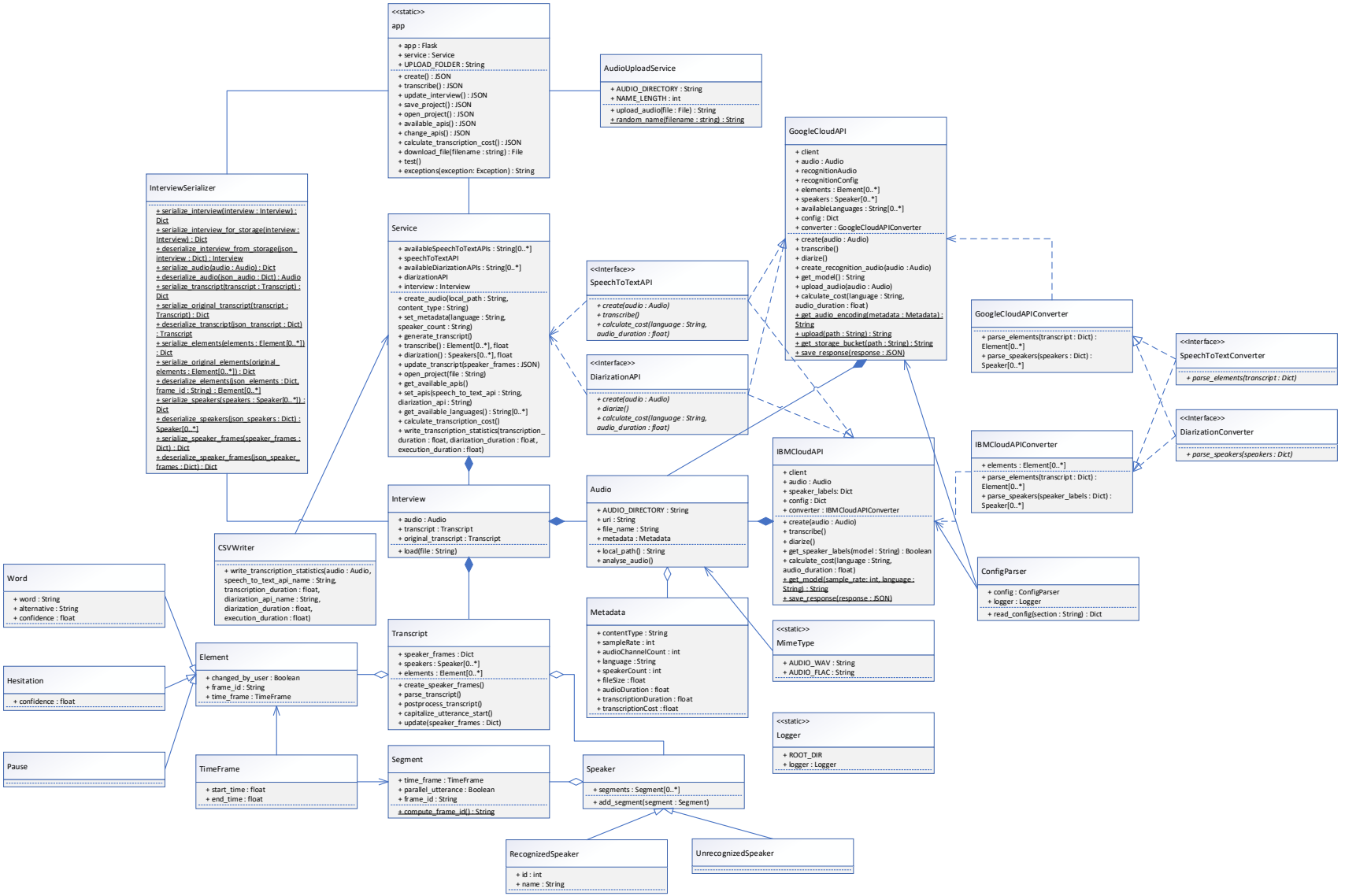
8.3.2 Übersicht der Backend Dateien

```
backend
├── Transcription-Statistics.csv    # Data collected about transcriptions
├── apis                            # API communication interfaces
│   ├── GoogleCloudAPI.py
│   └── IBMCloudAPI.py
├── app.py
├── backend.log
├── bin
│   ├── start                       # *NIX script to run backend
│   └── start.sh                    # DOS script to run backend
├── config.ini                      # Server config file
├── converters                      # Converter classes for API responses
│   ├── GoogleCloudAPIConverter.py
│   └── IBMCloudAPIConverter.py
├── example_config.ini              # Example config to be copied
├── google_cloud_credentials.json   # Google Cloud credentials file
├── ibm-credentials.env             # IBM Watson credentials file
├── interfaces                      # Main interfaces
│   ├── DiarizationAPI.py
│   ├── DiarizationConverter.py
│   ├── SpeechToTextAPI.py
│   └── SpeechToTextConverter.py
├── models                          # Model classes for various concepts
│   ├── Audio.py                    # Audio file
│   ├── Interview.py                # Interview project
│   ├── Metadata.py                 # Metadata belonging to an audio
│   ├── Segment.py                  # Utterance of a speaker
│   ├── TimeFrame.py                # Time frame of word or segment
│   ├── Transcript.py               # Transcript belonging to interview
│   ├── elements
│   │   ├── Element.py
│   │   ├── Hesitation.py
│   │   ├── Pause.py
│   │   └── Word.py
│   └── speakers
│       ├── RecognizedSpeaker.py
│       ├── Speaker.py
│       └── UnrecognizedSpeaker.py
├── requirements.txt                # Pip dependency file
├── services
│   ├── AudioUploadService.py
│   └── Service.py
├── tests                           # Test scripts for different parts
│   ├── ...
├── uploads                         # Folder where audio gets stored
│   ├── 1.wav
│   ├── 2.flac
│   └── ...
└── utils                           # Utility, parser & serializer classes
    ├── CSVWriter.py
    ├── ConfigParser.py
    ├── InterviewSerializer.py
    ├── Logger.py
    └── MIMEType.py
```

8.3.3 Übersicht der Frontend Dateien

```
frontend
├── build
│   └── icons                       # Icons for packaged application
│   │   └── ...
│   └── interscriber-*              # Packaged versions (win32, etc.)
├── config.json                     # Configuration file for frontend
├── package-lock.json
├── package.json                    # App info, dependencies & scripts
├── src
│   ├── index.ejs                   # Entry point compiled to index.html
│   ├── main                        # Electron window setup is done here
│   │   ├── index.dev.js
│   │   └── index.js
│   └── renderer                    # Vue application itself
│       ├── App.vue                 # Vue base component
│       ├── assets                  # Assets folder
│       │   ├── logo_w.png
│       │   └── zhaw_logo_w.png
│       ├── components              # Vue components
│       │   ├── Alert.vue
│       │   ├── AssignSpeaker.vue
│       │   ├── AudioSettings.vue
│       │   ├── AudioStatistics.vue
│       │   ├── ChangeLanguage.vue
│       │   ├── DeleteUtterance.vue
│       │   ├── Editor.vue          # Editor base component
│       │   ├── EditorSettings.vue
│       │   ├── ExportMixin.vue
│       │   ├── GeneralSettings.vue
│       │   ├── Header.vue
│       │   ├── HomeButton.vue
│       │   ├── LandingPage.vue     # Landing page / central component
│       │   ├── LoadAudio.vue
│       │   ├── LoadInterview.vue
│       │   ├── NewUtterance.vue
│       │   ├── ProgressBar.vue
│       │   ├── SaveInterview.vue
│       │   ├── Snackbar.vue
│       │   ├── TimeMixin.vue
│       │   └── TranscriptMixin.vue
│       ├── i18n.js                 # i18n internationalization setup
│       ├── main.js                 # Vue setup (use Vue plugins here)
│       ├── menu.js                 # Electron menu configuration
│       ├── router                  # Vue routing files (unused)
│       │   └── index.js
│       └── store                   # Vuex storage files
│           ├── index.js
│           └── ...
└── static                          # Static assets
    └── locales
        ├── de.json                 # German string translations
        └── en.json                 # English string translations
```

8.3.4 Vollständiges Klassendiagramm



8.3.5 Backend HTTP Schnittstelle

URL	Beschreibung
/create	Route für das Erstellen eines Audios
/transcribe	Route für das Transkribieren eines Audios
/update	Route für das Aktualisieren einer Transkription
/interview	Route für das Speichern eines Projekts
/interview/open	Route für das Laden eines Projekts
/apis	Route um die verfügbaren STT und Diarization Dienste, sowie die Sprachen zu holen
/change_apis	Route für Wechseln der aktuellen STT und Diarization API
/transcription_cost	Route für Berechnen des aktuellen Transkriptions-Preises
/files/<path:filename>	Route zu hochgeladenen Audio-Quellen

POST 200 - /create, /transcription_cost


```
{
    "audio": {
        "file_name": "jzdqqnvykgodkfu.wav",
        "metadata": {
            "audio_channel_count": 2,
            "audio_duration": 48.644,
            "content_type": "audio/wav",
            "file_size": 8.18,
            "language": "en-US",
            "sample_rate": 44100,
            "speaker_count": 2,
            "transcription_cost": 0.02,
            "transcription_duration": 0
        },
        "uri": "gs://.../jzdqqnvykgodkfu.wav"
    }
}
```


POST 200 - /transcribe, /update, /interview/open


```
{
    "interview": {
        "audio": {
            "file_name": "tmbnhdrcaahwivg.wav",
            "metadata": {
                "audio_channel_count": 2,
                "audio_duration": 48.644,
                "content_type": "audio/wav",
                "file_size": 8.18,
                "language": "en-US",
                "sample_rate": 44100,
                "speaker_count": 2,
                "transcription_cost": 0.02,
                "transcription_duration": 0
            },
            "uri": "gs://.../tmbnhdrcaahwivg.wav"
        },
```

```
        "transcript": {
             "speaker_frames": [
                 {
                     "elements": [
                         {
                             "alternative": "",
                             "changed_by_user": false,
                             "confidence": 91.28,
                             "frame_id": "YNbVWFkM2D",
                             "time_frame": {
                                 "end_time": 2,
                                 "start_time": 0
                             },
                             "word": "Excuse"
                         }
                     ],
                     "frame_id": "YNbVWFkM2D",
                     "parallel_utterance": false,
                     "speaker_id": 1,
                     "speaker_name": "",
                     "time_frame": {
                         "end_time": 9.2,
                         "start_time": 0
                     }
                 }
             ]
        }
    }
}
```


GET 200 - /interview


```
{
    "interview": {
        "audio": {
            "file_name": "tmbnhdrcaahwivg.wav",
            "metadata": {
                "audio_channel_count": 2,
                "audio_duration": 48.644,
                "content_type": "audio/wav",
                "file_size": 8.18,
                "language": "en-US",
                "sample_rate": 44100,
                "speaker_count": 2,
                "transcription_cost": 0.02,
                "transcription_duration": 0
            },
            "uri": "gs://.../tmbnhdrcaahwivg.wav"
        },
        "original_transcript": {
            "elements": [
                {
                    "alternative": "",
                    "confidence": 91.28,
                    "frame_id": "YNbVWFkM2D",
                    "time_frame": {
                        "end_time": 2,
                        "start_time": 0
                    },
                    "word": "Excuse"
                }
            ],
            "speakers": [
```

```
                {
                    "id": 1,
                    "name": "",
                    "segments": [
                        {
                            "frame_id": "YNbVWFkM2D",
                            "parallel_utterance": false,
                            "time_frame": {
                                "end_time": 9.2,
                                "start_time": 0
                            }
                        }
                    ]
                }
            ]
        },
        "transcript": {
            "elements": [
                {
                    "alternative": "",
                    "changed_by_user": true,
                    "confidence": 100,
                    "frame_id": "YNbVWFkM2D",
                    "time_frame": {
                        "end_time": 2,
                        "start_time": 0
                    },
                    "word": "Excuse"
                }
            ],
            "speakers": [
                {
                    "id": 1,
                    "name": "",
                    "segments": [
                        {
                            "frame_id": "YNbVWFkM2D",
                            "parallel_utterance": false,
                            "time_frame": {
                                "end_time": 9.2,
                                "start_time": 0
                            }
                        }
                    ]
                }
            ]
        }
    }
}
```


GET 200 - /apis


```
{
    "availableDiarizationAPIs": [],
    "availableLanguages": [
        {
            "model": "en-US",
            "name": "en-US"
        }
    ],
    "availableSpeechToTextAPIs": ["GoogleCloudAPI", "IBMCloudAPI"]
}
```

POST 200 - /change_apis

```
{
    "availableLanguages": [
        {
            "model": "en-US",
            "name": "en-US"
        }
    ]
}
```

8.4 USB Stick

Dem Bericht wird ein USB-Stick angehängt. Folgende Dateien und Ordner sind darin enthalten:

- **Beispiele:** Beispielhafte Audio Dateien und transkribierte Interview-Projekt-Dateien (.intts) von deutschen und englischen Konversationen.
- **BA19_ciel_01:** Vollständiger Quellcode der Applikation «InterScriber»
- Digitaler Bericht als .pdf Datei