System Requirements Specification

for

Kaldi ASR Team

Version 5.0

Prepared by

Tabitha O'Malley, Milan Haruyama, David Serfaty,

Tahmina Tisha, Adam Gallub

CS 490 RTube Kaldi ASR Team Spring 2024

03/03/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Tabitha, Milan, Tisha, David, Adam, Max	09/29/23	Starting the document	V1.0
Tabitha	10/25/23	Formatting Revision History Editing/Formatting Appendix Writing Section: 1.5	V2.1
Tabitha	10/26/23	Writing Sections: 1.2, 2.2, 2.5, 3.1	V2.2
Tabitha	10/27/23	Writing Requirements: 3.1	V2.3
Milan	10/27/23	Writing and Editing Requirements: 3.1	V2.4
David	10/27/23	Writing and Editing Requirements: 3.1	V2.5
Tabitha	10/29/23	Writing Section: 2.3, 2.4, 2.5	V2.6
David	10/29/23	Writing Section:2.3, 2.4, 2.5	V2.7
Tabitha	10/30/23	Writing Section: 4, 4.1, 4.2, 4.3, 4.4, 5.1, 5.2, 3	V2.8
Milan	10/30/23	Editing all Sections Writing Section: 5.1, 5.2, 2.1	V2.9
David	10/30/23	Writing Section: 2.1, 5.3	V2.10
Tabitha	10/31/23	Update Model Editing Sections	V2.11
Milan	10/31/23	Editing all sections	V2.12
David	10/31/23	Editing, 2.1, 3, Appendix A, 4.1, 4.2, 4.1.3, 4.2.3, 4.3, 4.3.3, 2.2 Writing Section: 5.2	V2.13
Adam	11/05/23	Writing Section: 2.4 and 2.6	V3.1
Milan	11/05/23	Editing all sections	V3.2
Tabitha	11/05/23	Rewrite: 2.3 Add to: 2.2	V3.3
David	11/05/23	Class Model: 2.3 Add to: 2.2	V3.4

Tisha	11/05/2023	Editing: 2.4, 2.5	V3.5
Tabitha	11/07/23	Adding/Editing: 3	V3.6
Milan	11/07/2023	Adding/Editing: 3	V3.7
David	11/07/2023	Adding/Edition: 3	V3.8
Adam	11/07/2023	Writing/Editing 2.6	V3.9
Tabitha	11/11/2023	Editing: 3	V3.10
Milan	11/11/2023	Editing: 3	V3.11
Adam	11/11/2023	Editing/Writing: 4.1.1	V3.12
Tabitha	11/14/2023	Update Requirements	V3.13
Adam	11/15/2023	Editing/Writing: 4	V3.14
Tisha	11/15/2023	Editing section: 5	V3.15
Tabitha	11/15/2023	Editing: 1.5, 2.3, 4.1, 5.1	V3.16
Tabitha	11/16/2023	Review/Edition/Commenting Section: 4 Update Class Diagram Update/Review Section: 2.3	V3.17
Tisha	11/18/2023	Rewriting section 5.2	V3.18
Tisha	11/18/2023	Edited section 5.2	V3.19
Milan	11/18/2023	Editing all sections	V3.20
David	11/19/2023	Adding/Editing/Rewriting: 4, 5.1, 5.2 Editing 2.2 Appendix B	V3.21
Tabitha	11/19/2023	Adding/Editing/Rewriting: 4, 5.1, 5.2 Appendix A, B	V3.22
Milan	11/19/2023	Editing all sections	V3.23
Milan	11/20/2023	Reviewing/editing all sections	V3.24
Milan	01/29/2024	Reviewing all sections	V4.0
Milan	01/30/2024	Finished Section 1.4	V4.1
Tabitha	02/04/2024	Updated Appendix	V4.2

		Read through Sections 4 Update Requirements	
Milan	02/05/2024	Reviewing and editing all sections	V4.3
Tabitha	02/26/2024	Update Requirements Section 4	V5.0
Tabitha	02/27/2024	Section 2.3	V5.1
Milan	03/03/2024	Reviewing and editing all sections	V5.2
			V6.0

1. Introduction

1.1 Purpose

The RTube web application shall provide an interface for users to track the flight paths of aircrafts, and to listen to live Air Traffic Control (ATC) tower radio transmissions with the option to transcribe speech to text in real time. The live speech-to-text transcription is performed using an acoustic automatic speech recognition (ASR) model developed using the Kaldi ASR toolkit, which is the focus of this document.

1.2 Document Conventions

Primary Body	**Times New Roman, 11 point**
Section Headers	**Times New Roman, 18 point, bold**
Subsection Header	**Times New Roman, 14 point, bold**
In-Progress	**Cyan highlight**
Pending Review/Update	**Yellow highlight**
Complete Overhaul	**Red highlight**
Completed	**Green highlight**
Important Terms	**Bold text within document body**

1.3 Intended Audience and Reading Suggestions

The intended audience for the RTube web application is aircraft pilots and ATC operators. One of the more specific applications for RTube shall be for Embry-Riddle Aeronautical University (ERAU) instructors and student pilots to evaluate communications between the student and ATC operators.

In order to improve understanding of this document and the RTube project itself, it is highly recommended to read documentation related to phonetics, aviation phraseology, ASR, NLP, and the Kaldi ASR Toolkit.

1.4 Product Scope

Learning to communicate with Air Traffic Control (ATC) is a daunting task for many new aircraft pilots. Aviation phraseology is often the biggest source of miscommunication (despite being designed to mitigate it) due to the highly intricate vernacular. The idiosyncrasies present within aviation phraseology require hundreds of hours of training for student pilots to learn. While there exist a few resources (e.g., the LiveATC website) for student pilots to study aviation phraseology, these resources do little to accommodate the major learning curve due to a lack of transcriptions of spoken speech for student pilots to read.

As such, the RTube web application shall bridge the learning gap faced by many student pilots by providing the ability to transcribe live ATC transmissions into text in real-time using an acoustic ASR model, as well as providing a live interface for users to track the flight paths of aircraft. With the ability to transcribe speech to text in real-time, student pilots can dramatically reduce the amount of training required to understand spoken aviation phraseology. In addition, the live interface helps students better understand different contexts in which specific phrases are used.

Currently, the RTube web application utilizes an end-to-end ASR model developed using the NVIDIA NeMo Toolkit. The Kaldi ASR Team aims to replace the end-to-end ASR model with an acoustic model developed using the Kaldi ASR Toolkit. The main advantage of replacing the end-to-end ASR model with an acoustic one is the faster training speed of the latter due to requiring less training data than end-to-end models. However, the improved training efficiency comes at the cost of a significantly more difficult development life cycle, as the Kaldi ASR Toolkit is highly complex and poorly documented for people unfamiliar with ASR model development.

1.5 References

Fagan, Jonathan. “What Is a Good Accuracy Level for Transcription?” *University Transcription Services*, 3 June 2020, www.universitytranscriptions.co.uk/what-is-a-good-accuracy-level-for-transcription/. Accessed 26 Oct. 2023

Kaldi ASR Team Product Vision Statement. Kaldi ASR Team. 19 September 2023. Kaldi ASR Team Drive.

Kaldi ASR Team Software Design Document. Kaldi ASR Team. 19 November 2023. Kaldi ASR Team Drive.

Ravihara, Ransaka. “Gaussian Mixture Model Clearly Explained.” *Medium*, Towards Data Science, 11 Jan. 2023, www.towardsdatascience.com/gaussian-mixture-model-clearly-explained-115010f7d4cf/.

“About Pandas.” *Pandas*, www.pandas.pydata.org/about/. Accessed 26 Oct. 2023.

“LiveATC FAQ.” *LiveATC.Net - Listen to Live Air Traffic Control over the Internet!*, www.liveatc.net/faq/. Accessed 26 Oct. 2023.

“A Python Library to Read/WRITE EXCEL 2010 Xlsx/XLSM Files¶.” *Openpyxl*, www.openpyxl.readthedocs.io/en/stable/. Accessed 26 Oct. 2023.

“What Is Kaldi?” *Kaldi*, www.kaldi-asr.org/doc/about.html/. Accessed 26 Oct. 2023.

“What Is Speech Recognition?” *IBM*, www.ibm.com/topics/speech-recognition/. Accessed 26 Oct. 2023.

Chester F. Carlson Center for Imaging Science | College of Science | RIT, www.cis.rit.edu/class/simg716/Gauss_History_FFT.pdf/. Accessed 1 Nov. 2023.

“About FFmpeg” *FFmpeg,* www.ffmpeg.org/about.html/. Accessed 19 Nov. 2023.

Linguistic Data Consortium. (1994). LDC94S14A: ATIS-3 Training Text, Version 1.1. University of Pennsylvania. https://catalog.ldc.upenn.edu/LDC94S14A. Accessed 25 February 2023

IDIAP Research Institute. (2024). Atco2 Corpus. GitHub. https://github.com/idiap/atco2-corpus. Accessed 25 February 2023

2. Overall Description

2.1 Product Perspective

The Kaldi ASR Toolkit is an open-source toolkit created in 2009 by Johns Hopkins University that is designed to create ASR models. The Kaldi ASR Team shall utilize the toolkit to develop its own acoustic ASR models to transcribe live or prerecorded ATC transmissions. A set of models shall be created, then trained with a 30-hour ATC dataset. By comparing the performance between each trained model (e.g., transcription accuracy, runtime, et cetera), the team shall assess which model is most suitable for creating an online speech recognition tool to transcribe live ATC transmissions in the future. The finished product for this semester shall be several trained models that transcribe speech signal files into text.

2.2 Product Functions

Func. 1	Receive WAV files for processing
Func. 2	Transcribe audio into text
Func. 3	Output a text file with the transcribed words from the audio file.

2.3 User Classes and Characteristics

This section shall provide an overview of each class of the Kaldi ASR Toolkit. The general purpose, any methods used, and all inputs and outputs of each class shall be discussed.

Dithering		
This class shall add Gaussian noise to the audio so that it never equals zero.		
Variables		
Inputs	**Outputs**	**Interal**
Audio wavFile	ditheredAudio	gaussianNoise
Methods		
addNoise(wavFile, gaussianNoise)		

Pre-emphasis filtering		
Ths class shall filter the audio to ensure a frequency range between 8 to 16 kHz.		
Variables		
Inputs	**Outputs**	**Interal**
ditheredAudio	filteredAudio	a
		n
Methods		
filterFrequencies(ditheredAudio, a, n)		

Sliding Intervals		
This class shall frame the audio into 25- millisecond intervals.		
Variables		
Inputs	**Outputs**	**Interal**
filteredAudio	framedAudio	n
Methods		
framingAudio(filteredAudio, n)		

Hamming Window		
The class shall window the data to prevent spectral leakage.		
Variables		
Inputs	**Outputs**	**Interal**
filteredAudio	framedAudio	n
		N
Methods		
WindowingAudio(framedAudio, N, n)		

Fast Fourier Transform (FFT)		
This class shall convert the windowed data from the time domain to the frequency domain.		
Variables		
Inputs	**Outputs**	**Interal**
WindowAudio	frequencyDomainAudio	N
		K
		k
		j
Methods		
discreteForierTransform(WindowAudio, N, K, j, k)		

Limit for Energy Frequency		
This class shall find the maximum energy frequency.		
Variables		
Inputs	**Outputs**	**Interal**
	maxEnergyFrequency	kLower
		kUpper
Methods		
maxEnergyCalc(kLower, kUpper)		

Domain Filter		
This class shall filter the data so that it does not exceed the maximum energy		
Variables		
Inputs	**Outputs**	**Interal**
frequencyDomainAudio	filterDomainAudio	k
maxEnergyFrequency		K
Methods		
filterDomainfrequency(maxEnergyFrequency, k, K, frequencyDomainAudio)		

MelVector		
This class shall convert the data into a mel vector.		
Variables		
Inputs	**Outputs**	**Interal**
filterDomainAudio	melVector	
Methods		
vectorData(filterDomainAudio)		

Fbank		
This class shall calculate the logarithm of the mel vector.		
Variables		
Inputs	**Outputs**	**Interal**
melVector	logMelVector	
Methods		
Logarithm(melVector)		

HTK		
This class shall convert frequencies to melody numbers.		
Variables		
Inputs	**Outputs**	**Interal**
frequency	melodyNumber	
Methods		
calculateMelodyNumber(frequency)		

Discrete Cosine Transform (DCT)		
This class shall mirror and smooth the data.		
Variables		
Inputs	**Outputs**	**Interal**
melNumber	dctVector	k
logMelVector		M
Methods		
discreteCosineTransform(melodyNumber, logMelVector, k, M)		

Mel-Frequency Cepstrum (MFC)		
This class shall convert frequency-domain frames into a 39-dimensional MFCC feature vector represented as a 39-dimensional array.		
Variables		
Inputs	**Outputs**	**Interal**
dctVector	MFCC[]... 37…[] featureVector	
Methods		
convertToMFCC(dctVector)		

<table>
<tr><th colspan="3">Gaussian Mixture Model (GMM)</th></tr>
<tr><td colspan="3">This class shall calculate the expectation maximization of the MFCC feature vector in reference to phoneme Gaussian distribution.</td></tr>
<tr><th colspan="3">Variables</th></tr>
<tr><th>Inputs</th><th>Outputs</th><th>Interal</th></tr>
<tr><td>initalProb</td><td rowspan="3">gamma</td><td rowspan="3"></td></tr>
<tr><td>mean</td></tr>
<tr><td>variance</td></tr>
<tr><th colspan="3">Methods</th></tr>
<tr><td colspan="3">recalculateExepectation(gamma)</td></tr>
<tr><td colspan="3">maximixizeValues(mean, variance, initialProb)</td></tr>
</table>

<table>
<tr><th colspan="3">Hidden Markov Model (HMM)</th></tr>
<tr><td colspan="3">Figures out the hidden state sequence.</td></tr>
<tr><th colspan="3">Variables</th></tr>
<tr><th>Inputs</th><th>Outputs</th><th>Interal</th></tr>
<tr><td>setOfStates</td><td rowspan="3">hiddenStateSequence</td><td>transitionMatrix</td></tr>
<tr><td>setOfObservations</td><td rowspan="2">emissionMatrix</td></tr>
<tr><td>setOfInitialProbabilities</td></tr>
<tr><th colspan="3">Methods</th></tr>
<tr><td colspan="3">calculateSequenceOfEvents(setOfStates, setOfObservations, setOfInitialProbabilities, transitionMatrix, emissionMatrix)</td></tr>
</table>

Preparation		
This class shall prepare the data for decoding		
Variables		
Inputs	**Outputs**	**Interal**
MFCC	hiddenStateSequence	
GMM		
HMM		
Methods		
prepareAudioForDecoding()		

Viterbi_Decoder		
This class shall find the most probable sequence of states.		
Variables		
Inputs	**Outputs**	**Interal**
stateSpace	mostProbableSequenceOfState	transitionMatrix
initialProbabilities		emissionMatrix
- observationSpace		
Methods		
calculateSquenceOfEvents(stateSpace, observerationSpace, initailProbablities, transitionMatrix, emissionMatrix)		

Decode		
This class shall prepare a sequence for output.		
Variables		
Inputs	**Outputs**	**Interal**
Viterbi_Decoder	mostProbableSequenceOfState	
Methods		
prepareSequenceForOutput()		

<table>
<tr><th colspan="3">WFST</th></tr>
<tr><td colspan="3">This class shall convert the sequence of states into sentences.</td></tr>
<tr><th colspan="3">Variables</th></tr>
<tr><th>Inputs</th><th>Outputs</th><th>Interal</th></tr>
<tr><td rowspan="4">mostProbableSequenceOfState</td><td rowspan="4">sentance</td><td>triphone</td></tr>
<tr><td>monophone</td></tr>
<tr><td>word</td></tr>
<tr><td>Monophones</td></tr>
<tr><th colspan="3">Methods</th></tr>
<tr><td colspan="3">gettriphone(mostProbableSequenceOfState)</td></tr>
<tr><td colspan="3">convertfromtritomono(triphone)</td></tr>
<tr><td colspan="3">createarrayof monophones(monophones)</td></tr>
<tr><td colspan="3">convertfrommonotoword(Monophones[], File lexicon)</td></tr>
<tr><td colspan="3">compilesentence(word)</td></tr>
</table>

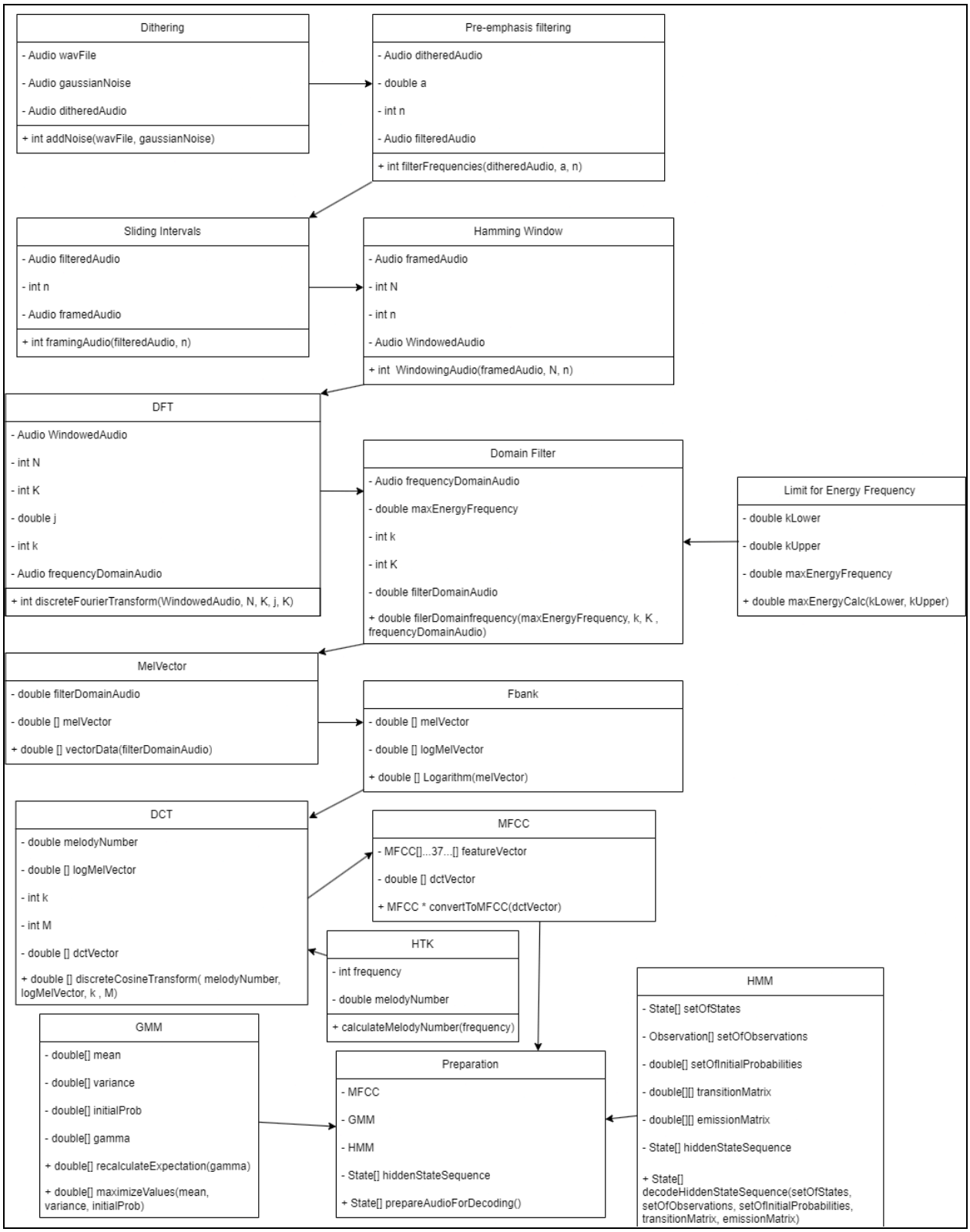

Figure 01: ASR Model Preparation Stage Class Diagram V5.2.2

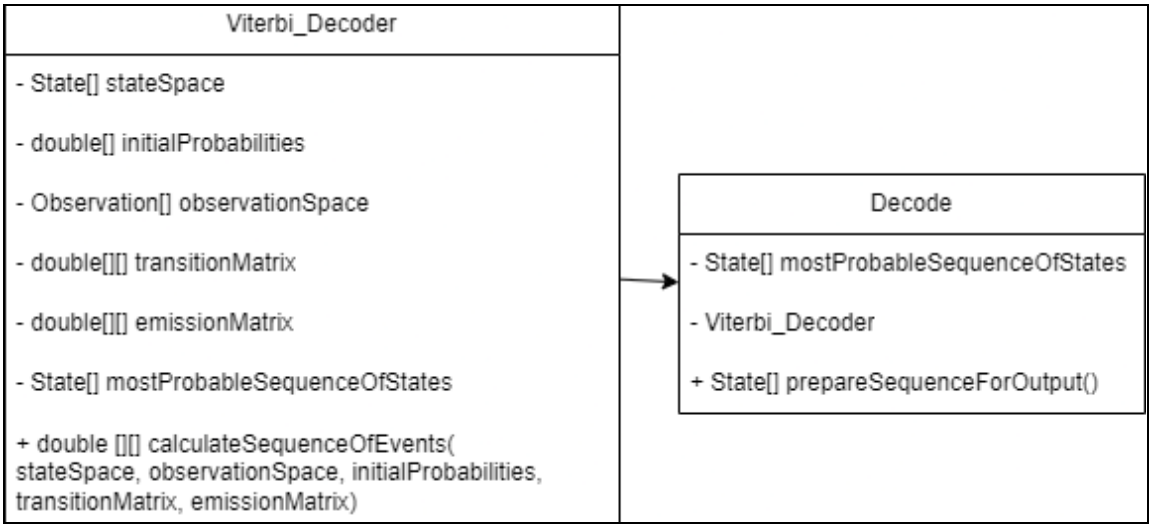

Figure 02: ASR Model Decode Stage Class Diagram V5.3.2

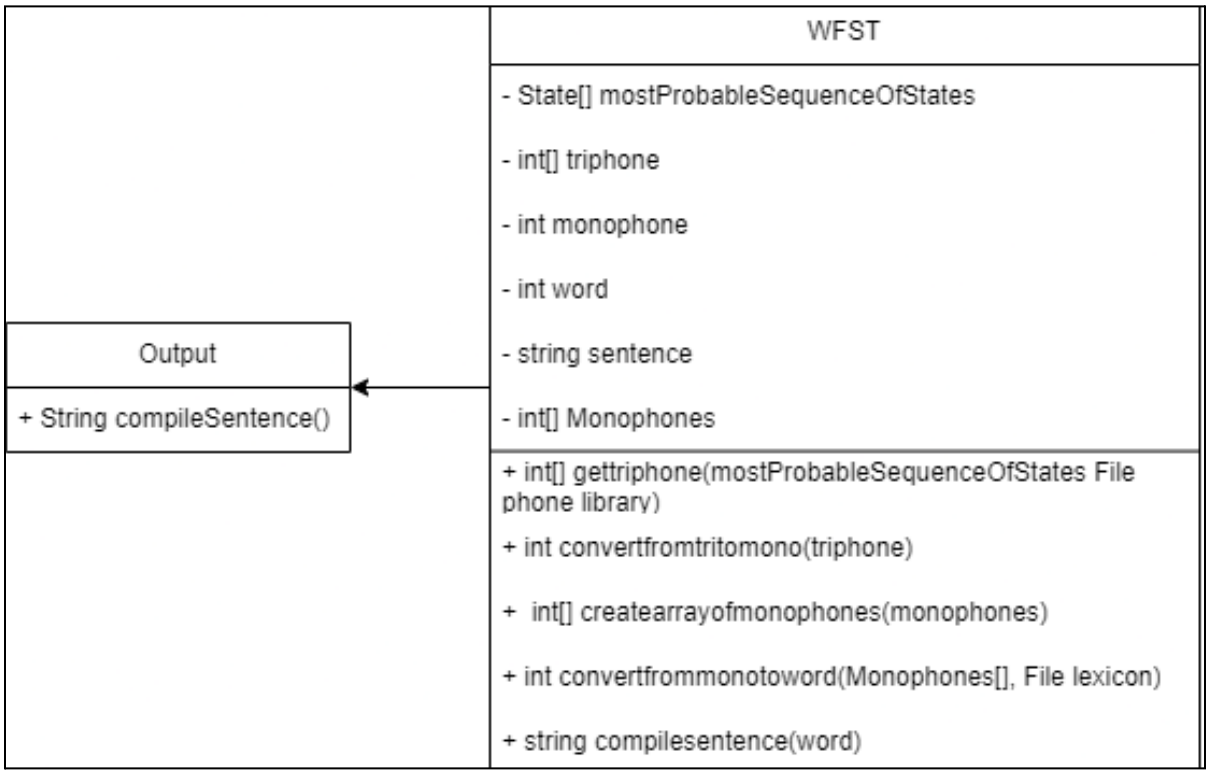

Figure 03: ASR Model Output Stage Class Diagram V5.4.1

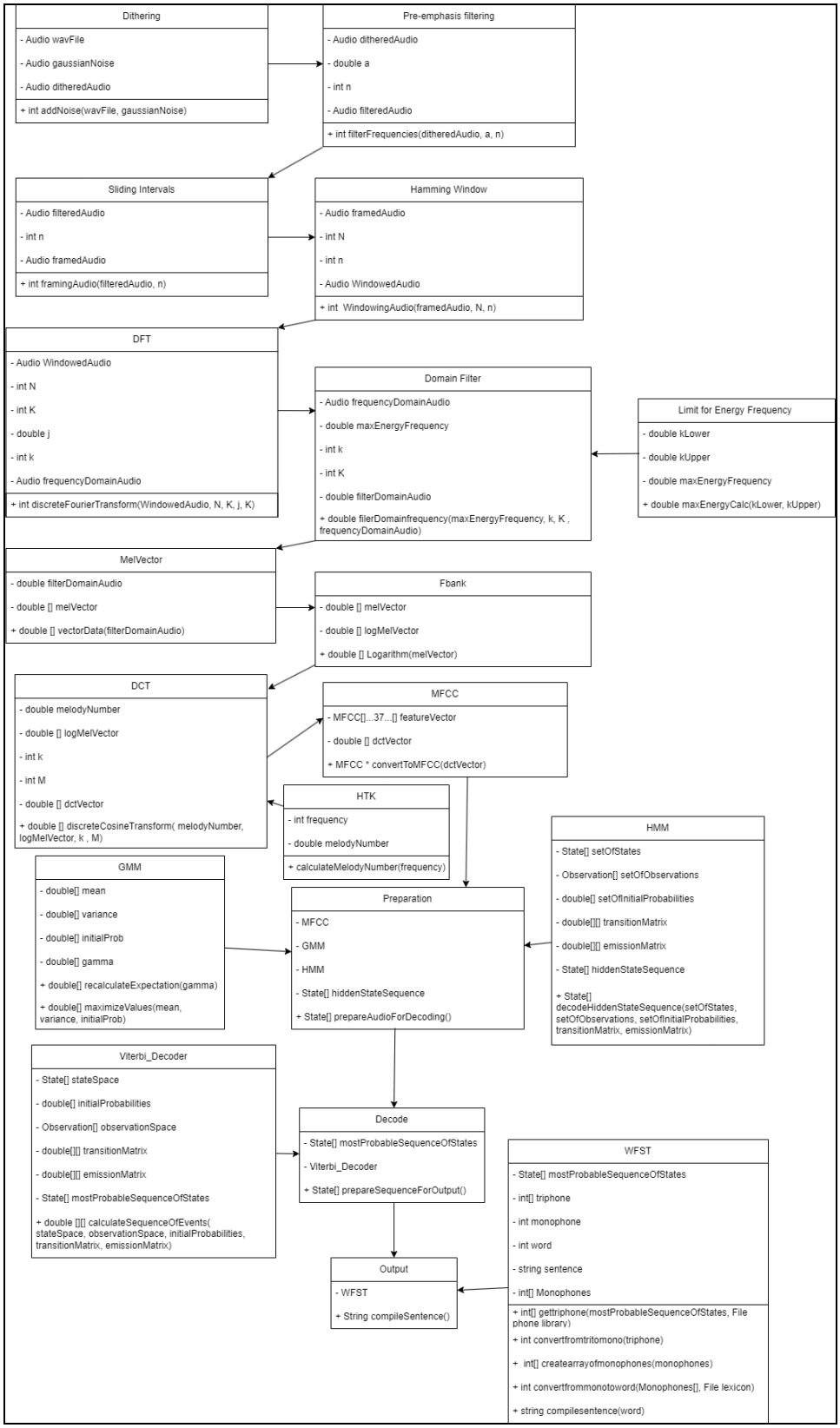

Figure 04: ASR Model Class Diagram V5.1.3

2.4 Operating Environment

The Kaldi ASR Toolkit and the acoustic ASR model developed by the Kaldi ASR Team shall run on a command-line interface via the Windows Subsystem for Linux (WSL). The specific operating system needed to install and run the interface is Ubuntu, a free and open-source Linux distribution. The Kaldi ASR Toolkit also uses code-level integration with finite state transducers (FSTs), and compiles using the OpenFst Toolkit.

2.5 Design and Implementation Constraints

The design constraints are as follows: any computer used to develop and train the acoustic ASR model must have at least 12 GB of video memory (VRAM) for the model to run within an hour; and must have a minimum storage size of 12.5 GB to install the Kaldi ASR Toolkit.

The implementation constraints are as follows: all inputs shall be WAV files; all audio shall be in General American English; and no punctuation or grammatical marks shall be transcribed by the ASR model.

2.6 Assumptions and Dependencies

The development of the acoustic ASR model to be used in the RTube web application is constrained by several factors. These include but are not limited to: the assumption that all speech from aircraft pilots and ATC is both clear and direct (i.e., speakers are not soft-spoken, speakers do not mumble their words, et cetera); the assumption that all audio input shall be in General American English; the assumption that the audio to be transcribed contains low radio interference and background noise (e.g., interfering signals, loud engine noise, et cetera); and the assumption that 30 hours of training data is sufficient to produce a transcription accuracy of at least 80%.

The development of the aforementioned model is also dependent on the developers having computers with enough video memory and storage to both store and run the Kaldi ASR toolkit to develop and train the model.

3. Software Interface Requirements

Data Preparation			
Audio file		Req. 1.1.1	The system shall only accept WAV files as input.
		Req. 1.1.2	The system shall convert other file types into WAV files using the FFMPEG Linux utility.
Mel-Frequency Cepstrum (MFC) & Mel-Frequency Cepstral Coefficients (MFCCs)	Pre-Emphasis	Req. 2.1.1	The system shall dither the signals.
		Req. 2.1.2	The system shall never let the signal be zero.
		Req. 2.2.1	The system shall filter the signal.
		Req. 2.2.2	The system shall use the formula x[n] - ax[n-1] to filter the data.
		Req. 2.2.3	The system shall define n as the sample.
		Req. 2.2.4	The system shall define x[n] as the input signal at n.
		Req. 2.2.5	The system shall define a as a value between 0.95 and 0.97.
	Signal Framing	Req. 3.1.1	The system shall partition the audio into overlapping 25-millisecond frames.
		Req. 3.1.2	The system shall obtain each frame in 10-millisecond sliding intervals starting from time equals 0.
		Req. 3.1.3	The system shall use the formula [10n, 10n + 25] to obtain the closed time intervals of each frame (e.g., [0, 25], [10, 35], [20, 45], …).
		Req. 3.1.4	The system shall define the value n as any non-negative integer. (i.e., $\{n \in \mathbb{Z} : n \geq 0\}$)
	Windowing	Req. 4.1.1	The system shall window the audio.
		Req. 4.1.2	The system shall use the formula $w[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1})$ to window the audio.
		Req. 4.1.3	The system shall define the variable N as the length of the window.
		Req. 4.1.4	The system shall define the variable n as N-1.

Data Preparation			
Mel-Frequency Cepstrum (MFC) & Mel-Frequency Cepstral Coefficients (MFCCs)	Fast Fourier Transform (FFT)	Req. 5.1.1	The system shall use a FFT to convert the audio data within the frames from the time domain to the frequency domain.
		Req. 5.1.2	The system shall use formula $X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{K}kn}$ to convert the audio from the time domain to the frequency domain.
		Req. 5.1.3	The system shall define the function x[n] as the windowed speech signal.
		Req. 5.1.4	The system shall define N is the length of the window.
		Req. 5.1.5	The system shall define the variable $K \geq N$.
		Req. 5.1.6	The system shall define the variable $j = \sqrt{-1}$.
		Req. 5.1.7	The system shall define the variable $k = \frac{\omega_k K}{2\pi}$.
	Mel Filter Bank	Req. 6.1.1	The system shall convert the results of the FFT into mel numbers.
		Req. 6.1.2	The system shall use the formula $m = 2595 \cdot log_{10}(1 + \frac{f}{700})$ to obtain the mel numbers.
	Limit Energy Frequency	Req. 7.1.1	The system shall filter the energy of the frequency.
		Req. 7.1.2	The system shall use the formula $a_m = \sum_{k=0}^{K/2+1} F_m[k]\|X[k]\|^2$ to limit the energy frequency of the audio file.
	Mel Vector	Req. 8.1.1	The system shall vectorize the data.
	Log of Vector	Req. 9.1.1	The system shall take the logarithm of the vector.

Data Preparation			
Mel-Frequency Cepstrum (MFC) & Mel-Frequency Cepstral Coefficients (MFCCs)	Discrete Cosine Transform	Req. 10.1.1	The system shall symmetrize the sequence with respect to the origin.
		Req. 10.1.2	The system shall use the formula $c[k] = \sum_{m=0}^{M-1} v[m] cos\,(\pi\,(m + 0.5)\,k/M)$ to symmetrize the sequence.
		Req. 10.1.3	The system shall define v[m] as the logarithm of the vector.
		Req. 10.1.4	The system shall define m as the melody number.
		Req. 10.1.5	The system shall define M as the number of dimensions from the vector.
		Req. 10.1.6	The system shall define k as $M - 1$.
	Mel Frequency Ceprtrum	Req. 11.1.1	The system shall reduce the dimensions of the vector from the DCT using to 12.
		Req. 11.1.2	The system shall set the thirteenth MFCC to the short time energy of the speech signal.
		Req. 11.1.3	The system shall use this formula to perform the derivative of the MFCC's, $c_{\Delta}[k,l] = \frac{\sum_{n=1}^{N} n(c[k,l+m] - c[k,l-m])}{2\sum_{n=1}^{N} n^2}$.
		Req. 11.1.4	The system shall derive the first-order derivative of the 13 MFCCs.
		Req. 11.1.5	The system shall derive the second derivative of the 13 MFCCs.
		Req. 11.1.6	The system shall produce a 39-dimensional MFC represented as an MFCC feature vector.
		Req. 11.1.7	The system shall use the first set of thirteen dimensions (i.e., [0, 12]) to represent the thirteen MFCCs.
		Req. 11.1.8	The system shall use the second set of thirteen dimensions (i.e., [13, 25]) to represent the first-order derivatives of the MFCCs.
		Req. 11.1.9	The system shall use the third set of thirteen dimensions (i.e., [26, 38]) to represent the second-order derivatives of the MFCCs.

Data Preparation			
Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM)	GMM	Req. 12.1.1	The system shall initialize three components: the mean of the component μk, the variance of the component Σk, the initial probability of the component πk.
		Req. 12.1.2	The systems shall compute the maximization of expectation of the MFCC feature vector matching up with a phoneme distribution.
		Req. 12.1.3	The systems shall use the formula $N(\mu_k, \Sigma_k) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)$ to compute the maximization.
		Req. 12.1.4	The system shall compute ɣ(Znk) to find the expected values.
		Req. 12.1.5	The system shall use ɣ(Znk) to recalculate μk, Σk, and πk.
		Req. 12.1.6	The system shall use the recalculated values to maximize the value of ɣ(Znk).
		Req. 12.1.7	The system shall continue to calculate and recalculate as it gets closer and closer to the ideal value.
	HMM	Req. 13.1.1	The system shall define the state space.
		Req. 13.1.2	The system shall intake the observations from the GMM to define the observation space.
		Req. 13.1.3	The system shall intake the initial state transition probabilities πk from the GMM
		Req. 13.1.4	The system shall calculate the likelihood of an observation given a state k.
		Req. 13.1.5	The system shall calculate the most likely sequence of state changes.
		Req. 13.1.6	The system shall then use the output to retrain the HMM and perform steps 14.1.3 through 14.1.5 again until no further improvement.
		Req. 13.1.7	The system shall decode the hidden state sequence.

Decoding Operations		
Viterbi Decoder	Req. 15.1.1	The system shall use a Viterbi decoder to find the most probable hidden phoneme state sequence.
	Req. 15.1.2	The system shall use the information computed in the GMM and HMM to perform the calculations for the Viterbi decoder.
	Req. 15.1.3	The system shall gather the Transition Matrix from the HMM.
	Req. 15.1.4	The system shall gather the Emissions Matrix from the HMM.
	Req. 15.1.5	The system shall gather the State space from the HMM.
	Req. 15.1.6	The system shall gather the Array of initial probabilities from the GMM.
	Req. 15.1.7	The system shall gather the sequence of observation space from GMM.
	Req. 15.1.8	The system shall gather the Observation Space from GMM.
	Req. 15.1.9	The system shall evaluate the model, then output the sequence of states and state transition probabilities that were calculated.

Output			
Weighted Finite State Transducer (WFST)	Weighted Finite State Transducer (WFST)	Req. 16.1.1	The system shall use four WFST models for further processing.
	Hidden Markov Model (HMM)	Req. 17.1.1	The system shall input the output of the Viterbi decoder into a HMM to output triphones.
	Context-Dependency Model	Req. 18.1.1	The system shall input triphones to convert the triphone states into monophones.
	Lexicon	Req. 19.1.1	The system shall use the lexicon to convert monophones into words.
	Language Model	Req. 20.1.1	The system shall use the words from the lexicon out to score the next predicted word.
	WFST	Req. 21.1.1	The system shall combine the four aforementioned models into a single simplified model.
Text file		Req. 22.1.1	The system shall output a text file upon completion of all aforementioned operations.
		Req. 22.1.2	The system shall only write transcribed words into the aforementioned text file.
		Req. 22.2.1	The system shall not write punctuation or grammatical marks into the aforementioned text file.
		Req. 22.3.1	The system shall transcribe words found in the lexicon in uppercase.
		Req. 22.3.2	The system shall transcribe words not found in the lexicon in lowercase.

4. System Features

It should be noted the following subsections are in reference to the sample ASR model provided by the Kaldi ASR Toolkit. Once the Kaldi ASR Team develops its own acoustic ASR model, the following subsections shall be updated in reference to it.

The following section details the system features described in Section 2.2 of this document. The section begins with the reception of the audio file, followed by the processing of the file, the transcription of the audio into text, the output of the transcribed text, and the storing of unknown words into a non-lexicon library.

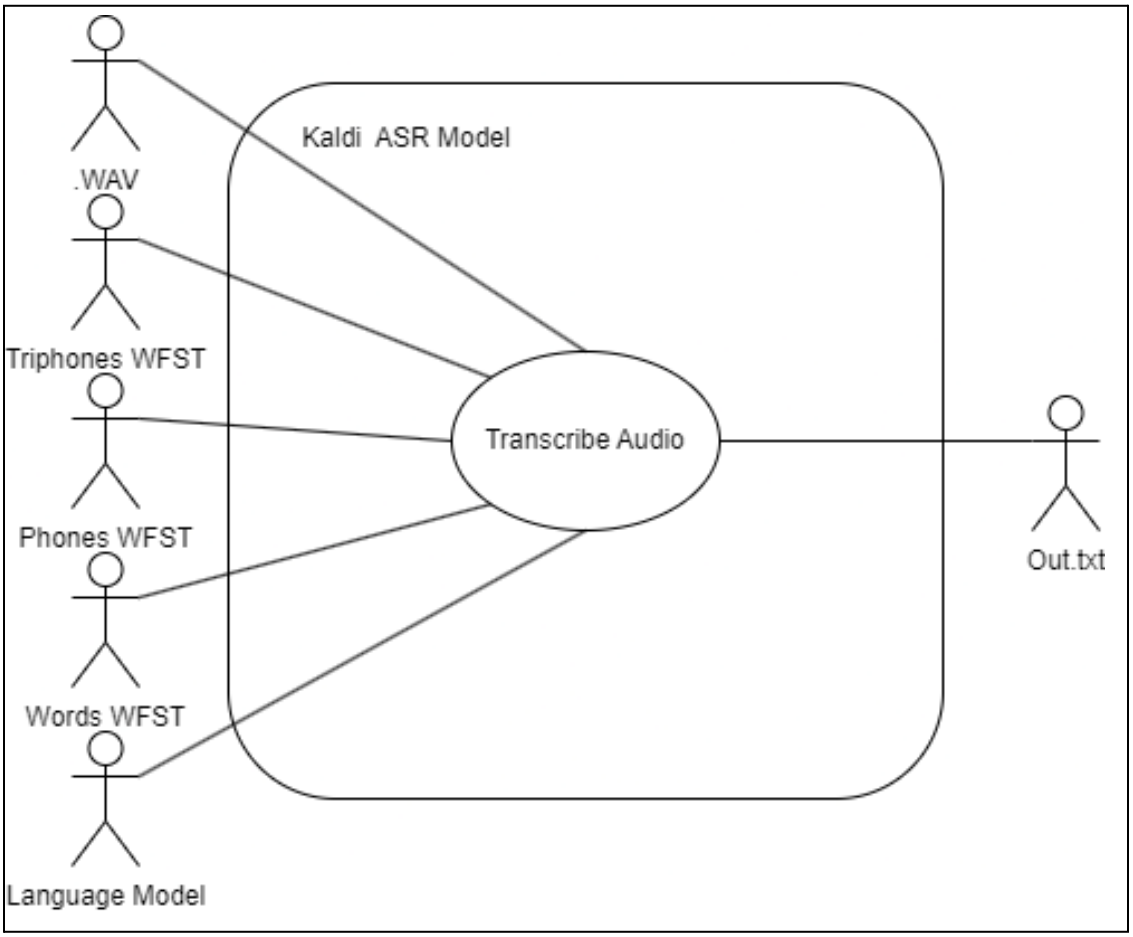

Figure 05: Kaldi ASR Toolkit Sample ASR Model Use Case Diagram V5.1.1

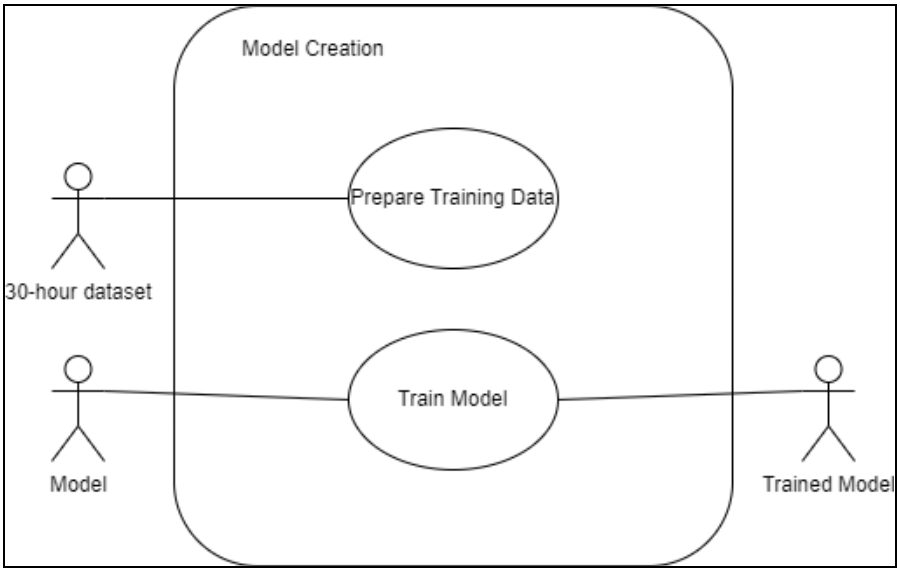

Figure 06: Kaldi ASR Model Training Use Case Diagram V5.2.2

Use Cases:

Use Case ID: UC1

Use Case Name: Transcribe Audio

Goal: The user inputs an audio file in the .WAV format and the ASR model shall return a text file containing the transcribed audio called “out.txt”.

Actors: WAV file, Phones WFST, Triphones WFST, Words WFST, Language Model, “out.txt”

Precondition: Model is properly installed and trained

Post-Condition: “out.txt” is stored in /kaldi/egs/mini_librispeech/s5/

Trigger Condition: The user inputs a WAV file

Main Success Scenario: The model outputs a text file

Step	Actor Action	Step	System Reaction
1	The actor shall input a WAV file	2	The system shall partition the audio data into 25-millisecond long frames in 10-millisecond long sliding intervals, allowing for three frames to overlap
		3	The system shall convert the frames from the time-domain to the frequency-domain
		4	The system shall generate a 39-dimensional MFCC feature vector and populate it with the initial frequency- domain data, the first-order derivative of the data, and the second-order derivative of the data to show changes in the frequency-domain.
6	The actor shall feed to the triphone WFST.	5	The system shall calculate the distance to the most probable phoneme based on the MFCC feature vector and the GMM data to produce a chain of HMM states that represents triphones.
8	The actor shall feed to the monophone WFST.	7	The system shall build triphones from the chain of HMM states using the Viterbi Decoder.
10	The actor shall feed to the word WFST.	9	The system shall convert triphones to monophones based on context-dependency.
12	The actor shall feed to the language model.	11	Builds words using the monophones

		13	Uses the previous two words to predict the following word
		14	The system shall use the language model to assemble the words into sentences.
16	The actor shall output the text file “out.txt”.	15	The system shall transfer sentences to a text file.

Exceptions (Alternative Scenarios of Failure):

1. User inputs non WAV audio file
2. User does not input anything
3. Transcription failure

Alternate Scenarios:

ALT 1: Step 1: The system does not accept any other audio files than WAV.
Step 1.1: The system throws an exception.
Step 1.2: Return to step 1.

ALT 2: Step 0: The system shall not execute if not WAV is imputed.
Step 0.1: Continue to wait

ALT 3: Step 2-15: The system fails to transcribe audio.
Step 2-15.1: The system displays an exception for the failed step.
Step 2-15.2: Return to the failed step.

Use Case ID: UC2

Use Case Name: Train Model

Goal: The ASR model shall be trained using the 30-hour training data.

Actors: Model, Training Data

Pre: ASR model exists, training data exists

Post: ASR model is trained

Main Success Scenario:

Step	Actor Action	Step	System Reaction
1	The actor shall input the model.		
2	The actor shall input the training data	3	The model shall train itself to recognize speech and sound through changes in frequency by using the training data
		4	The system shall attempt to match the labels and continues training until improvement plateaus
		5	The system shall return a trained model

Exceptions (Alternative Scenarios of Failure):
No possible exceptions due to assumption of adequate training data as per customer's specifications.

4.1 Reception of the audio file for processing

4.1.1 Description and Priority

Reception of the audio file for processing from the user via a Python3 script is of low priority. Input shall be given as a terminal command in the format, "python3 main.py filename.wav" (*Figure 3*). The argument, "python3", calls the execution of a Python3 file. The file in question, "main.py, is a script that initiates the execution of the sample ASR model. Lastly, the argument, "filename.wav", calls the WAV file that shall be transcribed by the sample ASR model. The file shall be located in the same directory as the script. In essence, the script shall only execute if it is called alongside a single WAV file in the Ubuntu terminal (*Figure 3*). Any input that does not adhere to this format shall cause the system to throw an exception and print a message in the terminal (*Figures 4, 5, 6*), explained in detail in the proceeding subsection.

4.1.2 Stimulus/Response Sequences

Calling any file type other than WAV shall cause the script to throw an exception and print, "Provided filename does not end in '.wav'" (*Figure 4*). Calling multiple files of any type shall cause the script to throw an exception and print, "Too many arguments provided. Aborting" (*Figure 5*). In the absence of a WAV file, the system shall attempt a traceback to the most recently called WAV file; if unsuccessful, the system shall throw a "ValueError" exception and print, "No .wav file in the root directory" (*Figure 6*).

It is highly advised that the user converts other audio file types (e.g., FLAC) into WAV beforehand using the FFmpeg Linux utility. The format for the terminal command is "ffmpeg filename.filetype filename.wav" (*Figure 7*). The argument, "ffmpeg", calls the execution of the FFmpeg utility. The "-i" flag denotes that the next argument, "filename.filetype", is the input audio file. Lastly, the result of the conversion is specified by the argument, "filename.wav", which shall ideally have the same filename as the input file. Once executed, the utility converts the audio file into a WAV file, and the user shall be able to execute the aforementioned Python3 script without issue (*Figure 3*).

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.wav
tree-info exp/chain_cleaned/tdnn_1d_sp/tree
tree-info exp/chain_cleaned/tdnn_1d_sp/tree
```

Figure 07: Sample ASR Model Successful Input (with code execution snippet)

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.flac
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 16, in <module>
    raise ValueError("Provided filename does not end in '.wav'")
ValueError: Provided filename does not end in '.wav'
```

Figure 08: Sample ASR Model Unsuccessful Input; non-WAV input exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py gettysburg.wav gettysburg.wav
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 18, in <module>
    raise ValueError("Too many arguments provided. Aborting")
ValueError: Too many arguments provided. Aborting
```

Figure 09: Sample ASR Model Unsuccessful Input; too many arguments exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ python3 main.py
Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 10, in <module>
    FILE_NAME_WAV = glob.glob("*.wav")[0]
IndexError: list index out of range

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/redhocus/project/kaldi/egs/kaldi-asr-tutorial/s5/main.py", line 12, in <module>
    raise ValueError("No .wav file in the root directory")
ValueError: No .wav file in the root directory
```

Figure 10: Sample ASR Model Unsuccessful Input; no file in directory exception

```
redhocus@LAPTOP-GTI83R2U:~/project/kaldi/egs/kaldi-asr-tutorial/s5$ ffmpeg -i gettysburg.flac gettysburg.wav
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl
--disable-stripping --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable-lib
dav1d --enable-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-
libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librabbitmq --enable-librubberband --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --
enable-libsrt --enable-libssh --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --enable-libx265 --enable-libxml2 --enable-l
ibxvid --enable-libzimg --enable-libzmq --enable-libzvbi --enable-lv2 --enable-omx --enable-openal --enable-opencl --enable-opengl --enable-sdl2 --enable-pocketsphinx --enable-librsvg -
-enable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libx264 --enable-shared
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, flac, from 'gettysburg.flac':
  Metadata:
    encoder         : Lavf58.76.100
  Duration: 00:00:10.00, start: 0.000000, bitrate: 178 kb/s
  Stream #0:0: Audio: flac, 22050 Hz, mono, s16
Stream mapping:
  Stream #0:0 -> #0:0 (flac (native) -> pcm_s16le (native))
Press [q] to stop, [?] for help
Output #0, wav, to 'gettysburg.wav':
  Metadata:
    ISFT            : Lavf58.76.100
  Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 22050 Hz, mono, s16, 352 kb/s
    Metadata:
      encoder         : Lavc58.134.100 pcm_s16le
size=     431kB time=00:00:09.92 bitrate= 355.6kbits/s speed= 769x
video:0kB audio:431kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.017682%
```

Figure 11: Execution of FFmpeg utility; conversion of FLAC to WAV

4.1.3 Functional Requirements

Req. 4.1.3.1	The script shall throw an exception if any file type other than WAV is called.
Req. 4.1.3.2	The script shall throw an exception if multiple files are called.
Req. 4.1.3.3	The script shall attempt a traceback of the most recently transcribed WAV file if a WAV file is not called.
Req. 4.1.3.4	The script shall throw an exception upon the failure of the traceback.
Req. 4.1.3.5	The system shall initiate the transcription process upon the successful execution of the script
Req 4.1.3.6	The system shall only accept input from a command-line interface (CLI).

4.2 Transcription of audio into text

4.2.1 Description and Priority

The ASR model transcribes audio into text by utilizing a database of phones, triphones, and words, as well as a series of models for each database (*Figure 8*). For this project, transcription is a high-priority feature. The system shall prepare the data for transcription by partitioning the audio into 25-millisecond frames in 10-millisecond sliding intervals that shall be converted from the time-domain to the frequency-domain. The system then performs calculations to convert each frequency-domain frame into a MFCC feature vector, computes expectation maximization of the MFCC feature vector in reference to phoneme Gaussian distribution, calculates the probability of a state switch based on a given observation for the entire feature vector, then calculates the state sequence and takes the sequence and converts it to sentences.

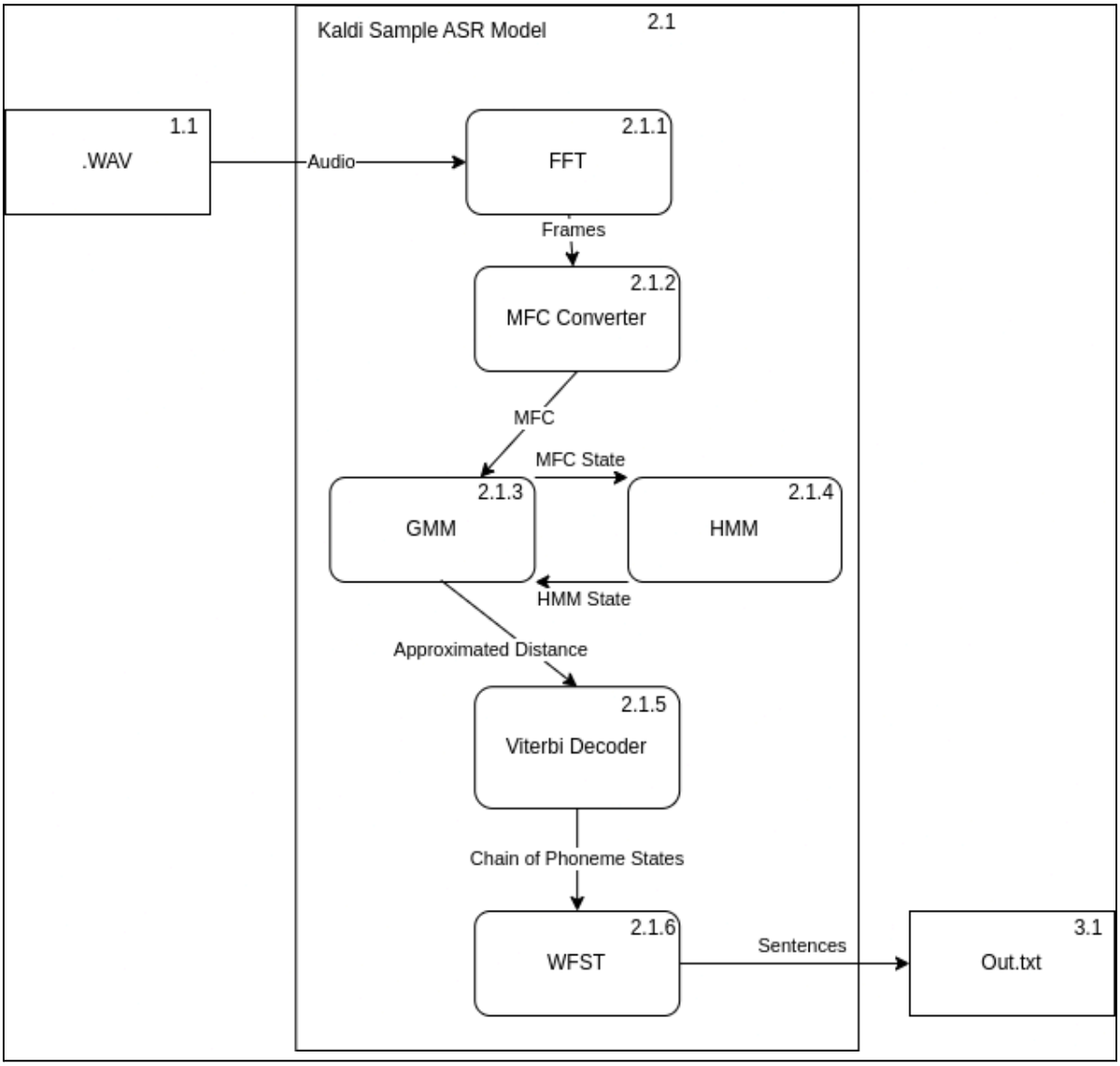

Figure 12: Kaldi ASR Toolkit Sample ASR Model Level 1 Data Flow Diagram V4.1.1

4.2.2 Stimulus/Response Sequences

	Step	System Reaction
UC1	2	The system shall partition the audio data into 25-millisecond long frames in 10-millisecond long sliding intervals, allowing for three frames to overlap
	3	The system shall convert frames from the time-domain to the frequency-domain
	4	The system shall generate a 39-dimensional MFCC feature vector and populate it with the initial frequency- domain data, the first-order derivative of the data, and the second-order derivative of the data to show changes in the frequency-domain.
	5	The system shall calculate the distance to the most probable phoneme based on the MFCC feature vector and the GMM data to produce a chain of HMM states that represents triphones.
	7	The system shall build triphones from the chain of HMM states using the Viterbi Decoder.
	9	The system shall convert triphones to monophones based on context-dependency
	11	The system shall build words using the monophones
	13	The system shall use the previous two words to predict the following word
	14	The system shall use the language model to assemble the words into sentences
	15	The system shall transfer sentences to a text file
	2	The system shall partition the audio data into 25-millisecond long frames in 10-millisecond long sliding intervals, allowing for three frames to overlap.
	3	The system shall convert frames from the time-domain to the frequency-domain

4.2.3 Functional Requirements

Req 1.1.1	The system shall pad the signals so that they never equal zero .
Req. 2.1.1	The system shall partition the WAV file into overlapping 25-millisecond frames.
Req. 2.2.1	The system shall obtain each frame in 10-millisecond sliding intervals starting from time equals 0.
Req. 2.2.2	The system shall use the formula $[10n,\ 10n\ +\ 25]$ to obtain the closed time intervals of each frame (e.g., $[0,\ 25]$, $[10,\ 35]$, $[20,\ 45]$, ...).
Req. 2.2.3	The system shall define the value n as any non-negative integer (i.e., $\{n \in \mathbb{Z} : n \geq 0\}$).
Req. 2.3.1	The system shall use a FFT to convert the audio data within the frames from the time-domain to the frequency domain.
Req. 3.1.1	The system shall convert the results of the FFT into Mel numbers.
Req. 3.2.1	The system shall use the summation of the cosine function oscillating on each frame to create a set of 13 MFCCs for that frame.
Req. 3.2.2	The system shall derive the first-order derivative of the 13 MFCCs.
Req. 3.2.3	The system shall derive the second derivative of the 13 MFCCs.
Req. 3.2.4	The system shall produce a 39-dimensional MFC represented as an MFCC feature vector.
Req. 3.2.5	The system shall use the first set of thirteen dimensions (i.e., [0, 12]) to represent the thirteen MFCCs.
Req. 3.2.6	The system shall use the second set of thirteen dimensions (i.e., [13, 25]) to represent the first-order derivatives of the MFCCs.
Req 3.2.7	The system shall use the third set of thirteen dimensions (i.e., [26, 38]) to represent the second-order derivatives of the MFCCs.
Req. 4.1.1	The system shall initialize three components: The mean of the component μk, the variance of the component Σk, the initial probability of the component πk
Req. 4.1.2	The system shall compute ɣ(Znk) find the expected values
Req. 4.1.3	The system shall use ɣ(Znk) to recalculate μk, Σk, and πk
Req. 4.1.4	The system shall use the recalculated values to maximize the value of ɣ(Znk)
Req. 4.1.5	The system shall continue to calculate and recalculate as it gets closer and closer to the ideal value
Req. 4.2.1	The system shall define the state space
Req. 4.2.2	The system shall intake the observations from the GMM to define the observation space
Req. 4.2.3	The system shall intake the initial state transition probabilities πk from the GMM
Req. 4.2.4	The system shall calculate the likelihood of an observation given a state k

Req. 4.2.5	The system shall calculate the most likely sequence of state changes
Req. 4.2.6	The system shall then use the output from Req. 4.2.5 to retrain the HMM and perform steps 4.2.3 through 4.2.6 again until no further improvement
Req. 4.2.7	The system shall decode the most probable sequence of hidden states
Req. 4.2.8	The system shall evaluate the model and then output the sequence of states and state transition probabilities that were calculated
Req. 5.1.1	The system shall use a Viterbi decoder to find the most likely hidden phoneme state sequence.
Req. 5.1.2	The system shall use the information computed in the GMM and HMM to perform the calculations for the Viterbi decoder.
Req. 5.1.3	The system shall gather the Transition Matrix from the HMM.
Req. 5.1.4	The system shall gather the Emissions Matrix from the HMM.
Req. 5.1.5	The system shall gather the State space from the HMM.
Req. 5.1.6	The system shall gather the Array of initial probabilities from the GMM.
Req. 5.1.7	The system shall gather the sequence of observation space from GMM.
Req. 5.1.8	The system shall gather the Observation Space from GMM.
Req. 6.1.1	The system shall use four WFST models for further processing.
Req 7.1.1	The system shall input the output of the Viterbi decoder into a HMM to output triphones.
Req. 8.1.1	The system shall input triphones to convert the triphone states into monophones.
Req. 9.1.1	The system shall use the lexicon to convert monophones into words.
Req. 10.1.1	The system shall use the words from the lexicon out to score the next predicted word.
Req. 11.1.1	The system shall combine the four aforementioned models into a single simplified model.

4.3 Output a text file with the words in the audio file

4.3.1 Description and Priority

Outputting a file “out.txt” containing the words transcribed from the audio file is of high priority. The sample ASR model shall output a file “out.txt” and overwrite any previously existing instance of the file in the same directory.

4.3.2 Stimulus/Response Sequences

	Step	System Reaction
UC1	17	The system shall transfer sentences to a text file.

4.3.3 Functional Requirements

Req 4.3.3.1	The system shall write the sentence output from the Language Model into a text file called “out.txt”.
Req 4.3.3.2	The system shall store “out.txt” in the folder: /model_name/s5.
Req 4.3.3.3	The system shall transcribe words stored in the lexicon in uppercase.
Req 4.3.3.4	The system shall transcribe words not stored in the lexicon in lowercase
Req 4.3.3.5	The system shall spell all words not in the lexicon phonetically.
Req 4.3.3.6	The system shall not transcribe punctuation or grammatical markings.
Req 4.3.3.7	The system shall overwrite any previously existing instance of “out.txt” upon every transcription.
Req 4.3.3.8	The system shall append transcribed words not stored in the lexicon into a separate library.

Appendix A: Glossary

Term	Definition
ATC	***Air Traffic Control;*** the service that elicits communications between pilots and helps to prevent air traffic accidents.
ASR	***Automatic Speech Recognition;*** the ability for computers to recognize and translate spoken speech.
CLI	***Command-Line Interface;*** text-based interface that allows interaction from the user to the computer program.
DNN	***Deep Neural Network;*** a machine learning technique that represents learning and processing data in artificial neural networks.
ERAU	***Embry Riddle Aeronautical University;*** an aviation-centered university located in Daytona Beach, Florida, United States.
FFmpeg	***Linux utility;*** a portable open-source utility that allows users to decode, encode, transcode, multiplex, demultiplex, stream, filter, and play most human- or machine-made multimedia.
FFT	***Fast Fourier Transform;*** algorithm used to obtain the spectrum or . frequency content of a signal.
General American English	The most spoken variety of the English language in the United States.
GMM	***Gaussian Mixture Model;*** used to calculate the distance between the MFC feature vector and the HMM state.
HMM	***Hidden Markov Model;*** used to find the state locations of the phonemes..
IPA	***International Phonetic Alphabet:*** an alphabetic system of phonetic notation developed by the International Phonetic Association; used to represent speech sounds in a standardized format.
Lexicon	*pertaining to speech;* a library of words that is understood by the language model.
MFC	***Mel-Frequency Cepstrum;*** a representation of the short-term power spectrum of a sound
MFCCs	***Mel-Frequency Cepstral Coefficients;*** the coefficients that a MFC is comprised of
NLP	***Natural Language Processing;*** the culmination of computer science, linguistics, and machine learning.
Phone	*pertaining to speech;* a distinct speech sound or gesture;
Phoneme	*pertaining to speech;* a set of phones that can distinguish one word from another
Triphone	*pertaining to speech;* a sequence of three consecutive phonemes
WER	***Word Error Rate;*** the rate at which error in words occurs

Appendix B: Analysis Models

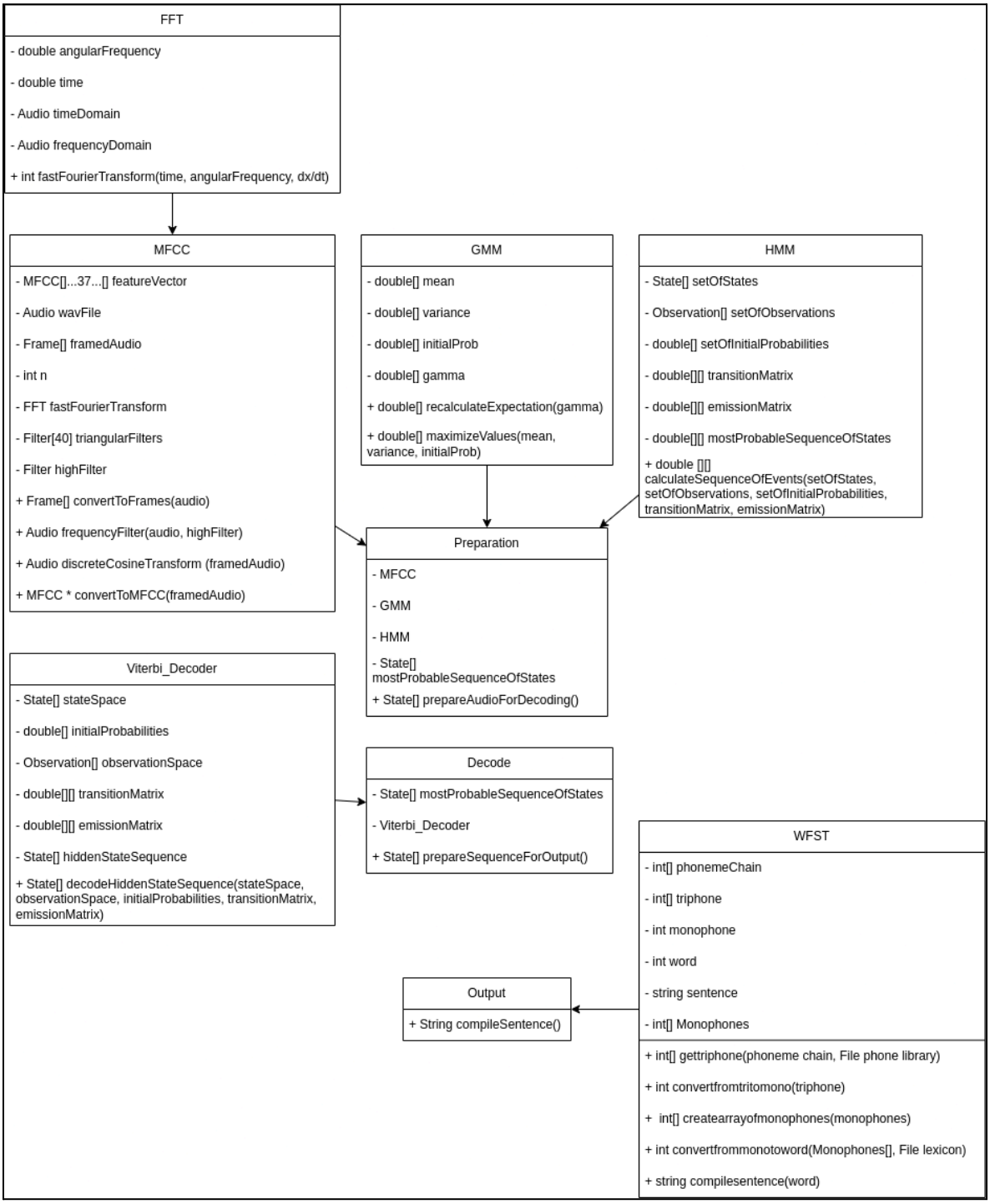

Figure 13: Kaldi ASR Class Diagram V4.1.3

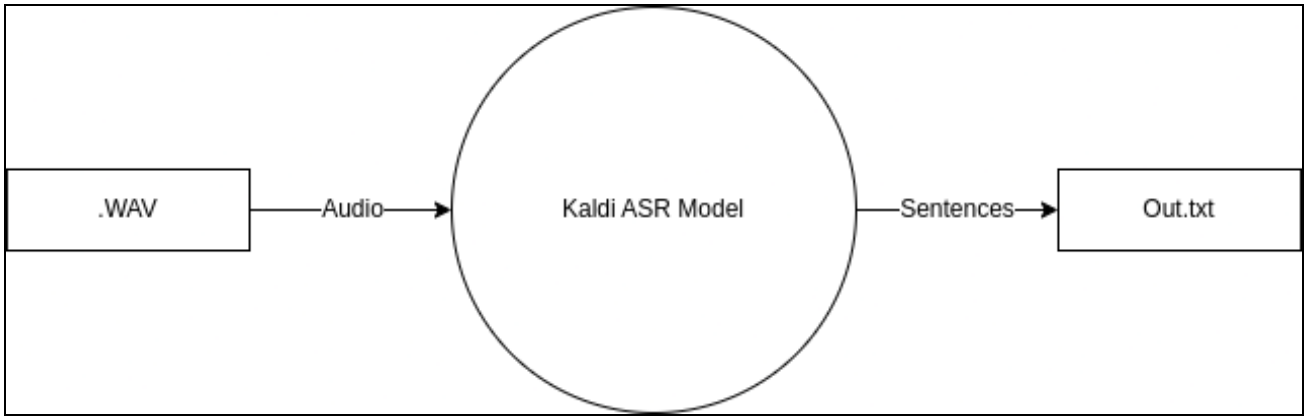

Figure 14: Kaldi Sample ASR Context Diagram V4.1

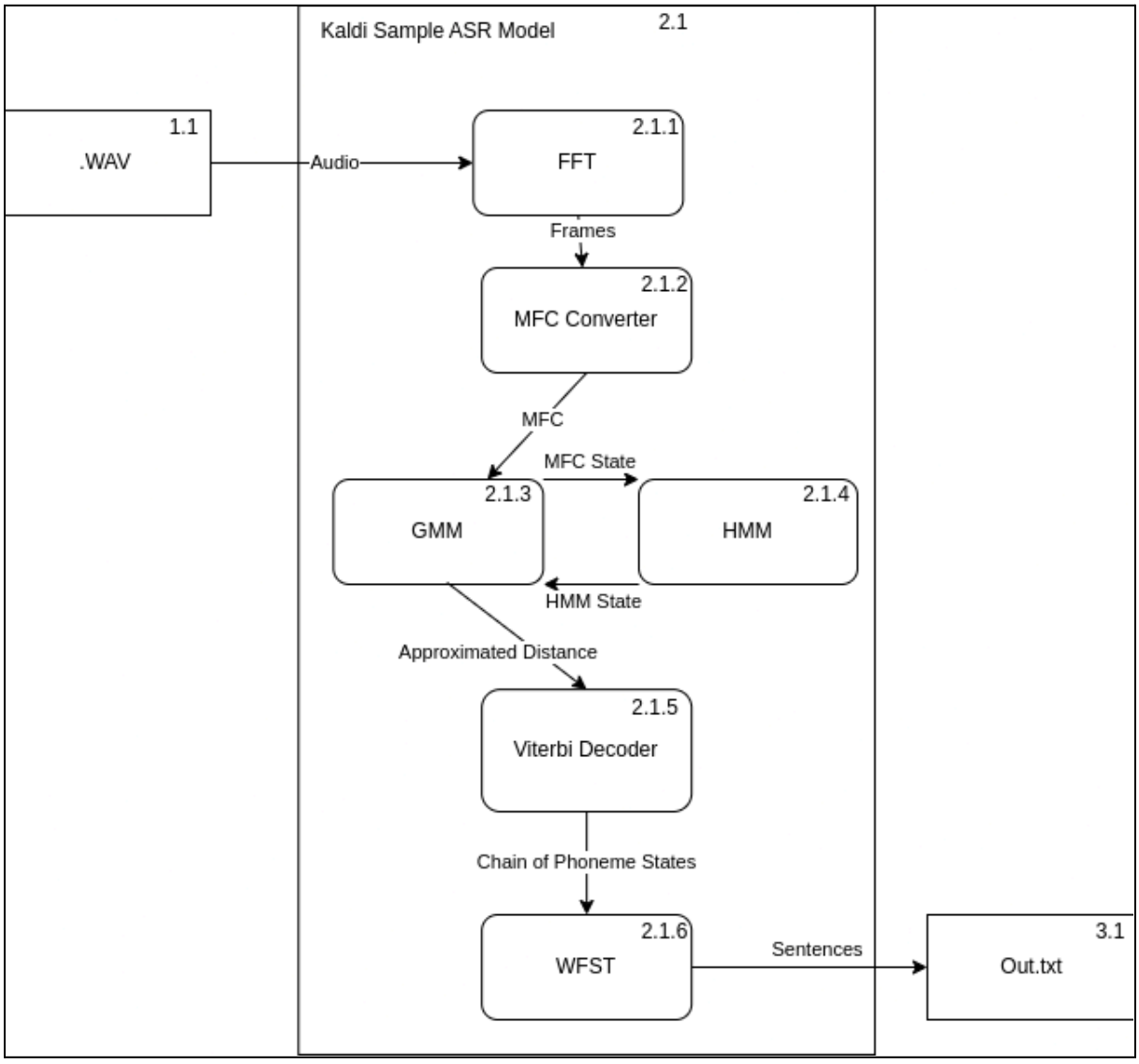

Figure 15: Kaldi ASR Toolkit ASR Model Level 1 Data Flow Diagram V4.1.1

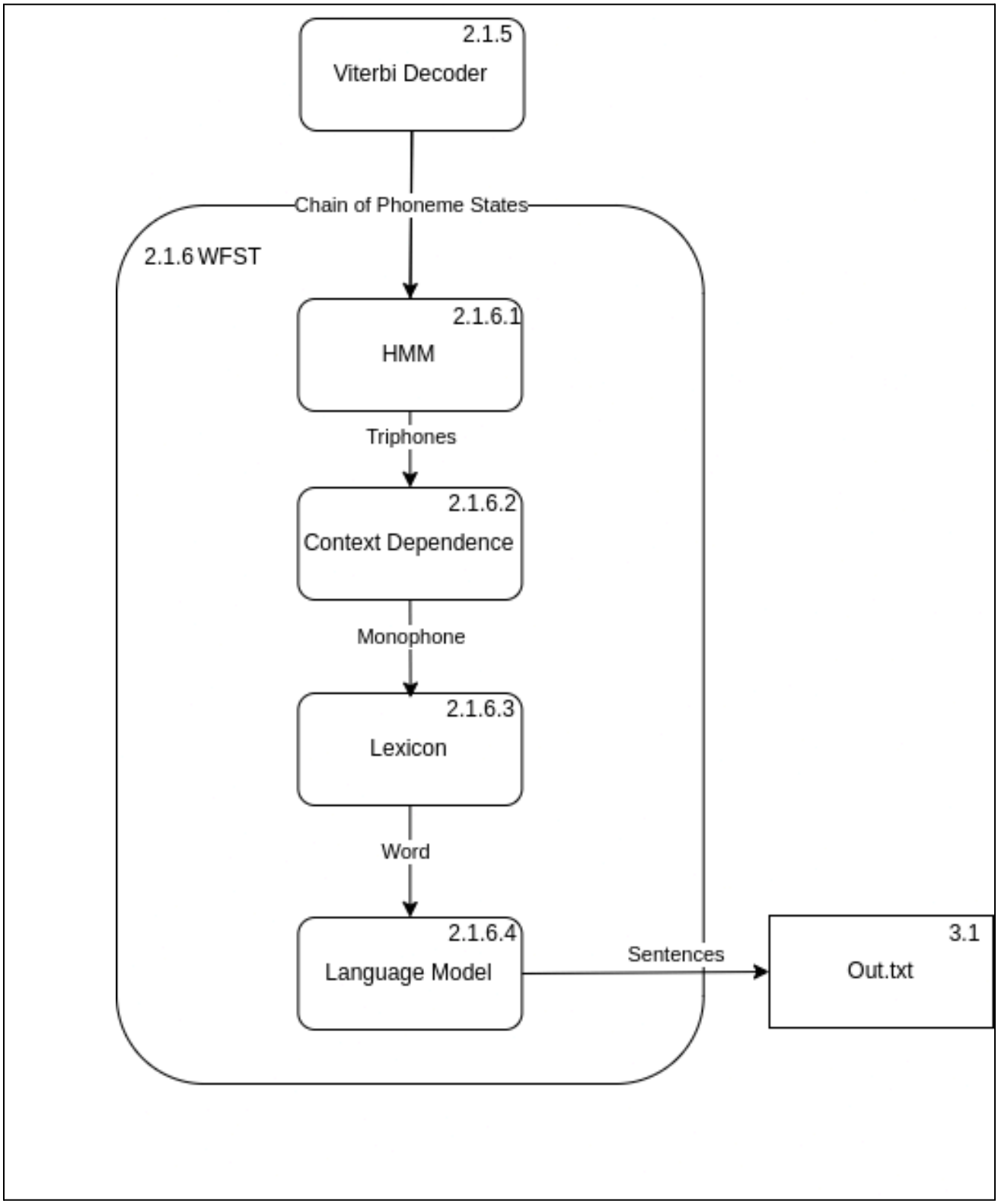

Figure 16: Kaldi ASR Toolkit ASR Model Level 2 Data Flow Diagram 4.2.1

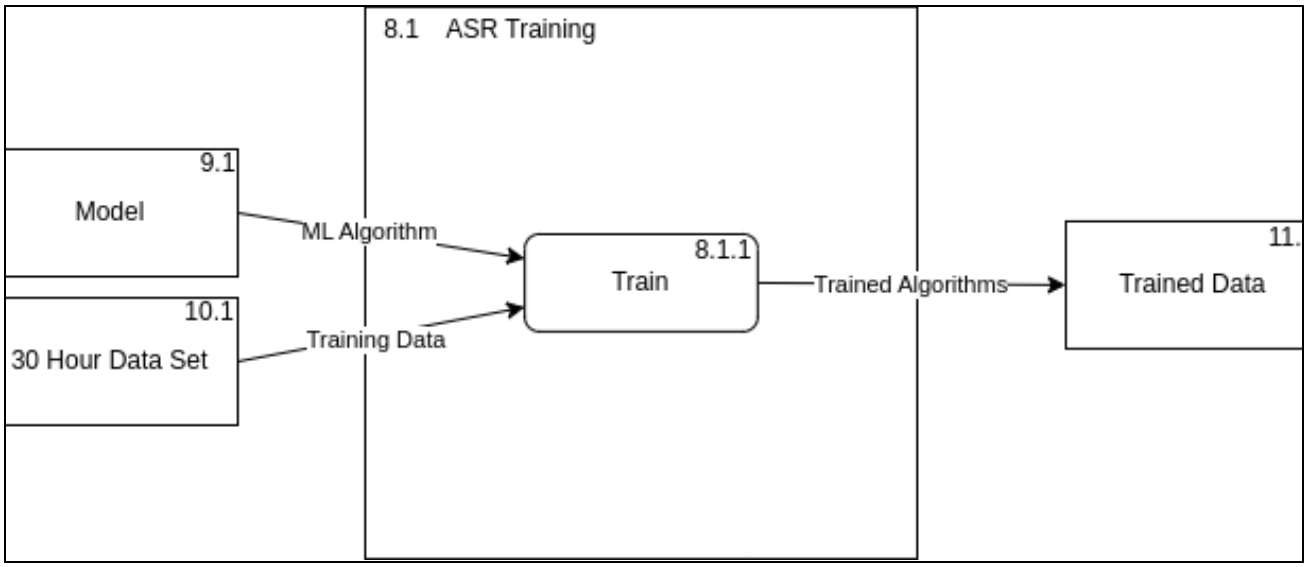

Figure 17: Kaldi ASR Training Model Level 1 Data Flow Diagram V4.3.1

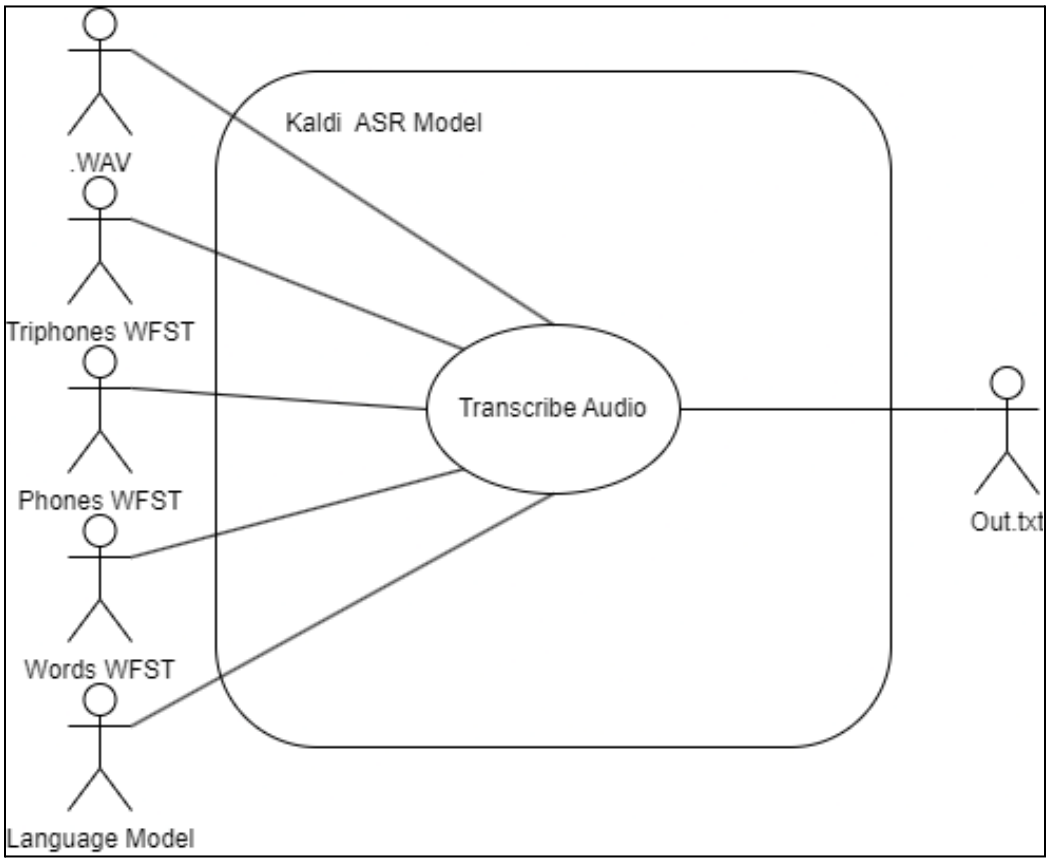

Figure 18: Kaldi ASR Toolkit ASR Model Use Case Diagram V5.1.1

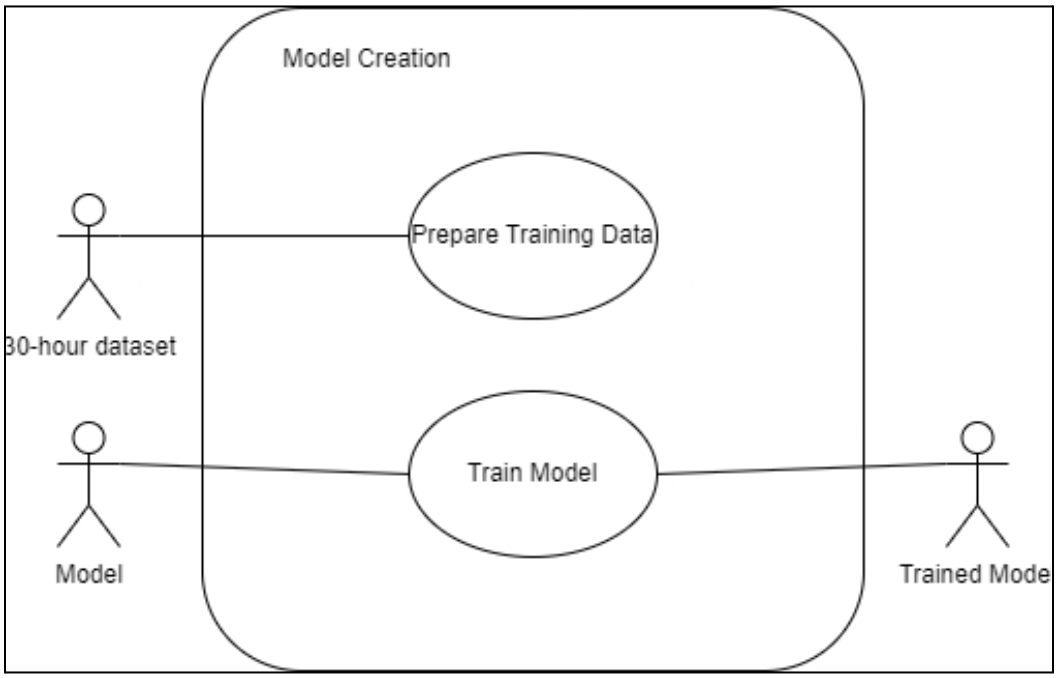

Figure 19: Kaldi ASR Toolkit ASR Model Use Case Diagram V5.2.2