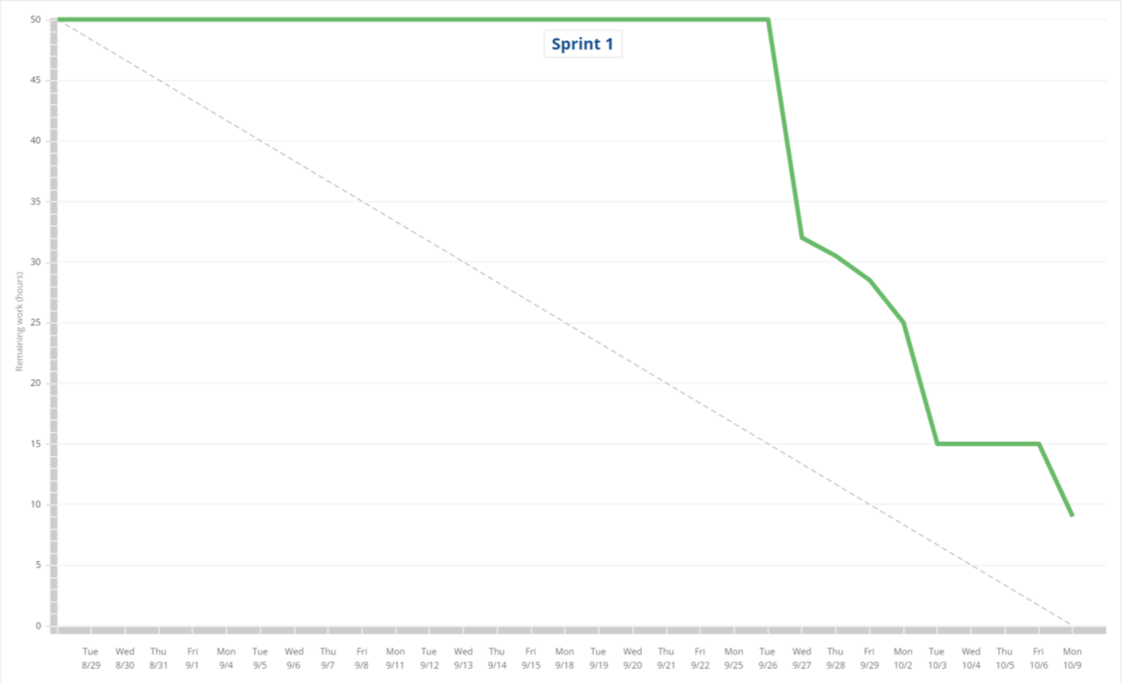
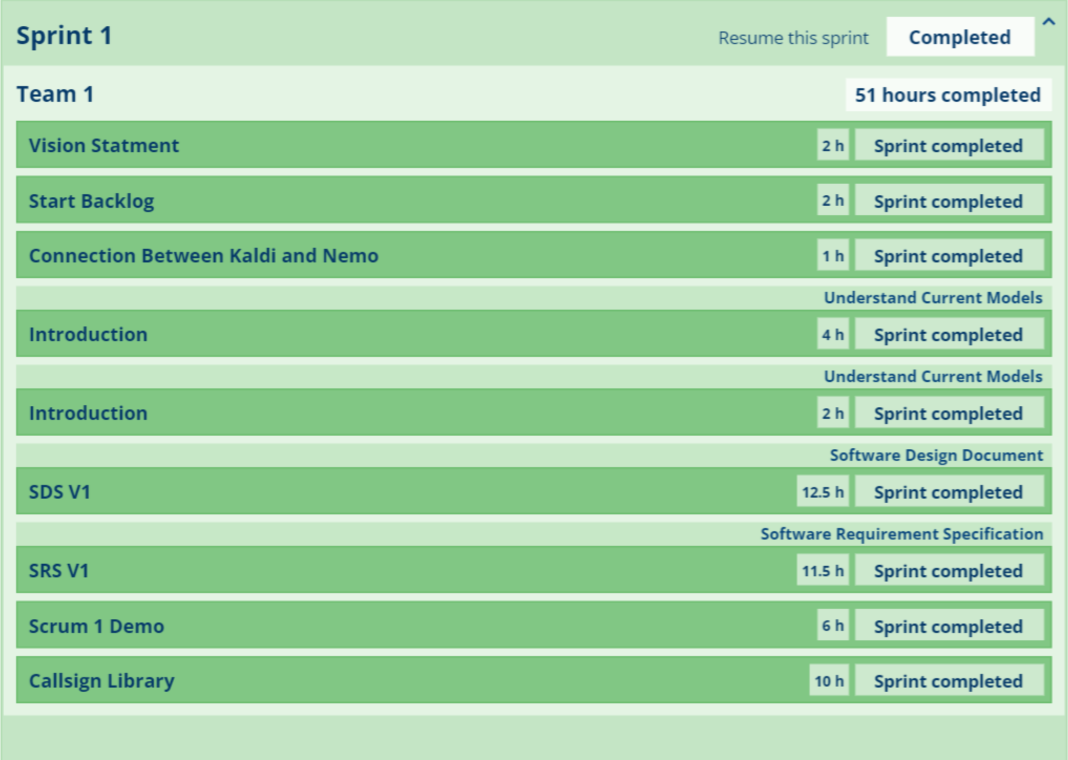
Engineering Notebook

David Serfaty

Sprint 1





Sprint 1 Resume this sprint Completed

Team 1 51 hours completed

Vision Statment 2 h Sprint completed

Start Backlog 2 h Sprint completed

Connection Between Kaldi and Nemo 1 h Sprint completed

Understand Current Models

Introduction 4 h Sprint completed

Understand Current Models

Introduction 2 h Sprint completed

Software Design Document

SDS V1 12.5 h Sprint completed

Software Requirement Specification

SRS V1 11.5 h Sprint completed

Scrum 1 Demo 6 h Sprint completed

Callsign Library 10 h Sprint completed

SDD

Name	Date	Reasons For Change	Version
Tabitha, Milan, David, Max, Tisha, Adam	09/29/2023	Starting Document	V1.0

SRS

Name	Date	Reason For Changes	Version
Tabitha, Milan, Tisha, David, Adam, Max	09/29/23	Starting the document	V1.0

Test Plan

Name	Date	Reason For Changes	Version
All	10/18/2023	Write wrong information	V1.0

9/21

Speech Recognizer
Text generator -> W [Speech generator -> (Signal Processing] X -> Speech Decoder) -> W^
W is the sequence of words from the speaker (Utterance)
X is the speech signal, using X we can extract O in the frequency domain
W^ is the sequence we want from given X or O

O is a vector made up of 39 frequency components

Speech sample -> Pre-emphasis -> Framing -> Windowing -> DFT -> Mel Frequency wrapping -> Log Operation -> {FBANK features, DCT -> MFCC features}
Each frame is 25 ms
Skip 10 ms from previous frame
Use Mel frequency to better simulate human ear
Use Log operation to reduce change range
Use discrete cosine transform to reduce correlation between dimensions (important for GMM [Gaussian Mixture Model])
KALDI will be using DCT to then generate MFCC features

Probability is cool and based
- Probability of an event
- An event to a random variable RV
  - PFM for discrete RV
  - PDF for continuous RV
- PDFs of common distributions
  - Uniform
  - Gaussian
- PDFs of parameters
  - Gaussian based on mean and variance

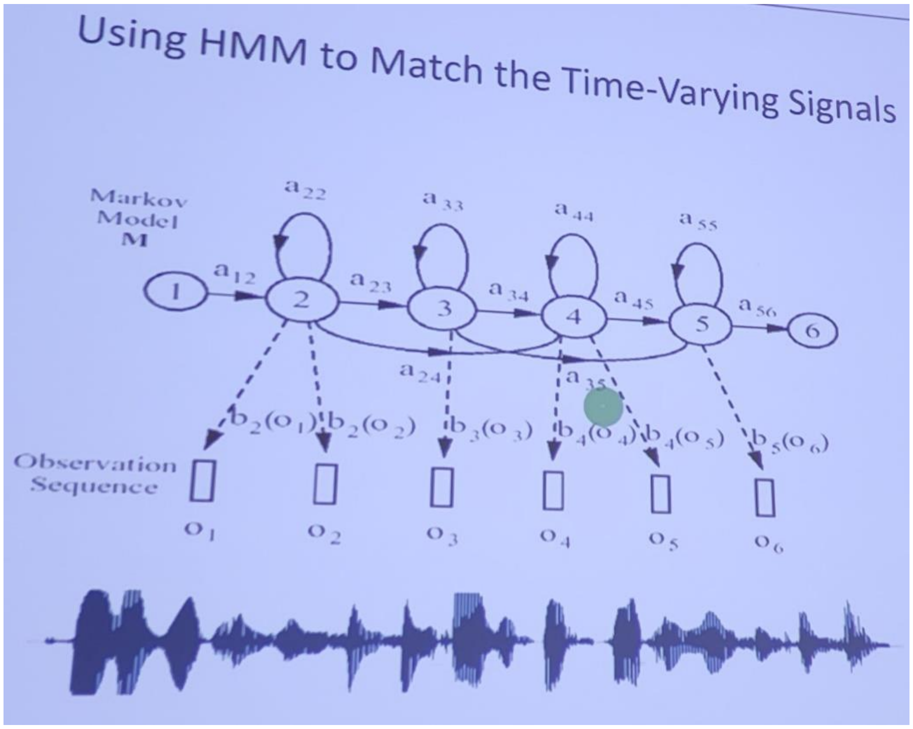
GMM and HMM
Gaussian mixture model for approximating complex PDFs
Multiple RVs for a certain event
Conditional probability
Bayesian theory (p(a|b) to p(b|a))
Markov chain (p(s2|s1, s0) = p(s2|s1))
Hidden Markov Model (HMM)
- System described by states
- State cannot be observed
- State transitions in a non-backward matter
- Each state transmits observable RV
- Use observable RV to infer the state of the HMM

Each word is a random variable
Conditional probability is used to determine future word based on previous words
Markov chain is probability of word given latest word should be same as probability given previous 2 words
HMM is Markov chain determined by unobservable states, states transition in only one direction
“The states will be generating something, just say that” - Dr. Liu

Each state is NOT a word OR letter, but the sounds themselves (please refer to Milan's notes as he is better with language than myself)

Phones and Triphones

Phones are used to model the pronunciation of words in an utterance
Speech recognition can be reformulated as phone recognition
We need to train the system to recognize different phones
These phones form a lexicon
To better model the transitions we use triphones (three phones combined [again refer to milan])
There are far fewer triphones required to form most words than phones making training an AI model far easier to do



Using HMM to Match the Time-Varying Signals

Two probability models to train

Phones and triphones cannot be observed, but are modeled using an HMM each
Each HMM has several states to better model change of pronunciation
We need to train the model that describes the transition from one state to another using the training dataset
For a given state the probability of MFCC is described by using a GMM
We train the GMM using the training dataset as well
The training is done by an iterative approach called EM (expectation maximization) as we don't have well aligned speech feature vs labels
We need to use the monophone model to estimate the aligned speech feature vs label pairs
Then we move on to the triphones

First phase is monophone training
Second phase is triphone training

Formulas for Speech Recognition

Check the powerpoint because fuck writing that

Decision Trees and Senone

The total number of triphones is too much to train or use

Decision trees can significantly shrink the size of useful triphones

We are going to rely on decision trees made by linguists

Each triphone is expressed using a HMM with three states including the start and end (Senone)

To further reduce the parameters of the model we can tie the GMM for some senones together

Weighted Finite State Transducer (WFST)

When we know the phones we can get the word using a WFST

- Input is a sequence of phones with weights
- Output is a sequence of word(s)
- Each word has a WFST

There are four WFSTs used in this model

- G (grammar): words in words out
- L (pronunciation): phones in words out
- C (context): triphones in phones out
- H (HMM): HMM states in triphones out

The above can be combined

The HMM states are estimated based on $P(S|O)$ using a Viterbi algorithm, S is state, O is a feature vector

A lattice is used for decoding

9/26

Product Backlog Sprint 1

Tahmina Tisha tishat@my.erau.edu (2545299),
Tabitha O'Malley hudsot12@my.erau.edu (2496633),
David Serfaty serfatyd@my.erau.edu (2540285),
Maxwell Moolchan moolcham@my.erau.edu (2526260),
Milan Haruyama haruyamm@my.erau.edu (2544936),
Adam Gallub gallubM@my.erau.edu (2507331),

Backlog

- Frequency Identification, Due TBD, 20 hours, Milan Haruyama
- Audio Input, Due TBD, 2 hours, David Serfaty, Milan Haruyama, Maxwell Moolchan, Tahmina Tisha
- Data Storage, Due TBD, 20 hours, Tahmina Tisha, Maxwell Moolchan
- Scrum 1 Demo, Due Oct 10, 2 - 3 hours, All members
- Add Backlog for for sprint 2, Due Oct 9, 1 hour, All members
- Software Requirement Documentation, Due TBD 60 hours, Tabitha Hudson, Milan Haruyama, David Serfaty, Maxwell Moolchan, Tahmina Tisha

To Do

SRS V1, Due Sep 28, 10 hours, Tabitha Hudson, Milan Haruyama, David Serfaty, Maxwell Moolchan, Tahmina Tisha

- Introduction
  - Purpose
  - Document Conventions
  - Intended Audience
  - Product Scope
  - References
- Overall Description
  - Product Perspective
  - Product Functions
  - User Classes
  - Operating Environment
  - Design and Implementation Constraints
  - User Documentation
  - Assumptions and Dependencies
- External Interface Requirements
  - User Interfaces
  - Hardware Interfaces
  - Software Interfaces
  - Communications Interfaces
- System Features

    - Systems Features
- Other Nonfunctional Requirements
    - Performance Requirement
    - Safety Requirement
    - Security Requirement
    - Software Quality Attributes
    - Business Rules
- Other Requirements
    - Other Requirements
    - Appendix A
    - Appendix B
    - Appendix C

SDS V1, Due Sept 28, 10 hours, Tabitha Hudson, Milan Haruyama, David Serfaty, Maxwell Moolchan, Tahmina Tisha

- Introduction
    - Purpose and Scope
    - Project Executive Summary
    - System overview
    - Design Constraints
    - Future Contingencies
    - Document Organization
    - Project References
    - Glossary
- System Architecture
    - System Hardware Architecture
    - System Software Architecture
    - Internal Communications Architecture
- Human-Machine Interface
    - Inputs
    - Outputs
- Detailed Design
    - Hardware Detailed Design
    - Software Detailed Design
    - Internal Communication Detailed Design
- External Interfaces
    - Interface Architecture
    - Interface Detailed Design
- System Integrity Controls
    -

- Programming Language Familiarization, Due TBD, 10 hours, Tabitha Hudson, Milan Haruyama, David Serfaty, Maxwell Moolchan, Tahmina Tisha
- Callsign Library, Due TBD, 20 hours, Milan Haruyama, Adam Gullub
- Model Understanding, Due TBD, 30 hours, All members

In Progress

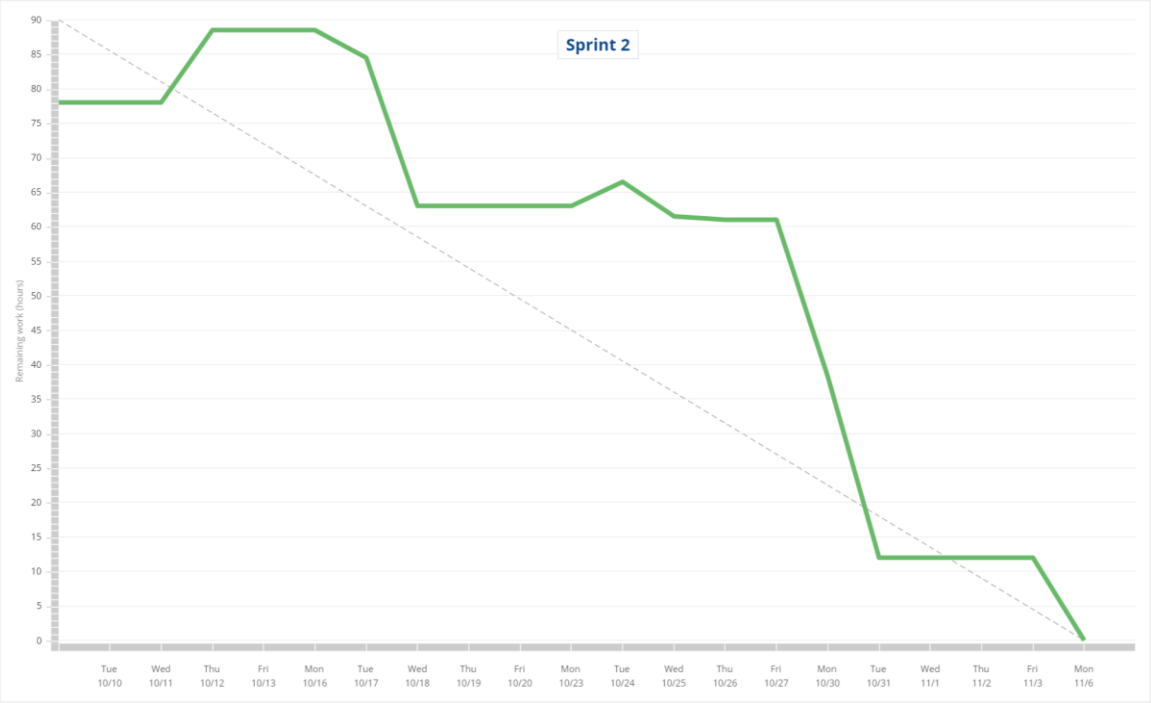
Done

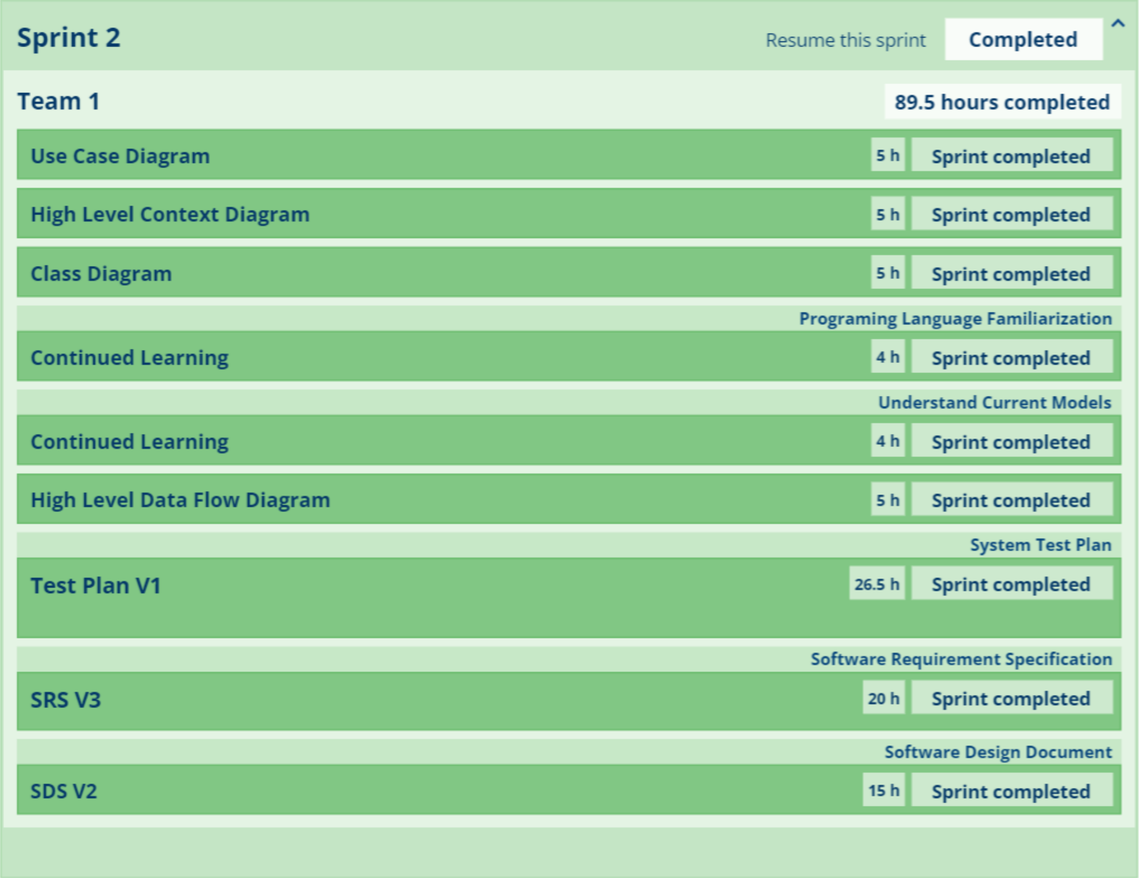
Vision Statement, Due Sep 19, 2 hours, All members

Start Backlog, Due Sep 19, 2 hours, All members

Connection between Kaldi and Nemo, Due TBD, 1 hour, Tabitha Hudson

Sprint 2





Sprint 2 Resume this sprint Completed

Team 1 89.5 hours completed

Use Case Diagram	5 h	Sprint completed
High Level Context Diagram	5 h	Sprint completed
Class Diagram	5 h	Sprint completed
Programing Language Familiarization		
Continued Learning	4 h	Sprint completed
Understand Current Models		
Continued Learning	4 h	Sprint completed
High Level Data Flow Diagram	5 h	Sprint completed
System Test Plan		
Test Plan V1	26.5 h	Sprint completed
Software Requirement Specification		
SRS V3	20 h	Sprint completed
Software Design Document		
SDS V2	15 h	Sprint completed

SDD

Name	Date	Reasons For Change	Version
Tabitha	10/24/2023	Writing Section:1.2.1	V2.1
Tisha, Tabitha, Milan	10/24/2023	Rewriting the section: 2.2	V2.2
Tabitha, Tisha, Milan	10/24/2023	Writing the section, Rewriting, and editing: 1.2	V2.3
Tabitha	10/25/2023	Writing Sections: 2.1, 1.5	V2.4
Tabitha	10/26/2023	Writing Sections: 1.1, 1.2.2, 1.3	V2.5
David	10/26/2023	Writing Sections: 1.2, 1.5, 3.1, 3.2	V2.6
Milan	10/26/2023	Writing Sections: 1.1, 1.2, 1.5	V2.7
Adam	10/28/2023	Asking TA: 2.1 Write Section: 4.1	V2.8
Tisha	10/28/2023	Asking TA: 2.1 Writing Section: 2.1, 5.1	V2.9
Tabitha	10/29/2023	Writing/Rewriting Sections: : 1.2.1, 2.1, 2.2, 3.1, 3.2, 4, 4.1, 4.2, 5.2	V2.10
David	10/29/2023	Writing/Rewriting Sections: 1.2.1, 1.2.2, 1.2.3, 2.1. 2.2, 3.1, 3.2, 4, 4.1, 5, 5.1, 6	V2.11
Tisha	10/29/2023	Writing/Rewriting Section: 2.1	V2.12
Milan	10/29/2023	Editing All Sections	V2.13
Milan	10/30/2023	Editing All Sections	V2.14
Tabitha	10/30/2023	Rewriting Section: 2.1	V2.15
Tabitha	10/31/2023	Updating Models Editing Sections Writing Section: 4.2	V2.16
David	10/31/2023	Rewriting sections: 1.5, 5.2 Editing sections Updating models (context, use case, DFD)	V2.17
Milan	10/31/2023	Editing all sections	V2.18

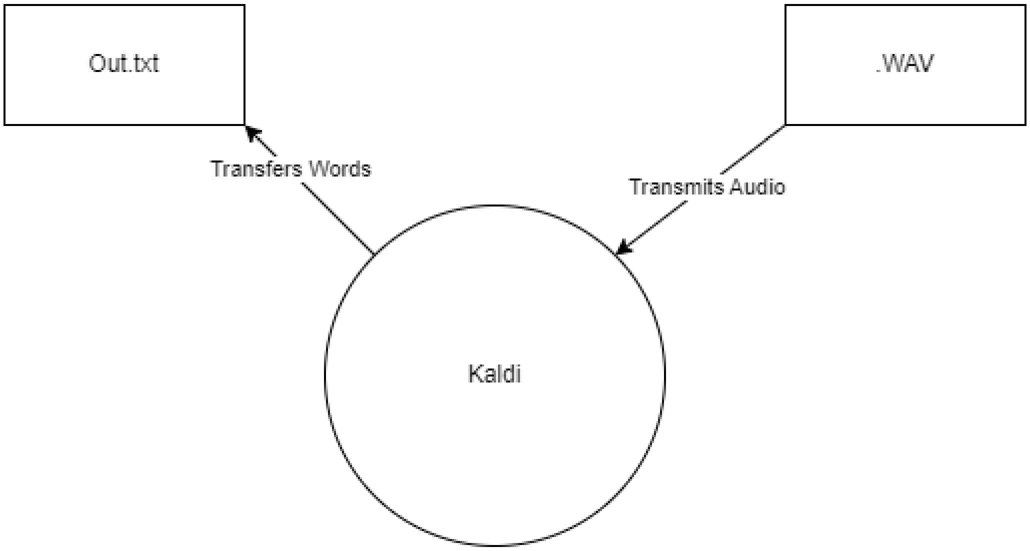
SRS

Name	Date	Reason For Changes	Version
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

Test Plan

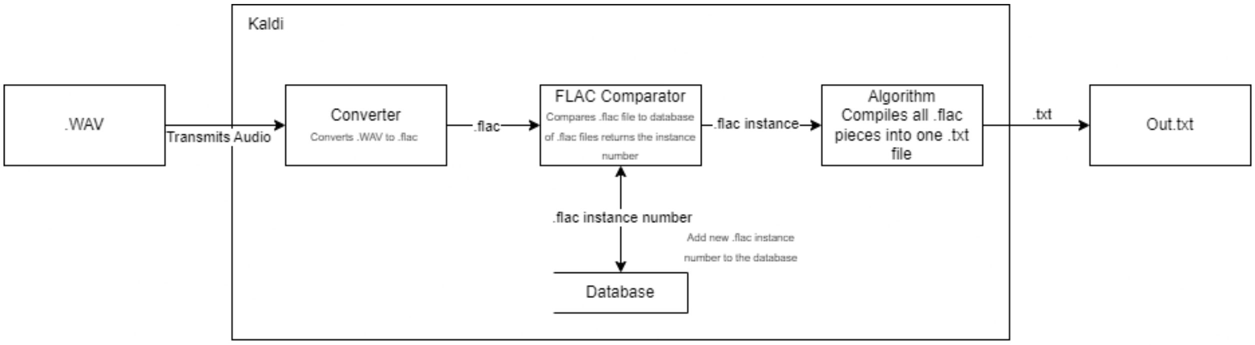
N/A

10/24



Context Diagram V0

10/25

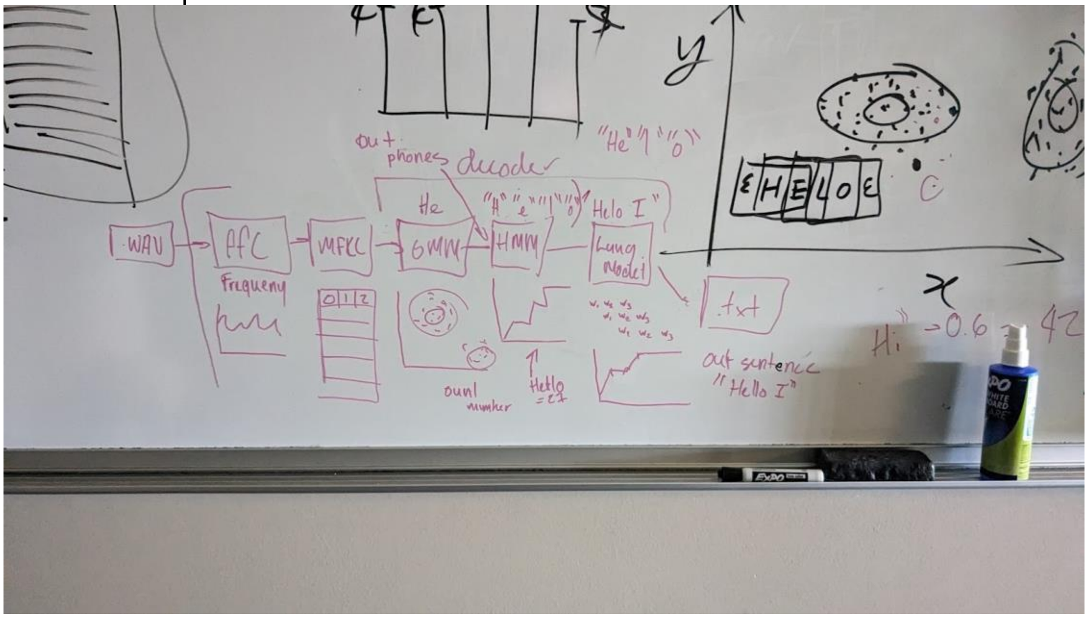

Kaldi
.WAV
Transmits Audio
Converter
Converts .WAV to .flac
.flac
FLAC Comparator
Compares .flac file to database of .flac files returns the instance number
.flac instance
Algorithm
Compiles all .flac pieces into one .txt file
.txt
Out.txt
.flac instance number
Add new .flac instance number to the database
Database

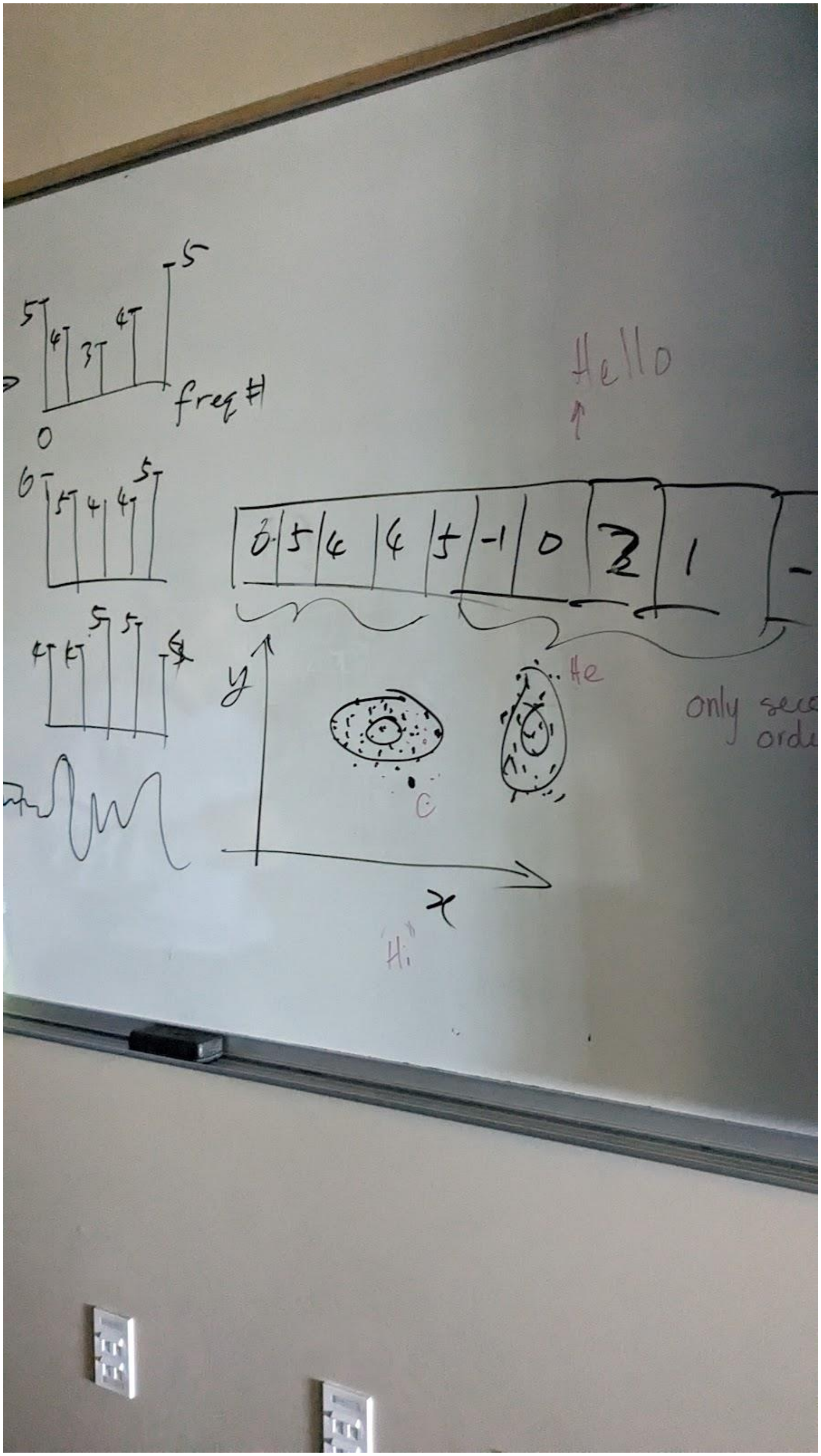
DFD V1.1

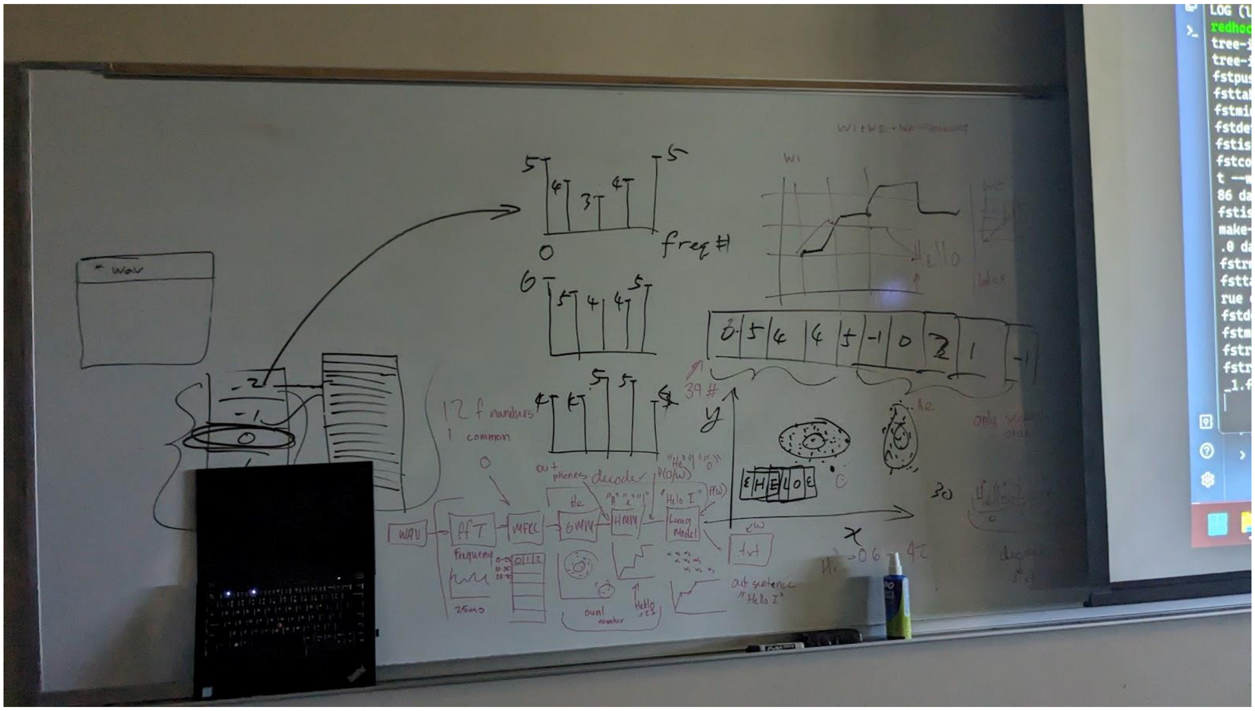
10/26

How kaldi works

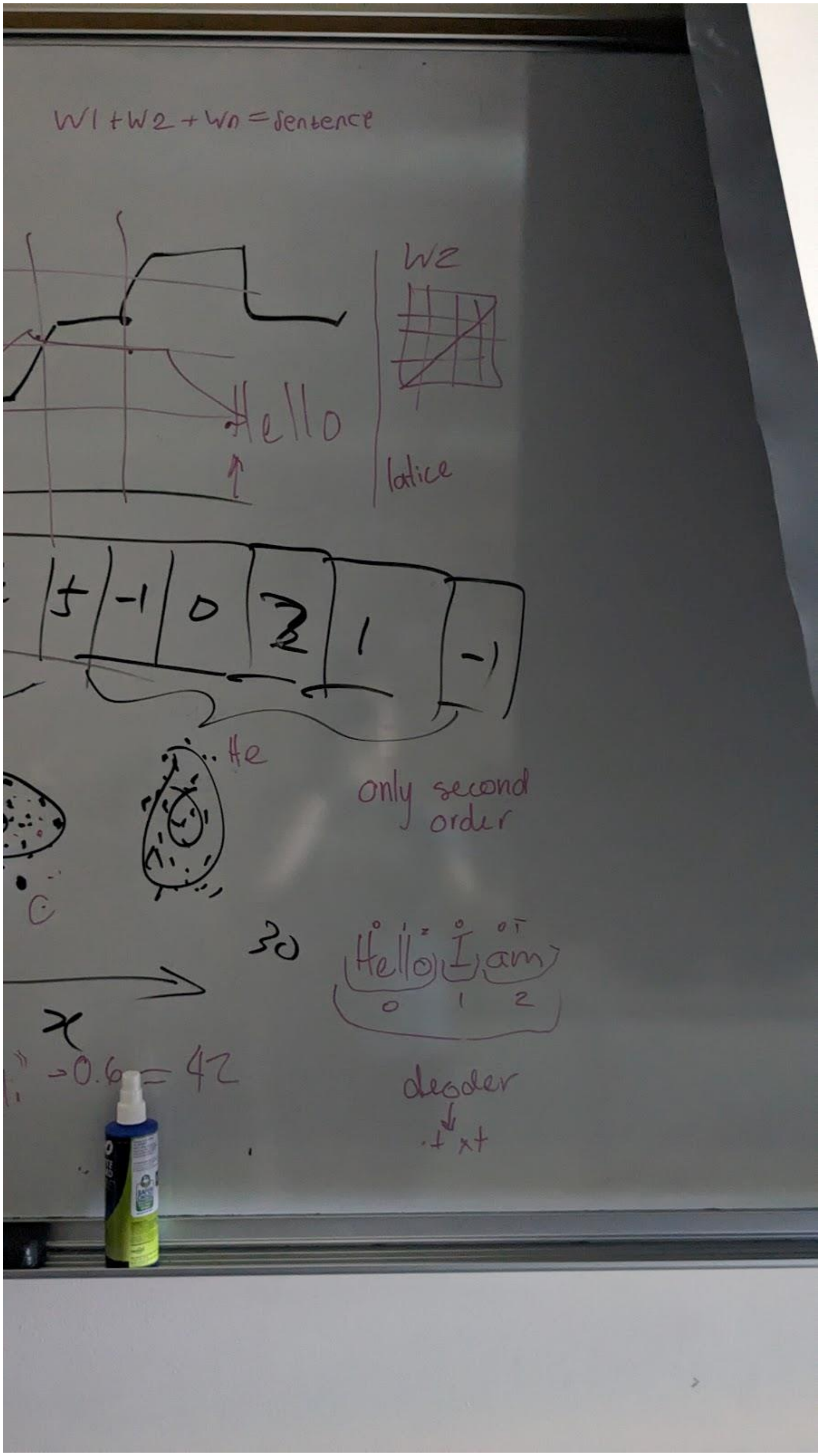
1. Input .wav (.wav OR .FLAC converted to .wav through ffmpeg)
2. .wav is split into 25ms frames every 10ms [0-25, 10-35, 20-45, n-n+25] (the frames overlap to assist in finding the beginning and end of words and to eliminate noise)
3. Frame is put into FFT to convert to frequency graph
4. MFC lines up frames into an array and compares the current frame with the frames before and after it by 2 orders [-2|-1|0|1|2].
5. Each element of the array is put into the GMM to determine the most likely phone, and returns a numerical value equivalent to one of the phones in the phone lexicon
6. The numbers from the GMM are then input into the HMM to assemble triphones out of the phones
7. The triphones are then input into another HMM to become the most likely possible word by comparing the result of the HMM to the word library
8. The word returned from the HMM is input into the language model and using the two previous words the model tries to predict the next most likely word [-2|-1|0]
9. The HMM outputs the converted sentence to the out.txt file




5
5
4
3
4
0
freq #
6
5
4
4
5
4
4
5
5
4
3 5 4 4 5 -1 0 2 1 -
Hello
y
x
He
C
Hi
only sec
ord



```
LOG (l
redhoc
tree-i
tree-i
fstpus
fsttab
fstmin
fstdet
fstis
fstco
t --w
86 da
fstis
make-
.0 da
fstrm
fstta
rue
fstde
fstm
fstr
fstr
_1.f
```


W1 + W2 + Wn = sentence
W2
Hello
latice
5
-1
0
3
1
-1
He
only second order
30
Hello I am
0 1 2
x
-0.6 = 42
deoder
txt

Requirement:

The terminal must convert .flac to .wav (kaldi asr model can only recognize .wav files)

Use Case

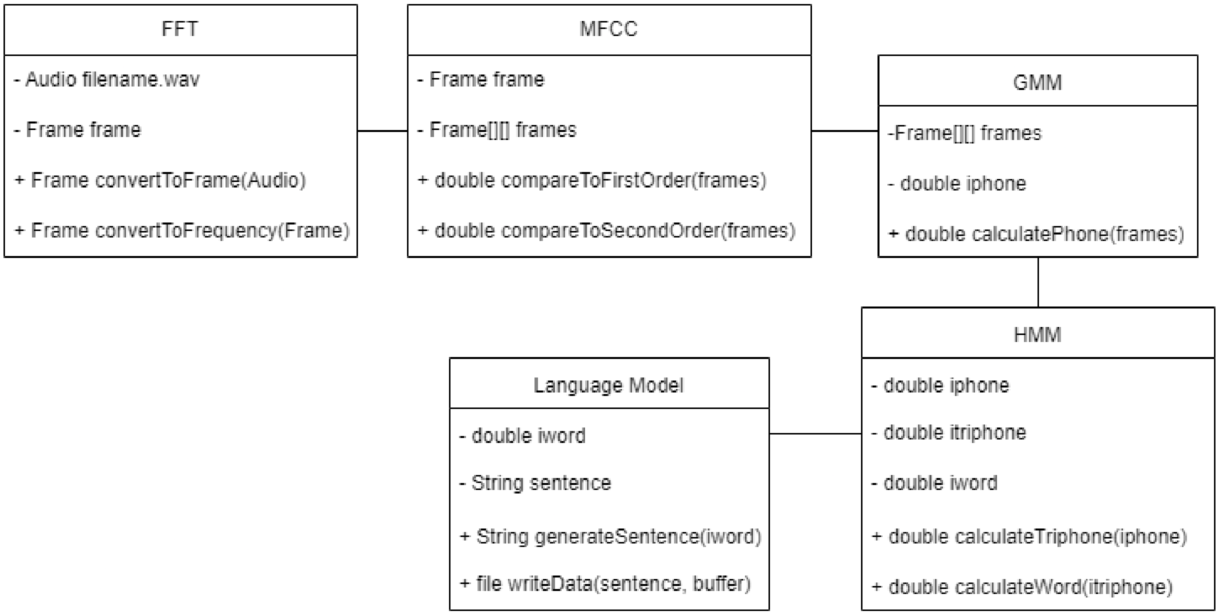
Actors

1. .Wav
2. Out.txt

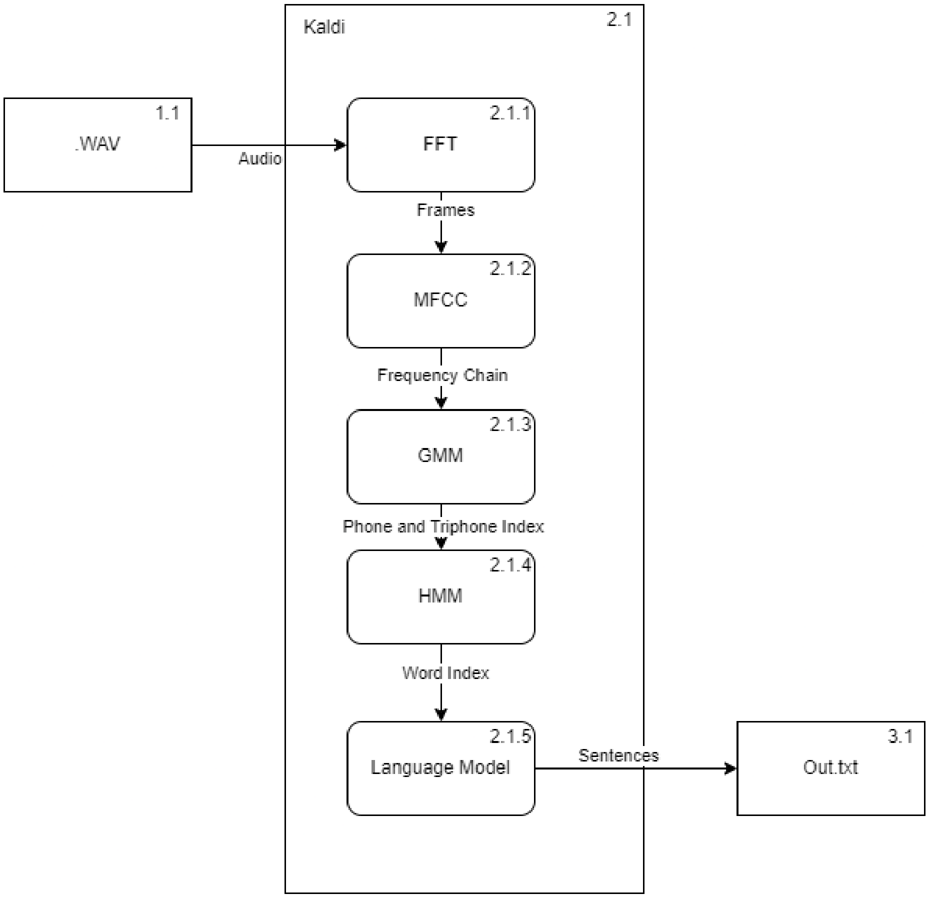
Systems

1. Phones Database
2. Triphone Database
3. Word Database

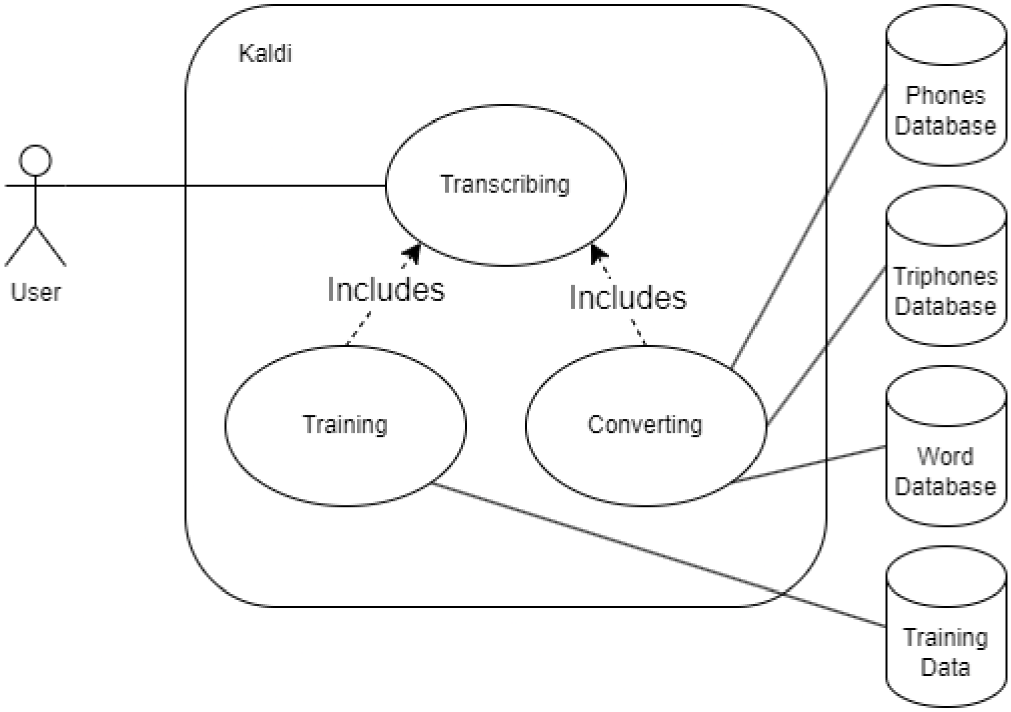
10/29


FFT
- Audio filename.wav
- Frame frame
+ Frame convertToFrame(Audio)
+ Frame convertToFrequency(Frame)
MFCC
- Frame frame
- Frame[][] frames
+ double compareToFirstOrder(frames)
+ double compareToSecondOrder(frames)
GMM
-Frame[][] frames
- double iphone
+ double calculatePhone(frames)
Language Model
- double iword
- String sentence
+ String generateSentence(iword)
+ file writeData(sentence, buffer)
HMM
- double iphone
- double itriphone
- double iword
+ double calculateTriphone(iphone)
+ double calculateWord(itriphone)

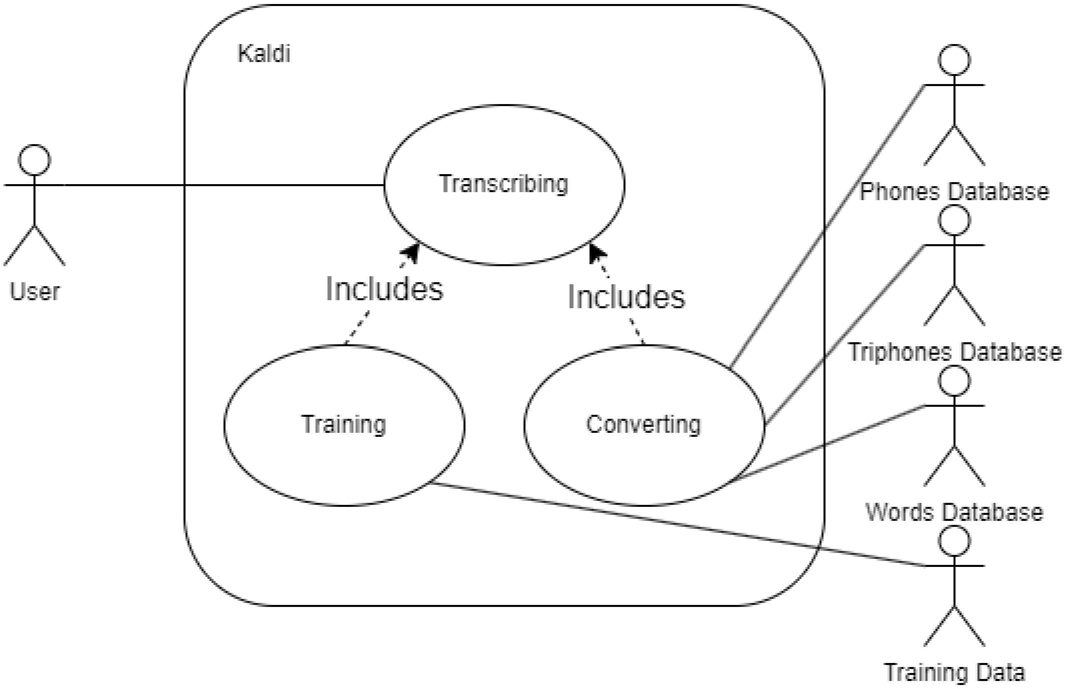
Class Model V1



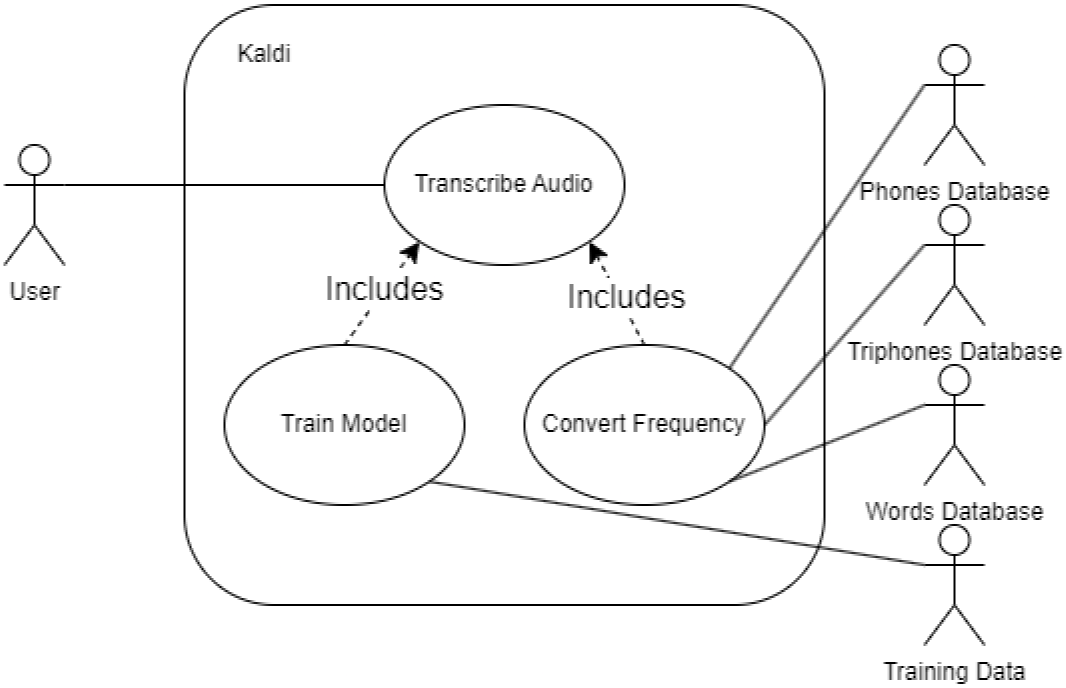
DFD V2.3



Use Case Diagram V2



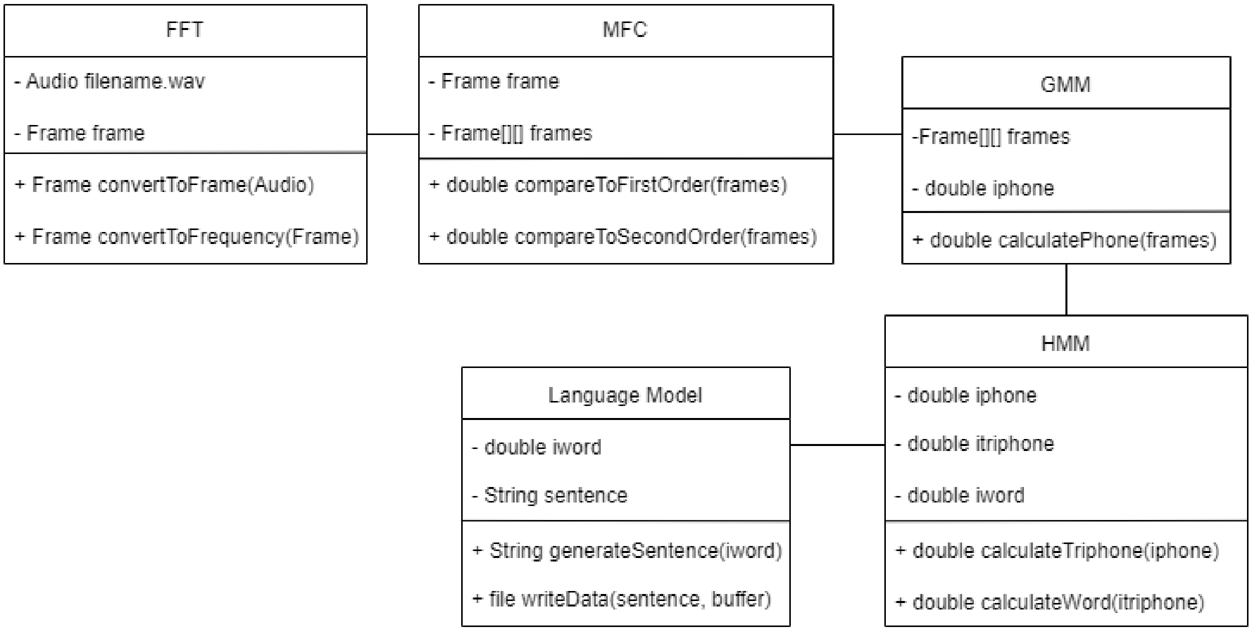
Use Case Diagram V3



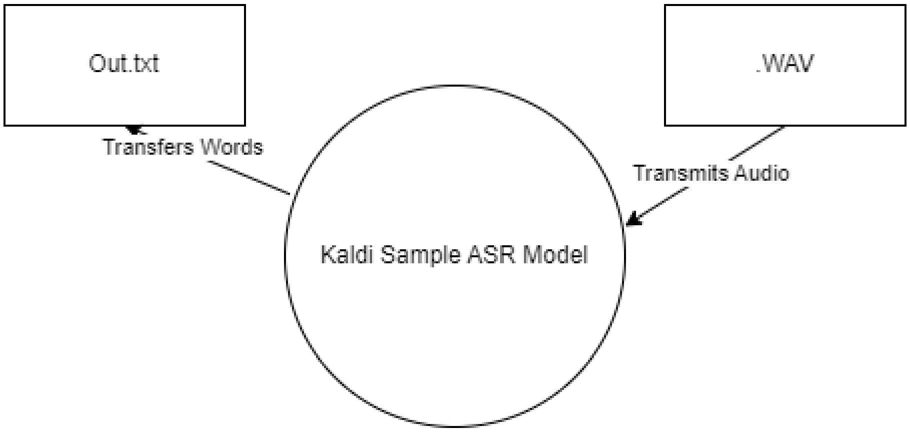
Use Case Diagram V4

Attempt was made at Collaboration Graph before discarding model entirely

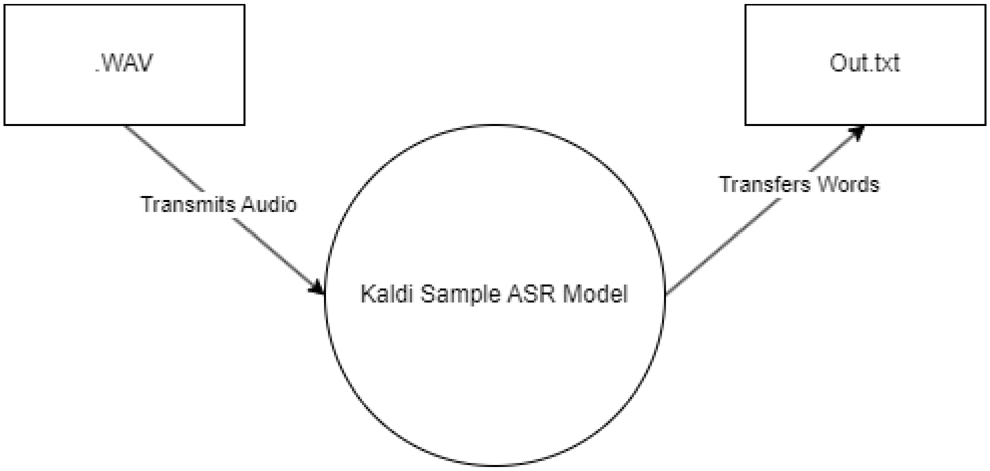
10/31


FFT
- Audio filename.wav
- Frame frame
+ Frame convertToFrame(Audio)
+ Frame convertToFrequency(Frame)
MFC
- Frame frame
- Frame[][] frames
+ double compareToFirstOrder(frames)
+ double compareToSecondOrder(frames)
GMM
-Frame[][] frames
- double iphone
+ double calculatePhone(frames)
HMM
- double iphone
- double itriphone
- double iword
+ double calculateTriphone(iphone)
+ double calculateWord(itriphone)
Language Model
- double iword
- String sentence
+ String generateSentence(iword)
+ file writeData(sentence, buffer)

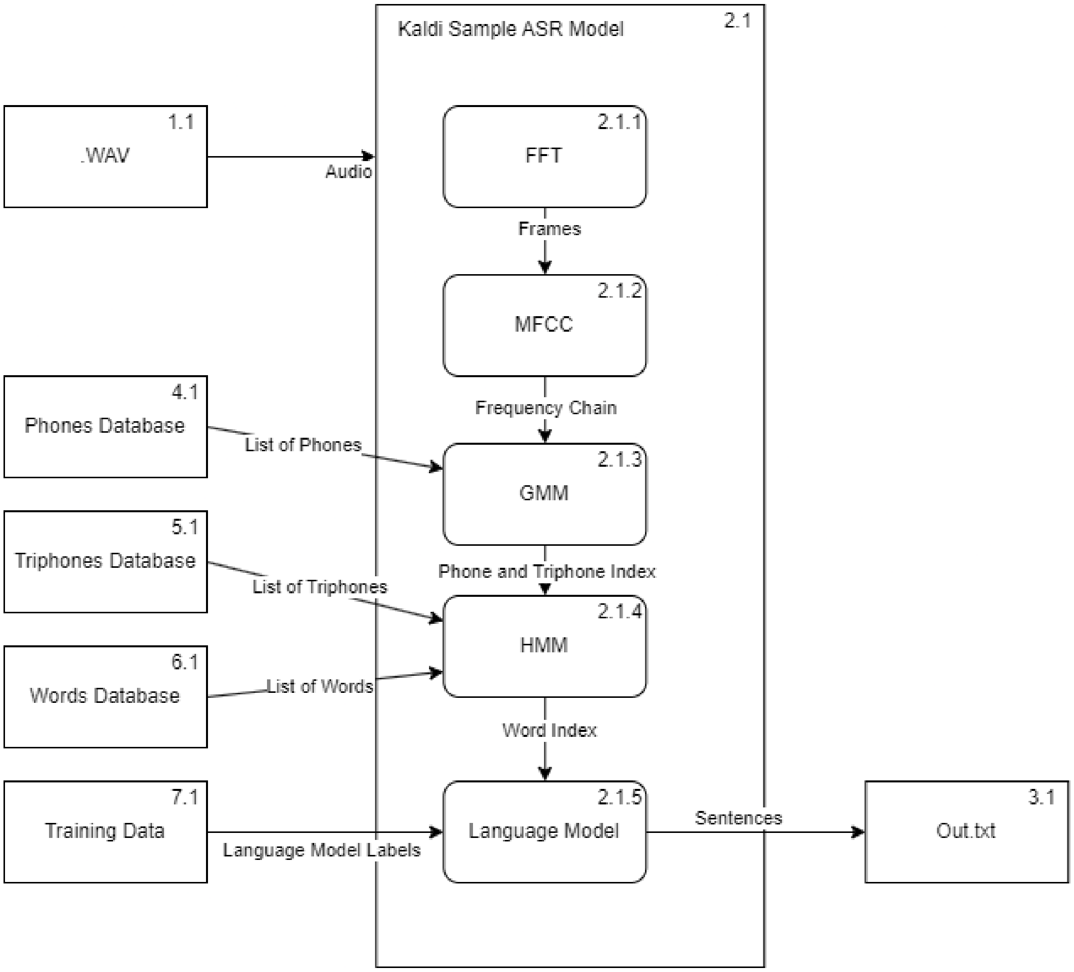
Class Diagram V1.3



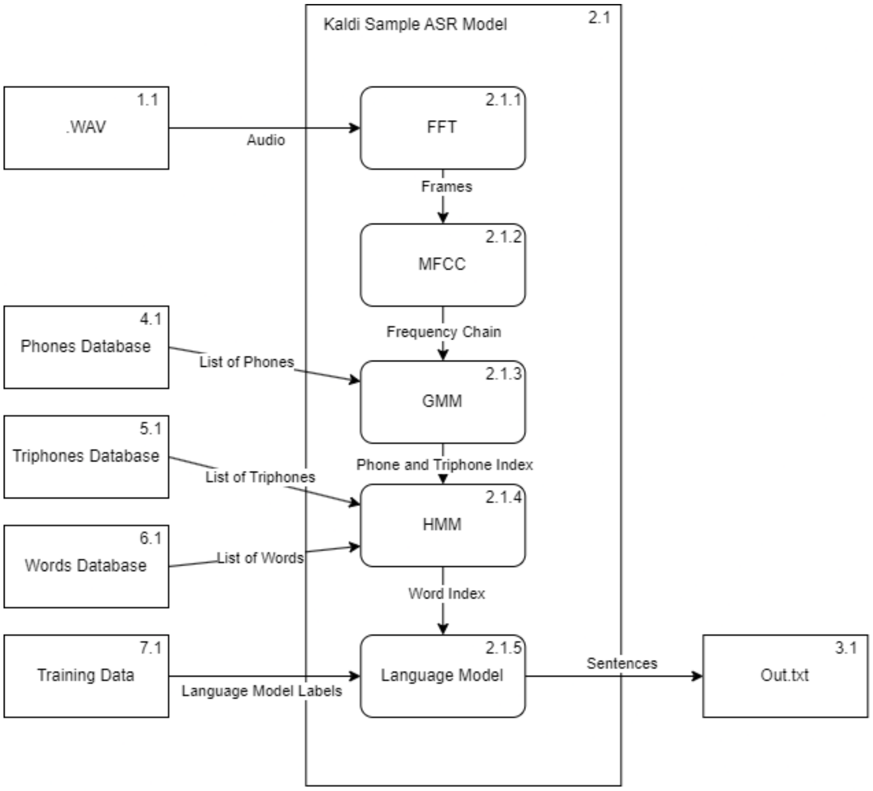
Context Diagram V1.2



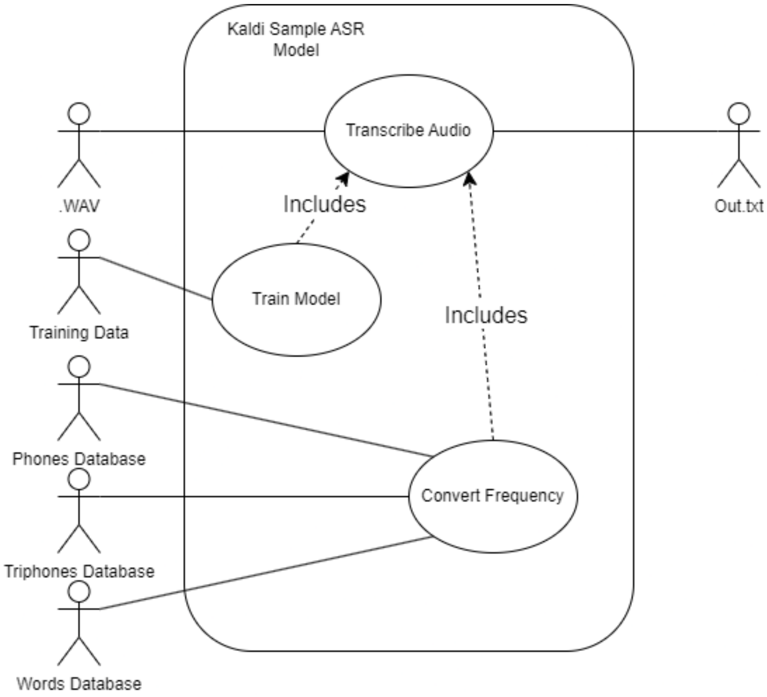
Context Diagram V2



DFD V2.4

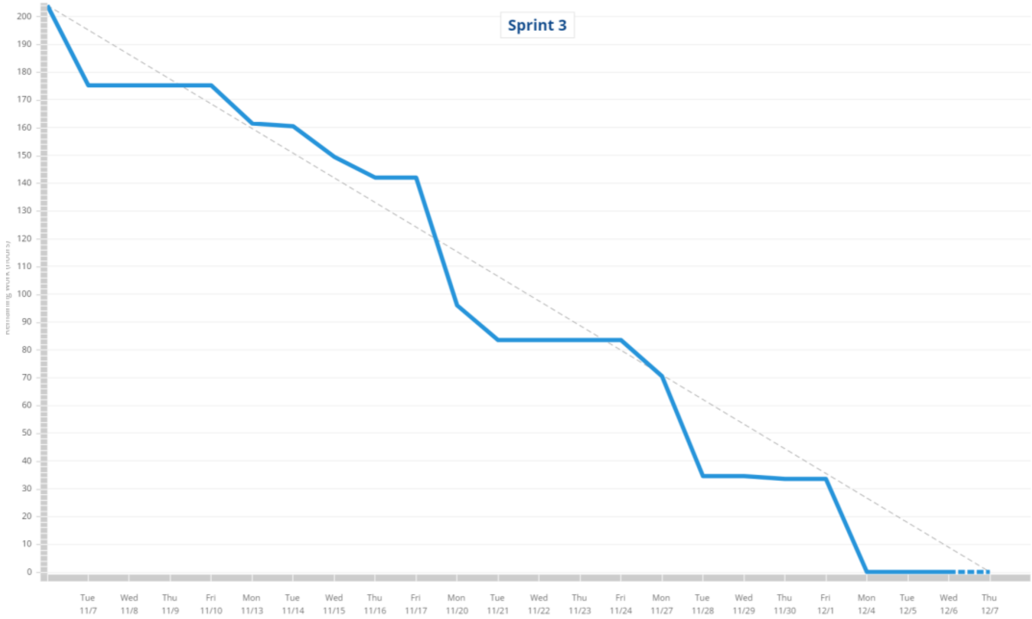


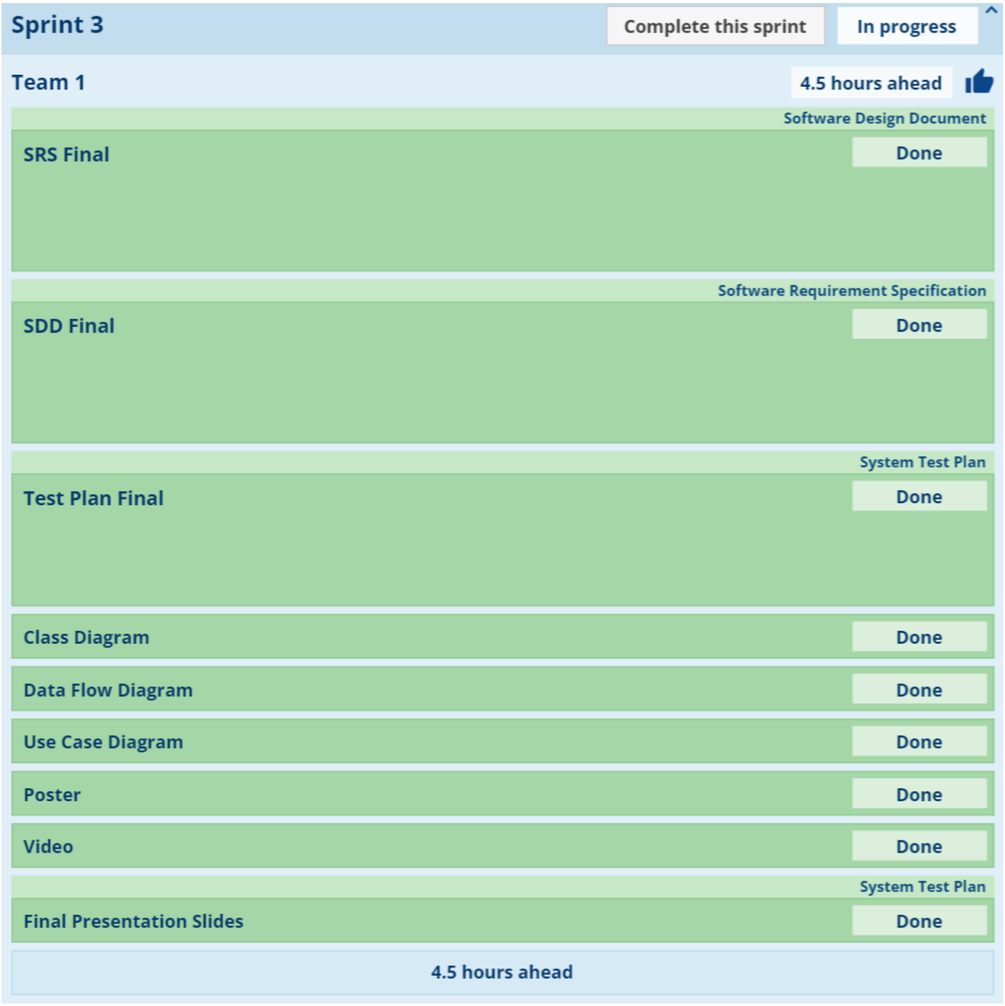
DFD V2.5



Use Case Diagram V6

Sprint 3





Sprint 3

Team 1 4.5 hours ahead

Software Design Document

SRS Final Done

Software Requirement Specification

SDD Final Done

System Test Plan

Test Plan Final Done

Class Diagram Done

Data Flow Diagram Done

Use Case Diagram Done

Poster Done

Video Done

System Test Plan

Final Presentation Slides Done

4.5 hours ahead

SDD

Name	Date	Reasons For Change	Version
Tabitha	11/05/2023	Class Diagram: 2.1	V3.1
David	11/05/2023	Class Diagram: 2.1	V3.2
Tisha	11/05/2023	Edited Section 2.2	V3.3
Adam	11/07/2023	Writing/Rewriting: 3.1, 3.2	V3.4
Tisha	11/07/2023	Writing/Rewriting 3.1	V3.5
Adam	11/07/2023	Writing/Rewriting 3.1	V3.5
Milan	11/07/2023	Editing all Sections, reformatting Table of Contents	V3.6
David	11/11/2023	Editing and Rewriting: 4.2 Updating DFD model	V3.7
Tabitha	11/11/2023	Editing and Rewriting: 4.2	V3.8
Tisha	11/11/2023	Editing 4.1	V3.9
Milan	11/11/2023	Editing all sections	V3.10
Milan	11/11/2023	Editing: 2.2	V3.11
Tisha	11/12/2023	Edited Section 4.1 (added the last Use Case)	V3.12
Tabitha	11/15/2023	Update all DFD Models Updating/Rewriting Section; 4.2 Editing/Adding: 5.1	V3.13
Tisha	11/16/2023	section 4.1 (needs to be checked)	V3.14
Tabitha	11/16/2023	Reading and Commenting Sections : 2-5 For accuracy to the current requirements Update Classes Diagram Update/Rewrite Section: 2.1	V3.15
David	11/18/2023	Editing and rewriting Use Cases Remaking Use Case Diagram V2.1.1 Making Use Case Diagram V2.2.2 Editing DFD V3.1.3, V3.2.2, V3.3.1 Rewriting and editing 4.1	V3.16
Tabitha	11/18/2023	Editing and rewriting Use Cases Remaking Use Case Diagram V2.1.1 Making Use Case Diagram V2.2.2 Editing DFD V3.1.3, V3.2.2, V3.3.1 Rewriting and editing 4.1	V3.17

Tisha	11/18/2023	Rewriting section 5.2 (needs to checked for accuracy)	V3.18
Milan	11/18/2023	Editing all sections	V3.19
David	11/19/2023	Editing 4.2, 2.1, 5.1, 5.2 Editing Class Diagram	V3.20
Tabitha	11/19/2023	Editing: 5.1, 5.2	V3.21
Milan	11/19/2023	Editing all sections	V3.22
Milan	11/20/2023	Reviewing/editing all sections	V3.23
Tabitha	11/22/2023	Section 1.2.3 and Figure 3	V3.23

SRS

Name	Date	Reason For Changes	Version
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

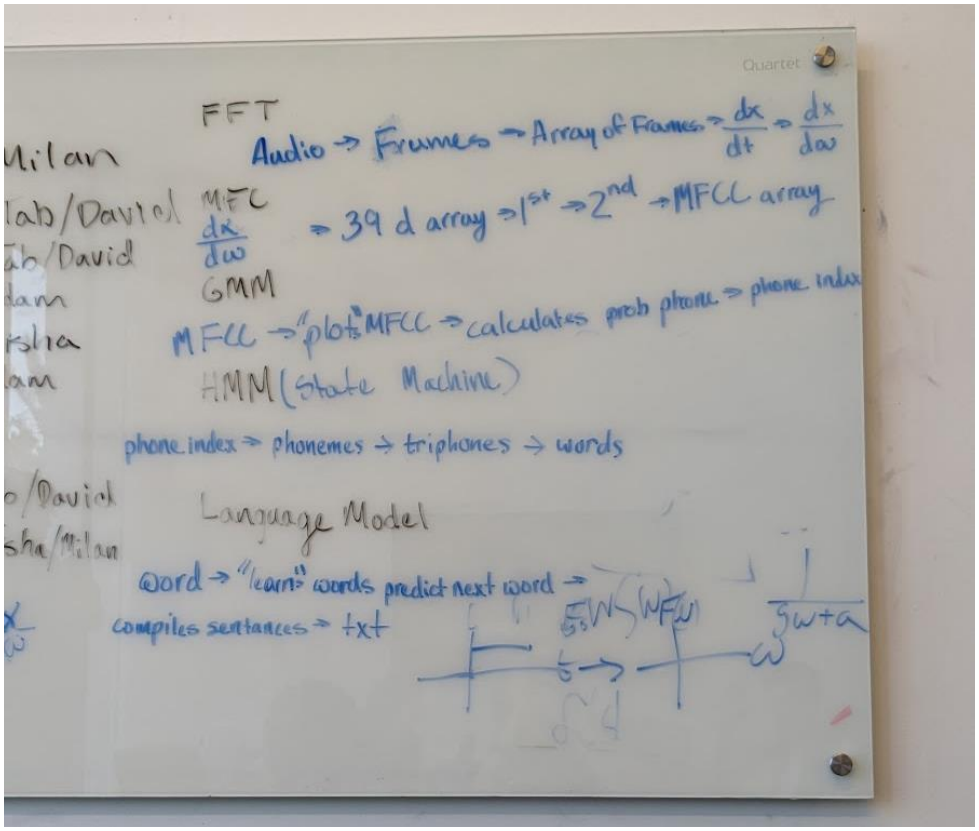
Tabitha	11/15/2023	Editing: 1.5, 2.3, 4.1, 5.1	V3.16
Tabitha	11/16/2023	Review/Edition/Commenting Section: 4 Update Class Diagram Update/Review Section: 2.3	V3.17
Tisha	11/18/2023	Rewriting section 5.2	V3.18
Tisha	11/18/2023	Edited section 5.2	V3.19
Milan	11/18/2023	Editing all sections	V3.20
David	11/19/2023	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

Test Plan

Name	Date	Reason For Changes	Version
Tabitha	11/20/2023	Added in comments	V2.1
Tisha	11/27/2023	Sections 1, 1.2	V2.2
Tabitha	11/27/2023	Sections 3.3, 3.3.1, 3.3.2, 7, 8	V2.3
Milan	11/27/2023	Section 3.4; Editing all sections	V2.4
Tisha	11/28/2023	Section 3 Editing	V2.5
David	11/28/2023	Section 3.4, 8, 4.1.1, 4.1.2, 4.1.3, 1.2, 3.1.1, 3.3.1, 3.4.1, 3.4.2	V2.6
Tabitha	11/28/2023	Section 1.1 Editing/Reading All Sections Editing 3.3.2, 4, 5	V2.7
Adam	11/28/2023	Section 2, 3.2; Editing, Writing to the sections	V2.8
Milan	11/28/2023	Editing all sections	V2.9
Tabitha	11/29/2023	Section 3.1	V2.10
Tisha	11/29/2023	Section 3.; Edited the section	V2.11
Milan	12/03/2023	Editing all sections	V2.12

Milan	12/04/2023	Editing all sections	V2.13

11/5



FFT

Milan — Audio → Frames → Array of Frames → $\frac{dx}{dt}$ → $\frac{dx}{d\omega}$

lab/David — MFC

ab/David — $\frac{dx}{d\omega}$ → 39 d array → 1st → 2nd → MFCC array

dam — GMM

sha — MFCC → "plots" MFCC → calculates prob phone → phone index

am — HMM (State Machine)

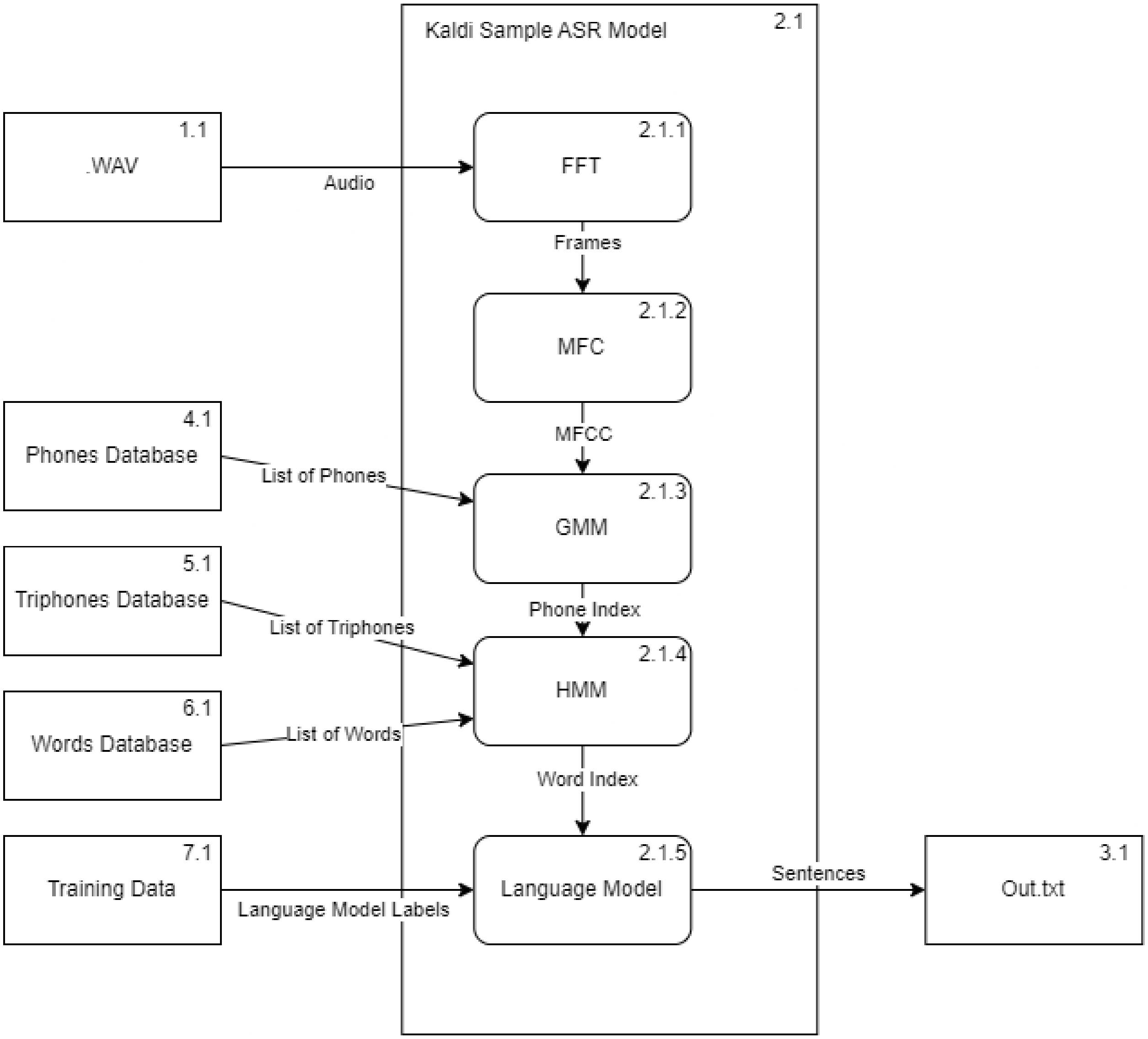
phone index → phonemes → triphones → words

o/David — Language Model

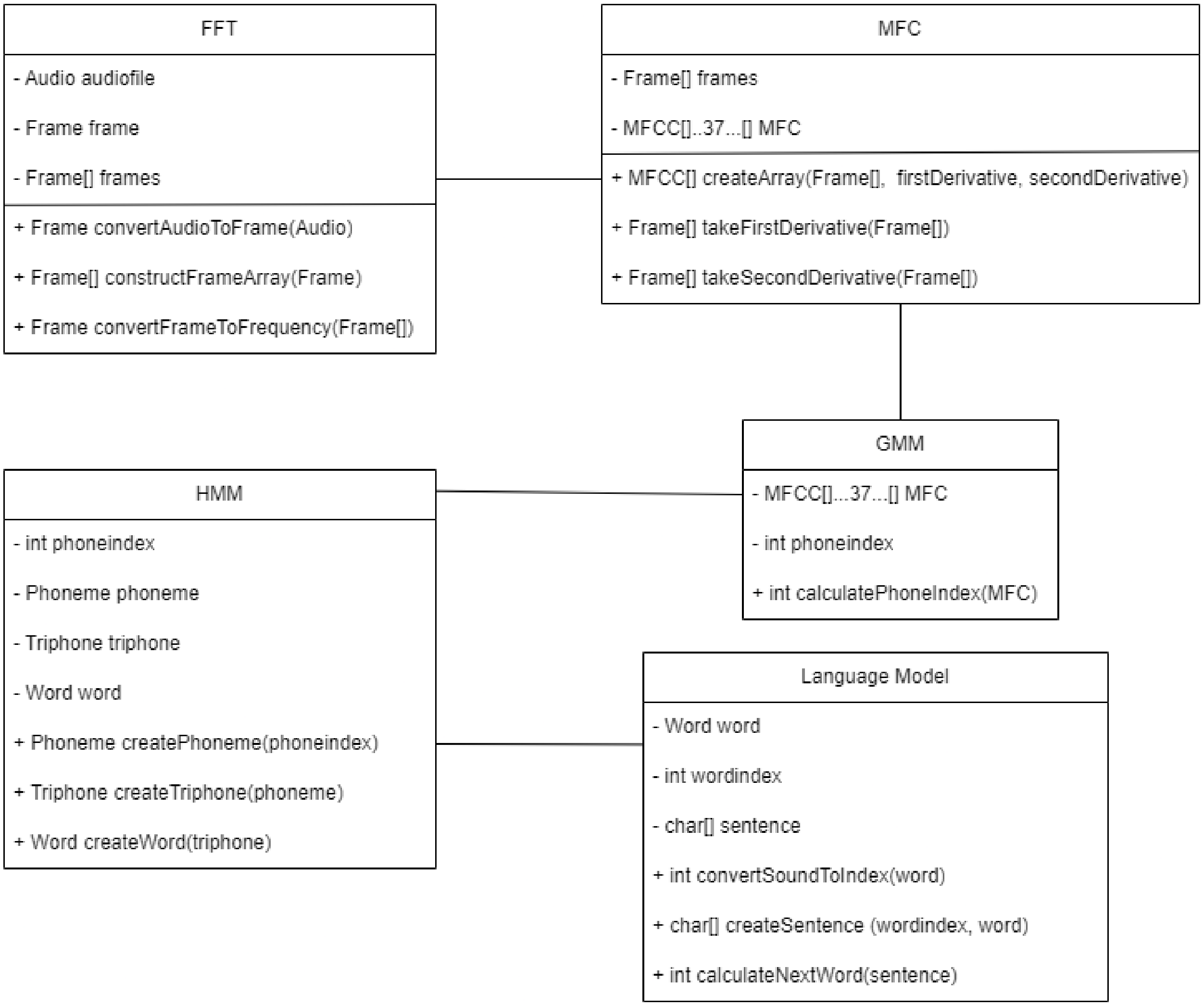
sha/Milan

word → "learns" words predict next word →

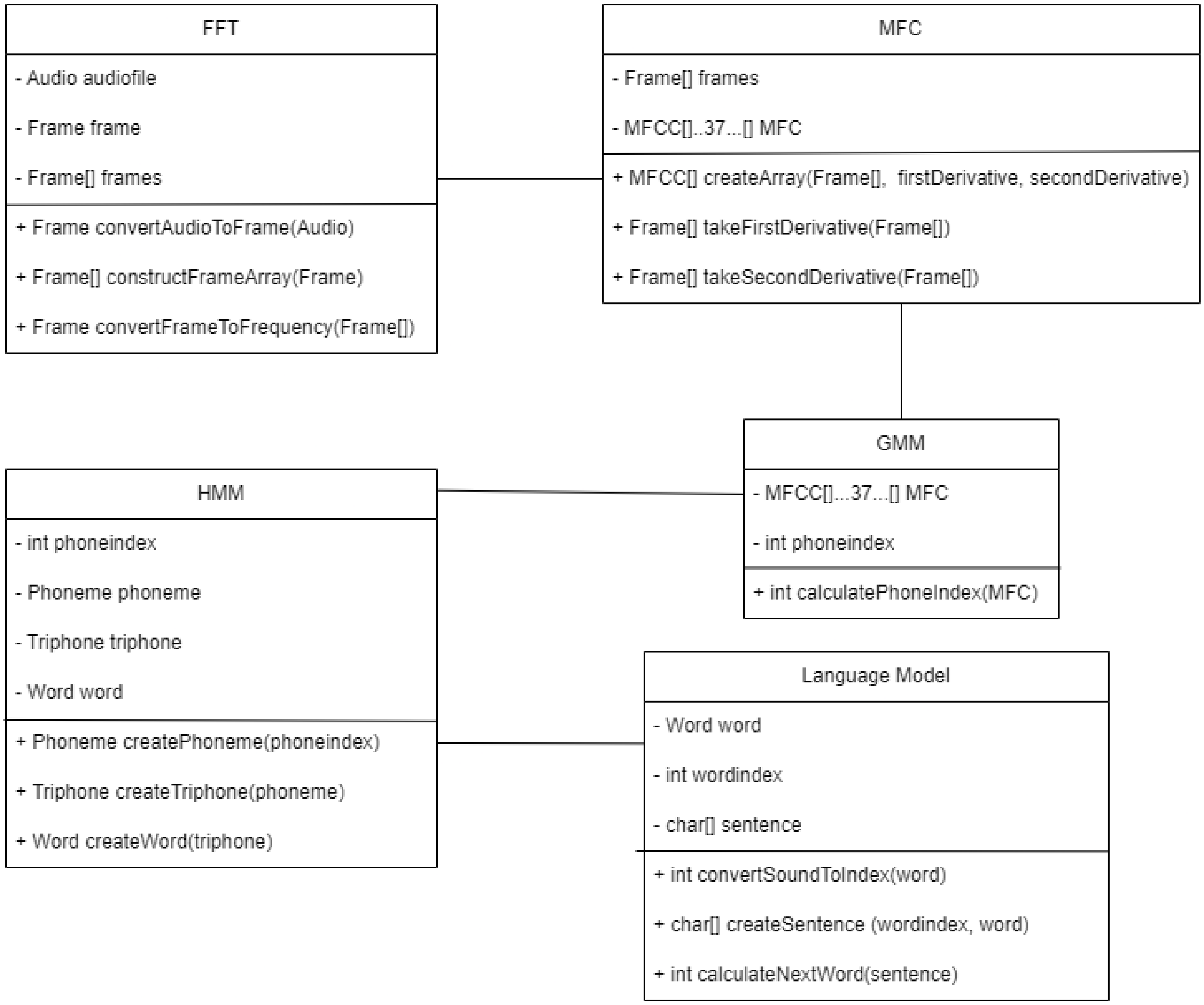
compiles sentances → txt


Kaldi Sample ASR Model
2.1
1.1
.WAV
Audio
2.1.1
FFT
Frames
2.1.2
MFC
MFCC
4.1
Phones Database
List of Phones
2.1.3
GMM
Phone Index
5.1
Triphones Database
List of Triphones
2.1.4
HMM
6.1
Words Database
List of Words
Word Index
7.1
Training Data
Language Model Labels
2.1.5
Language Model
Sentences
3.1
Out.txt

DFD V3.1

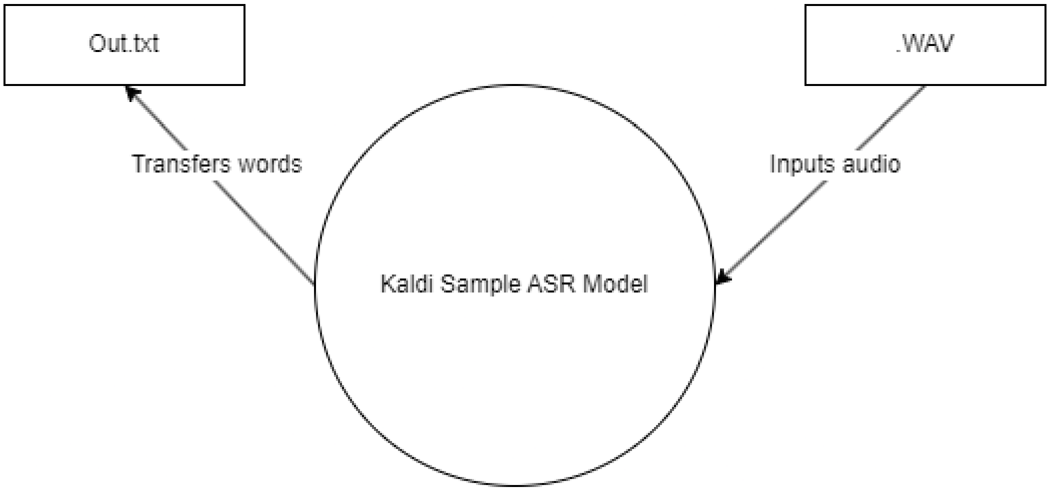

FFT
- Audio audiofile
- Frame frame
- Frame[] frames
+ Frame convertAudioToFrame(Audio)
+ Frame[] constructFrameArray(Frame)
+ Frame convertFrameToFrequency(Frame[])
MFC
- Frame[] frames
- MFCC[]..37...[] MFC
+ MFCC[] createArray(Frame[], firstDerivative, secondDerivative)
+ Frame[] takeFirstDerivative(Frame[])
+ Frame[] takeSecondDerivative(Frame[])
GMM
- MFCC[]...37...[] MFC
- int phoneindex
+ int calculatePhoneIndex(MFC)
HMM
- int phoneindex
- Phoneme phoneme
- Triphone triphone
- Word word
+ Phoneme createPhoneme(phoneindex)
+ Triphone createTriphone(phoneme)
+ Word createWord(triphone)
Language Model
- Word word
- int wordindex
- char[] sentence
+ int convertSoundToIndex(word)
+ char[] createSentence (wordindex, word)
+ int calculateNextWord(sentence)

Class Diagram V2.1

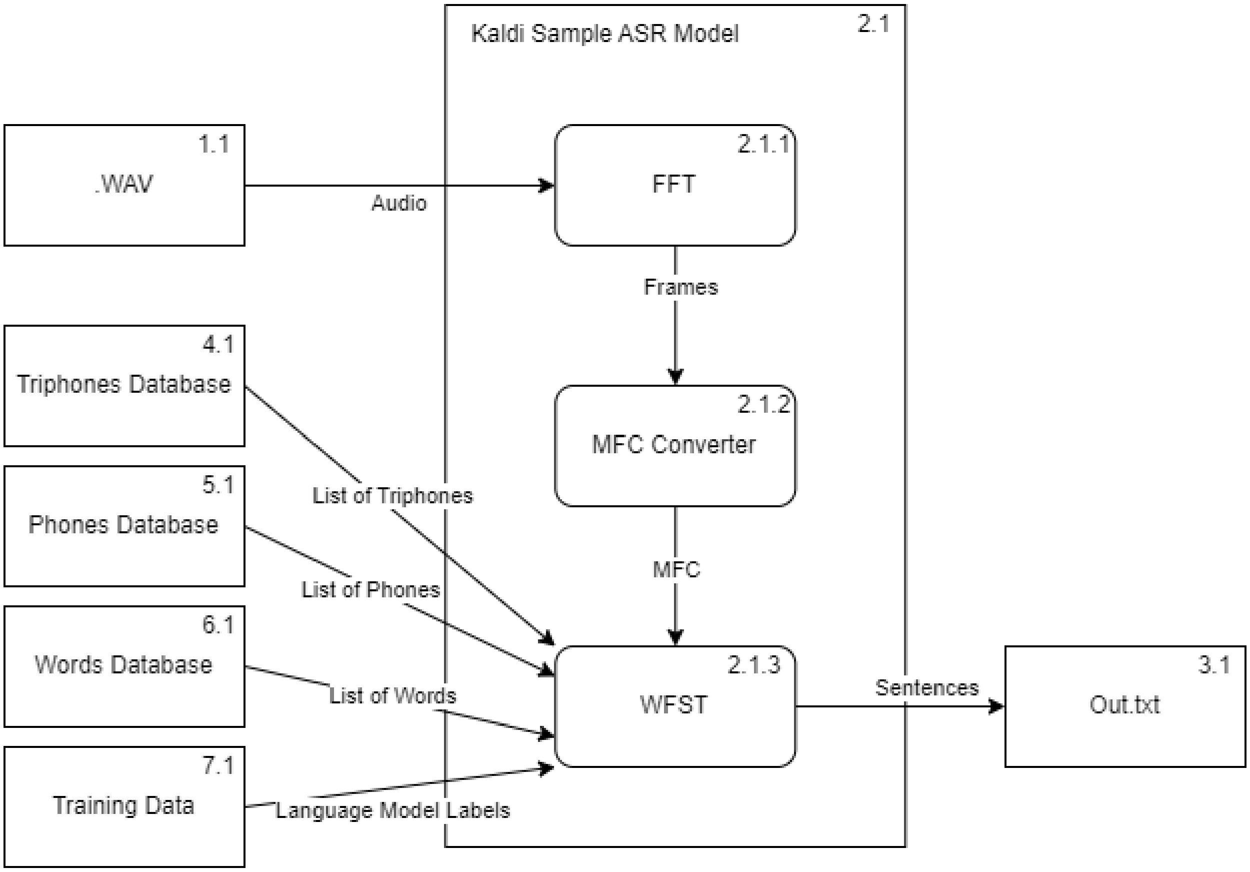

FFT
- Audio audiofile
- Frame frame
- Frame[] frames
+ Frame convertAudioToFrame(Audio)
+ Frame[] constructFrameArray(Frame)
+ Frame convertFrameToFrequency(Frame[])
MFC
- Frame[] frames
- MFCC[]..37...[] MFC
+ MFCC[] createArray(Frame[], firstDerivative, secondDerivative)
+ Frame[] takeFirstDerivative(Frame[])
+ Frame[] takeSecondDerivative(Frame[])
GMM
- MFCC[]...37...[] MFC
- int phoneindex
+ int calculatePhoneIndex(MFC)
HMM
- int phoneindex
- Phoneme phoneme
- Triphone triphone
- Word word
+ Phoneme createPhoneme(phoneindex)
+ Triphone createTriphone(phoneme)
+ Word createWord(triphone)
Language Model
- Word word
- int wordindex
- char[] sentence
+ int convertSoundToIndex(word)
+ char[] createSentence (wordindex, word)
+ int calculateNextWord(sentence)

Class Diagram V2.2

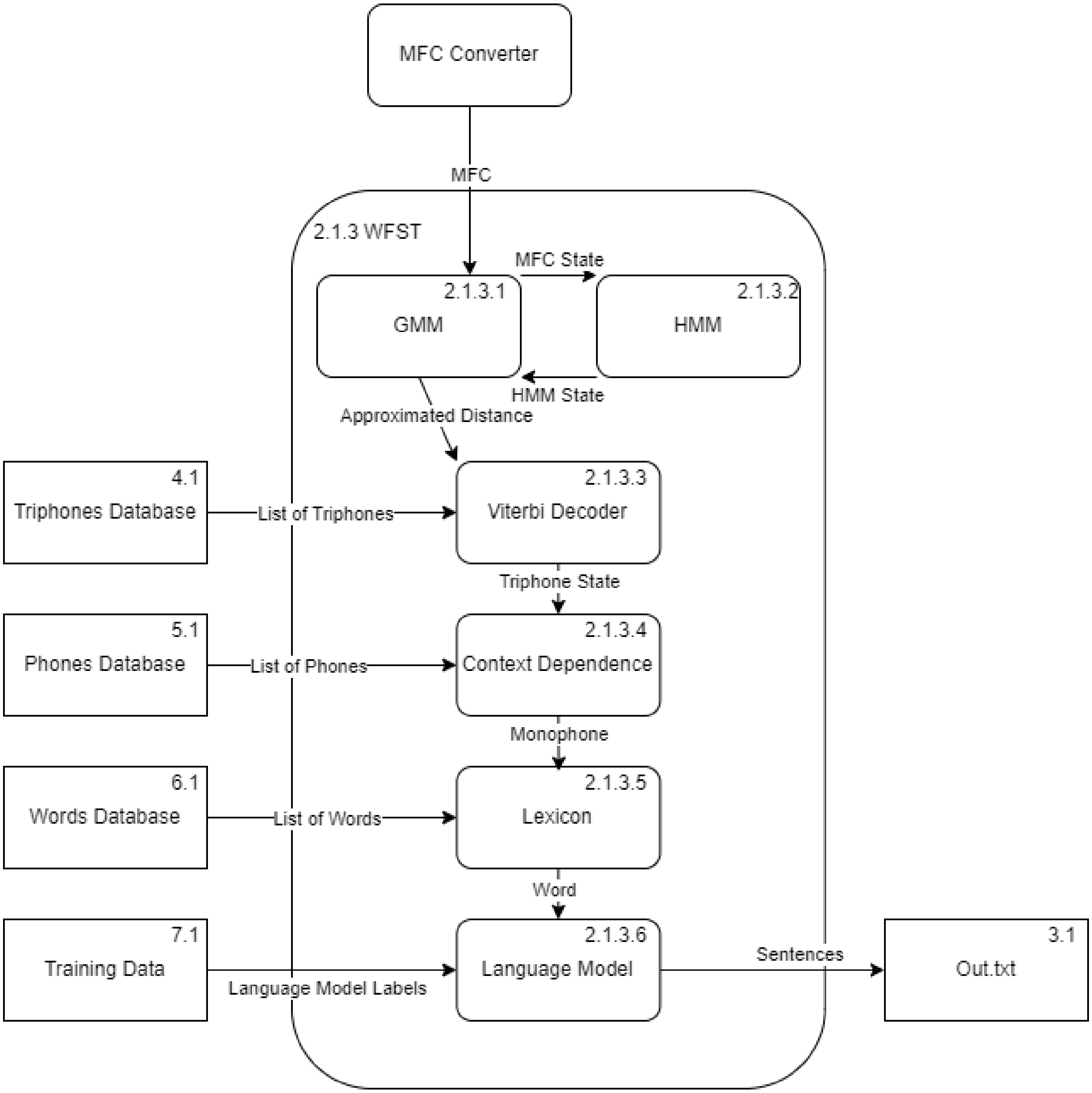
11/11



Context Diagram V3

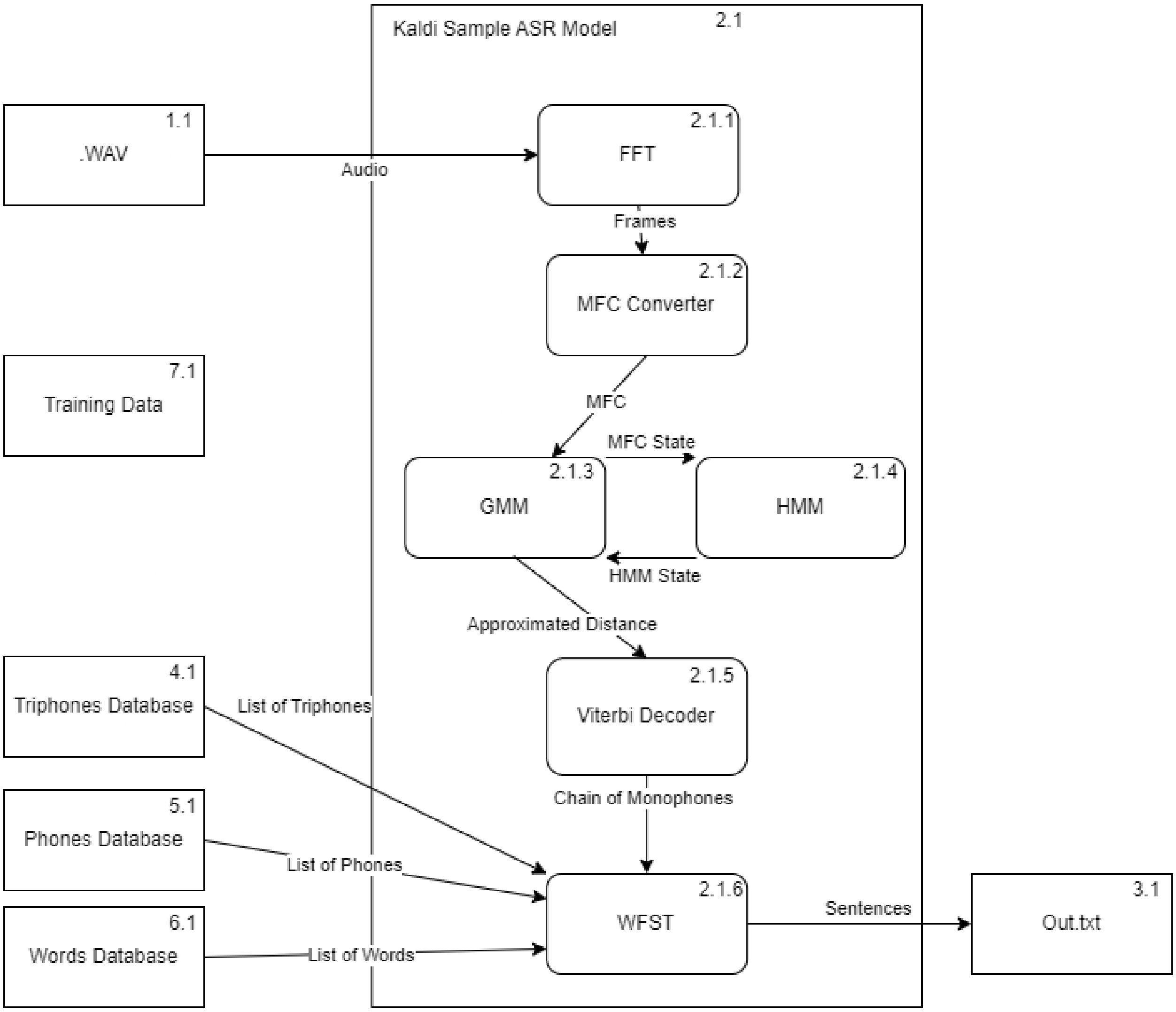


DFD V4.1

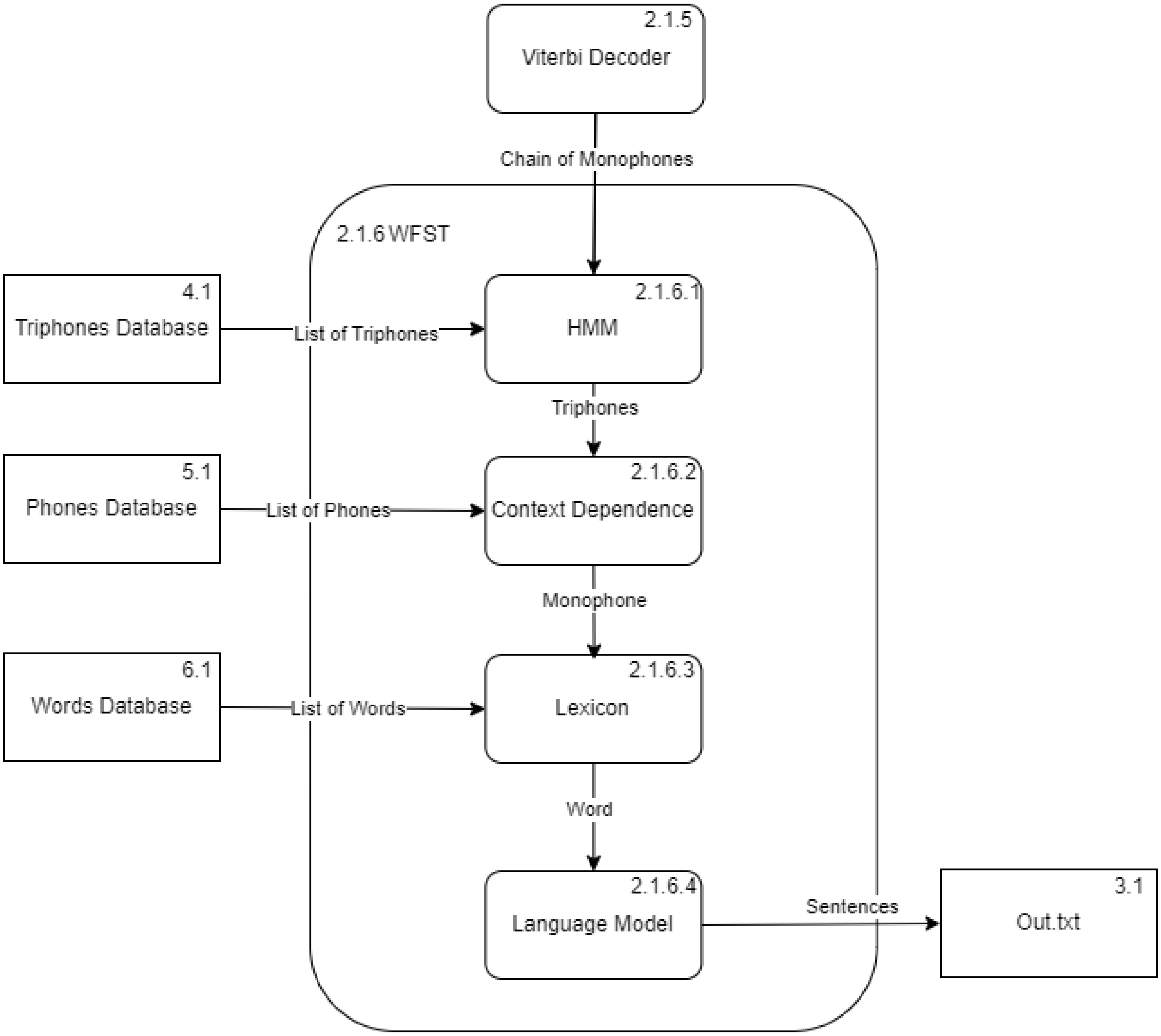

MFC Converter
MFC
2.1.3 WFST
MFC State
2.1.3.1
GMM
2.1.3.2
HMM
HMM State
Approximated Distance
4.1
Triphones Database
List of Triphones
2.1.3.3
Viterbi Decoder
Triphone State
5.1
Phones Database
List of Phones
2.1.3.4
Context Dependence
Monophone
6.1
Words Database
List of Words
2.1.3.5
Lexicon
Word
7.1
Training Data
Language Model Labels
2.1.3.6
Language Model
Sentences
3.1
Out.txt

DFD V4.1.1

11/15

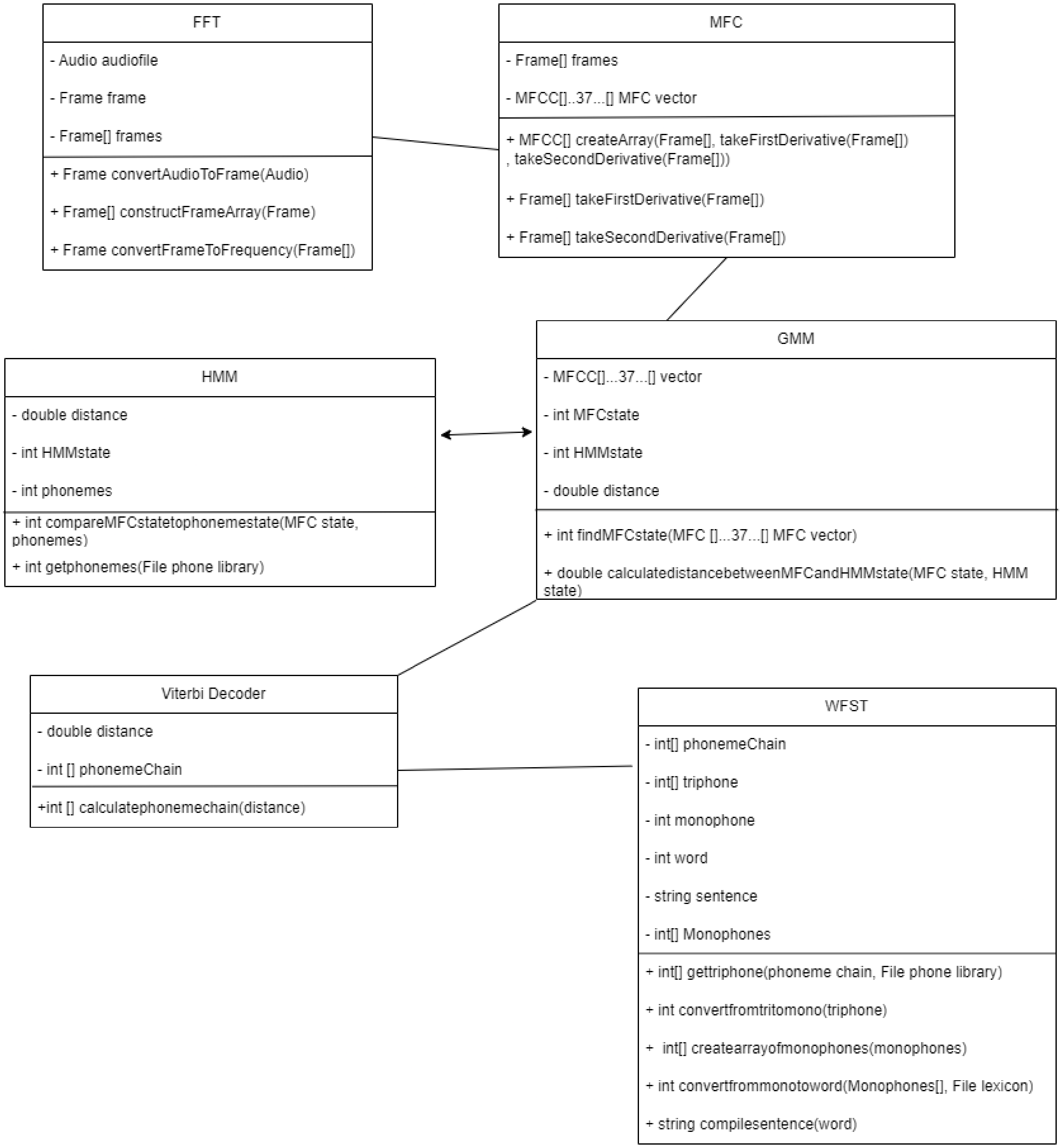

Kaldi Sample ASR Model
2.1
1.1
.WAV
Audio
2.1.1
FFT
Frames
2.1.2
MFC Converter
7.1
Training Data
MFC
MFC State
2.1.3
GMM
2.1.4
HMM
HMM State
Approximated Distance
4.1
Triphones Database
List of Triphones
2.1.5
Viterbi Decoder
Chain of Monophones
5.1
Phones Database
List of Phones
2.1.6
WFST
Sentences
3.1
Out.txt
6.1
Words Database
List of Words

DFD V4.2

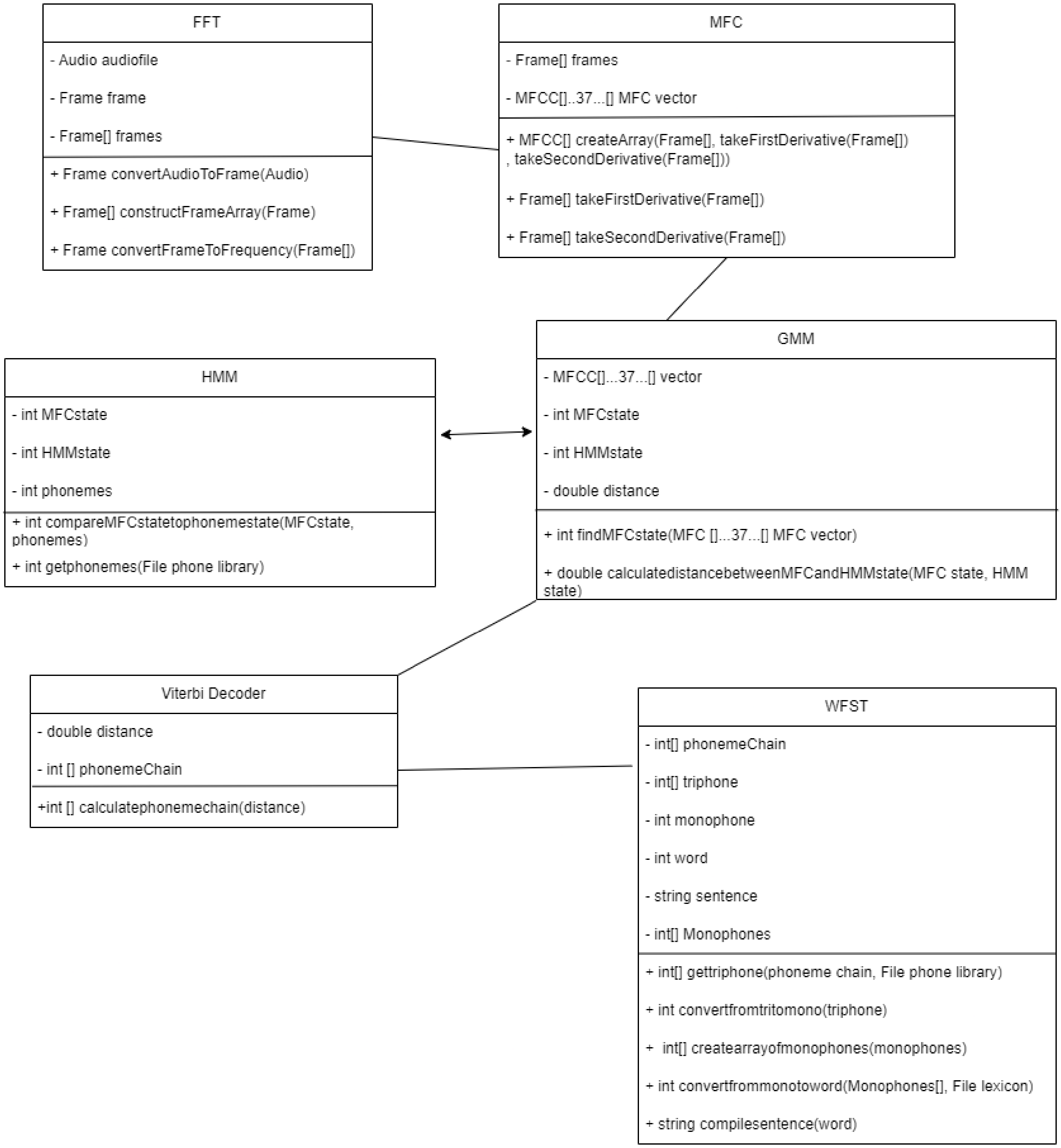

2.1.5
Viterbi Decoder
Chain of Monophones
2.1.6 WFST
4.1
Triphones Database
List of Triphones
2.1.6.1
HMM
Triphones
5.1
Phones Database
List of Phones
2.1.6.2
Context Dependence
Monophone
6.1
Words Database
List of Words
2.1.6.3
Lexicon
Word
2.1.6.4
Language Model
Sentences
3.1
Out.txt

DFD V4.2.1

11/16

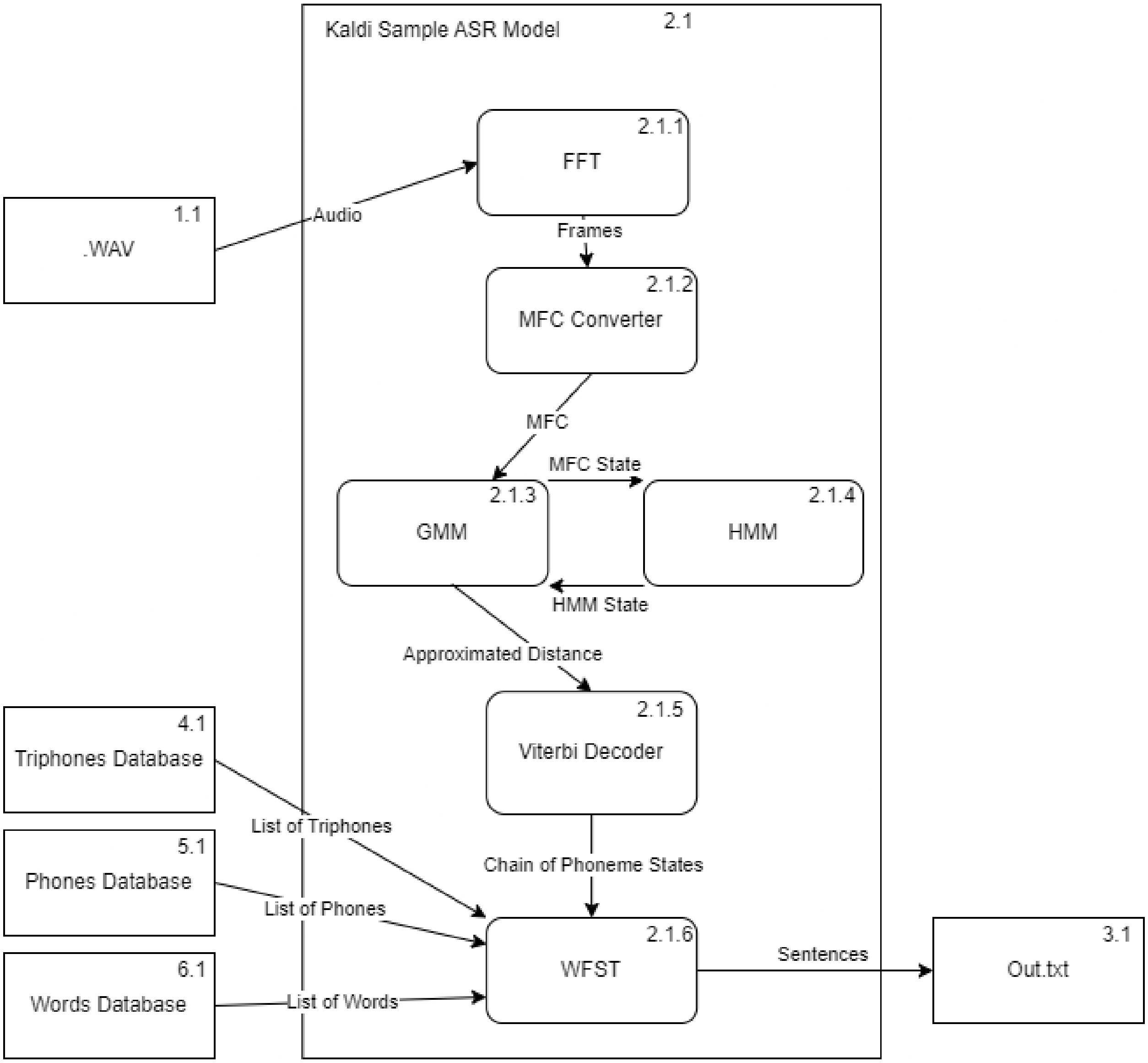

FFT
- Audio audiofile
- Frame frame
- Frame[] frames
+ Frame convertAudioToFrame(Audio)
+ Frame[] constructFrameArray(Frame)
+ Frame convertFrameToFrequency(Frame[])
MFC
- Frame[] frames
- MFCC[]..37...[] MFC vector
+ MFCC[] createArray(Frame[], takeFirstDerivative(Frame[]) , takeSecondDerivative(Frame[]))
+ Frame[] takeFirstDerivative(Frame[])
+ Frame[] takeSecondDerivative(Frame[])
HMM
- double distance
- int HMMstate
- int phonemes
+ int compareMFCstatetophonemestate(MFC state, phonemes)
+ int getphonemes(File phone library)
GMM
- MFCC[]...37...[] vector
- int MFCstate
- int HMMstate
- double distance
+ int findMFCstate(MFC []...37...[] MFC vector)
+ double calculatedistancebetweenMFCandHMMstate(MFC state, HMM state)
Viterbi Decoder
- double distance
- int [] phonemeChain
+int [] calculatephonemechain(distance)
WFST
- int[] phonemeChain
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(phoneme chain, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)
+ string compilesentence(word)

Class Diagram V3.1

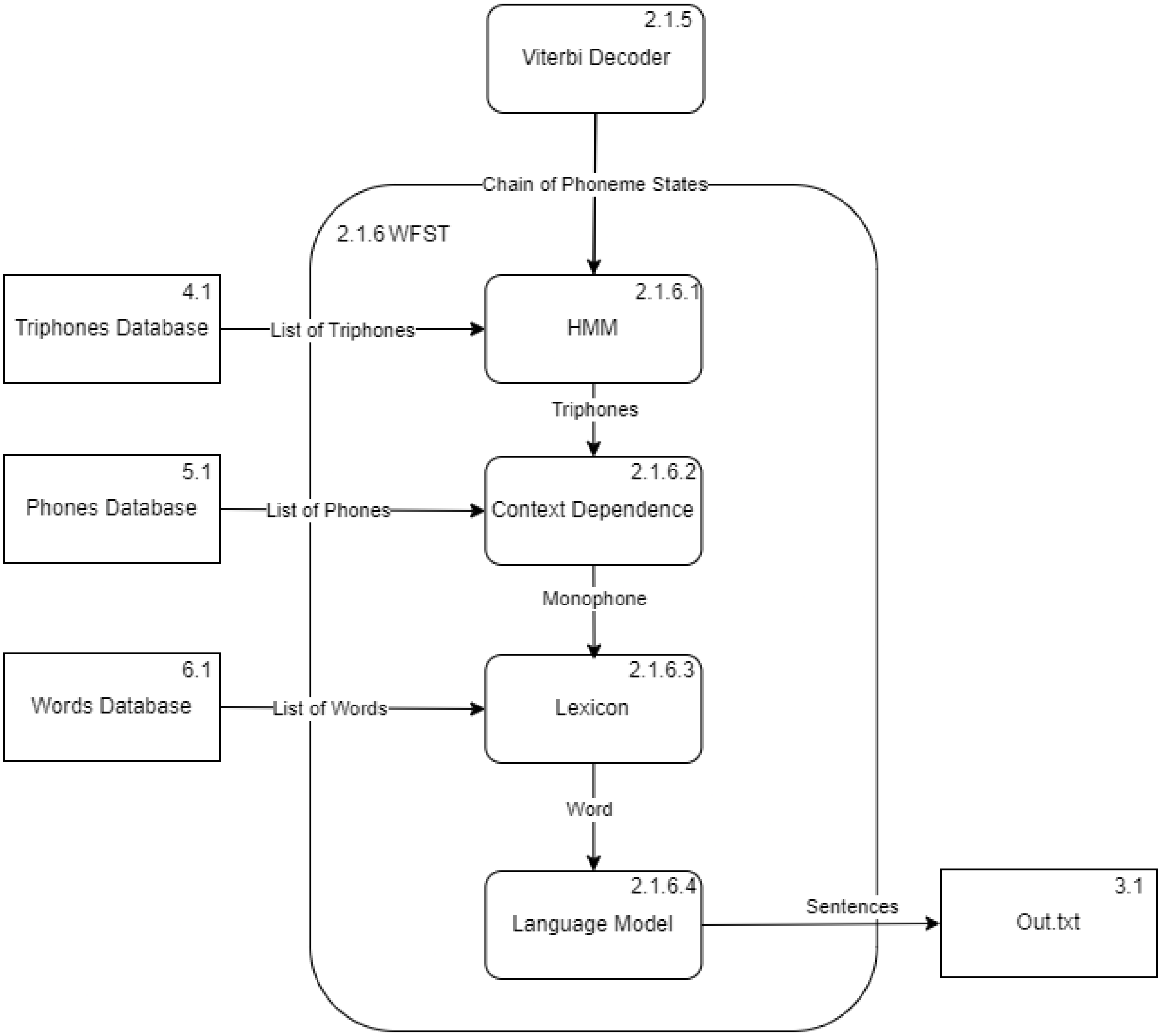

FFT
- Audio audiofile
- Frame frame
- Frame[] frames
+ Frame convertAudioToFrame(Audio)
+ Frame[] constructFrameArray(Frame)
+ Frame convertFrameToFrequency(Frame[])
MFC
- Frame[] frames
- MFCC[]..37...[] MFC vector
+ MFCC[] createArray(Frame[], takeFirstDerivative(Frame[])
, takeSecondDerivative(Frame[]))
+ Frame[] takeFirstDerivative(Frame[])
+ Frame[] takeSecondDerivative(Frame[])
HMM
- int MFCstate
- int HMMstate
- int phonemes
+ int compareMFCstatetophonemestate(MFCstate, phonemes)
+ int getphonemes(File phone library)
GMM
- MFCC[]...37...[] vector
- int MFCstate
- int HMMstate
- double distance
+ int findMFCstate(MFC []...37...[] MFC vector)
+ double calculatedistancebetweenMFCandHMMstate(MFC state, HMM state)
Viterbi Decoder
- double distance
- int [] phonemeChain
+int [] calculatephonemechain(distance)
WFST
- int[] phonemeChain
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(phoneme chain, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)
+ string compilesentence(word)

Class Diagram V3.2

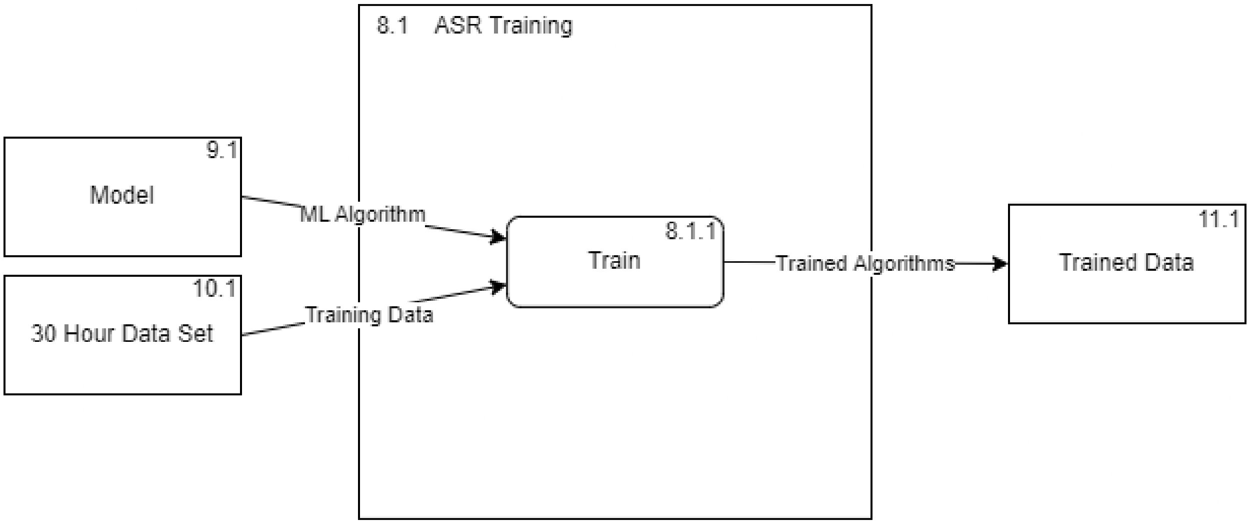
11/18



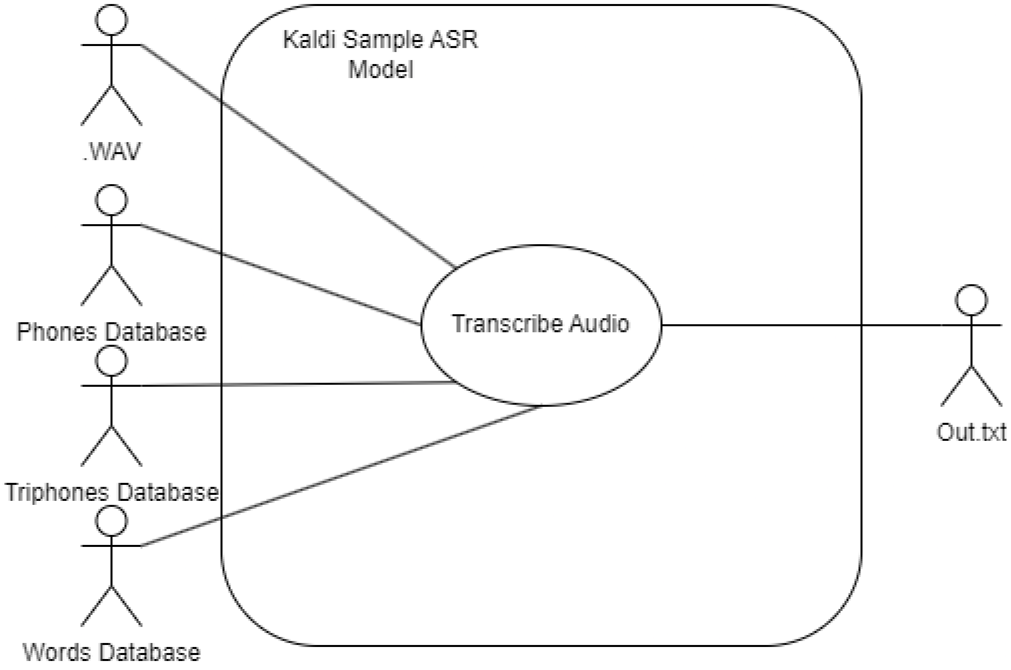
DFD V3.1.3


2.1.5
Viterbi Decoder
Chain of Phoneme States
2.1.6 WFST
4.1
Triphones Database
List of Triphones
2.1.6.1
HMM
Triphones
5.1
Phones Database
List of Phones
2.1.6.2
Context Dependence
Monophone
6.1
Words Database
List of Words
2.1.6.3
Lexicon
Word
2.1.6.4
Language Model
Sentences
3.1
Out.txt

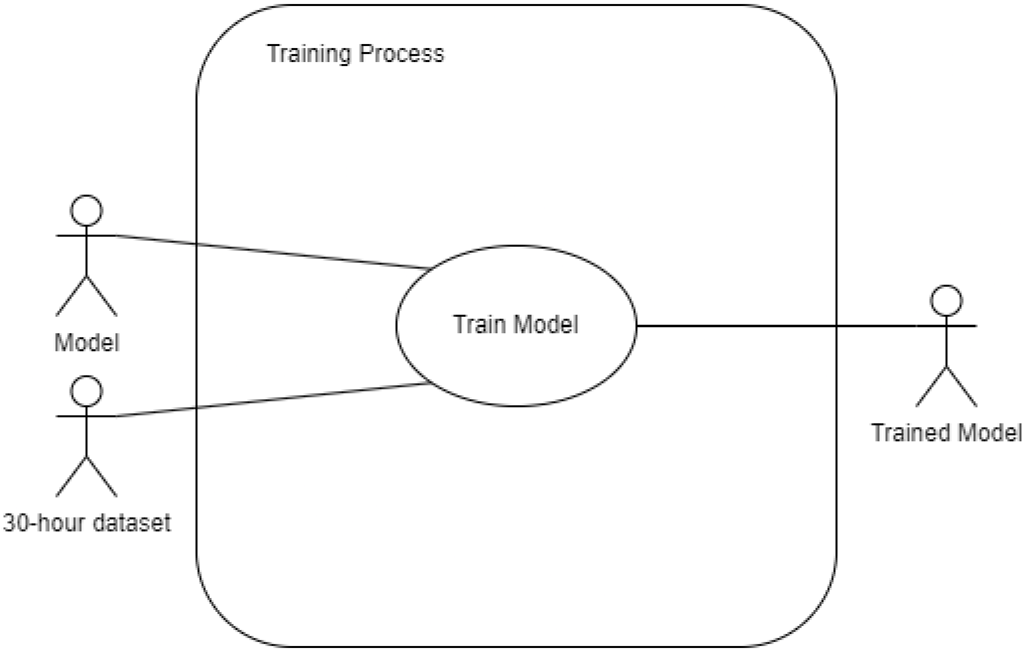
DFD V3.2.2



DFD 3.3.1

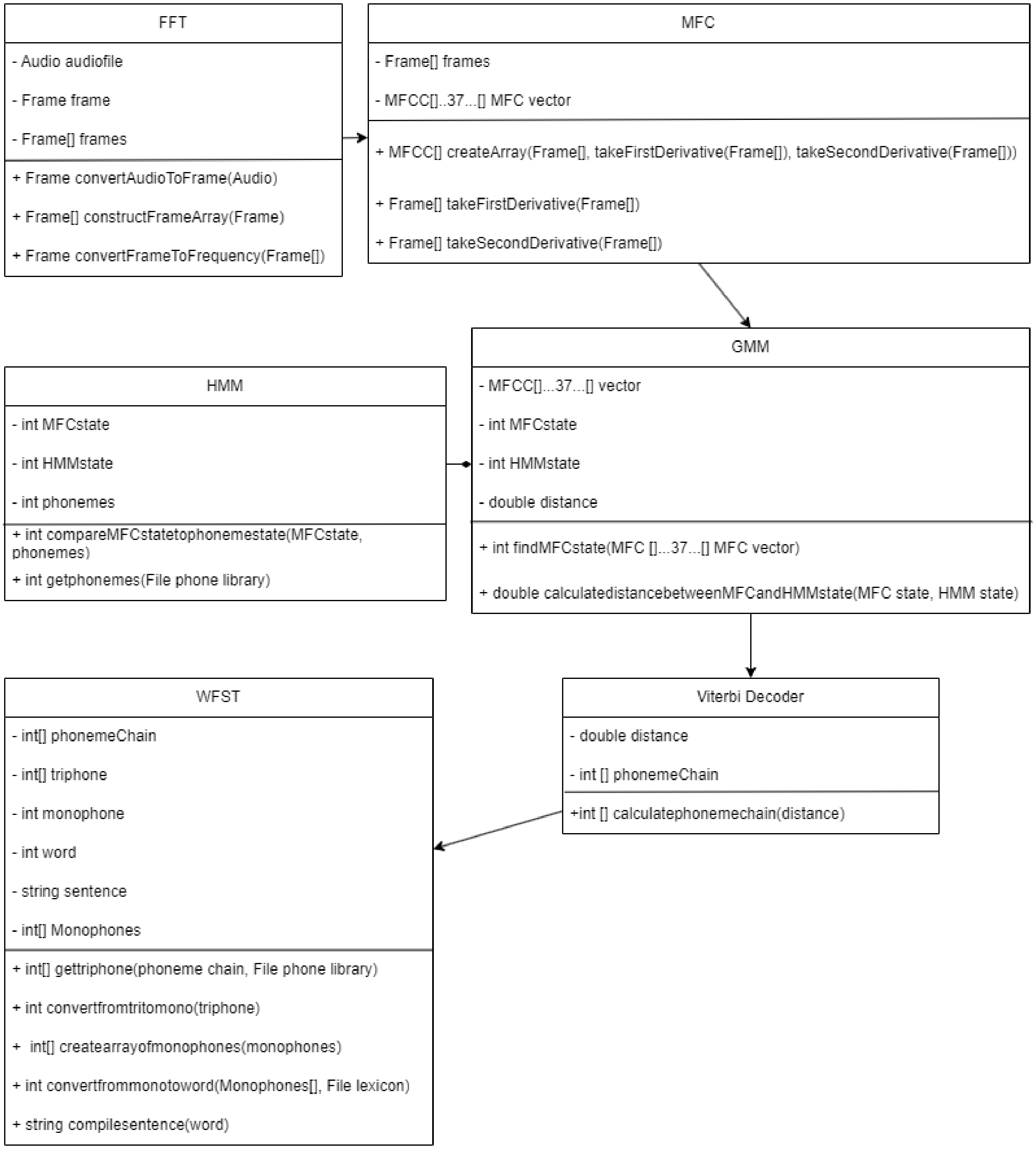


Use Case Diagram ASR Model V2.1.1



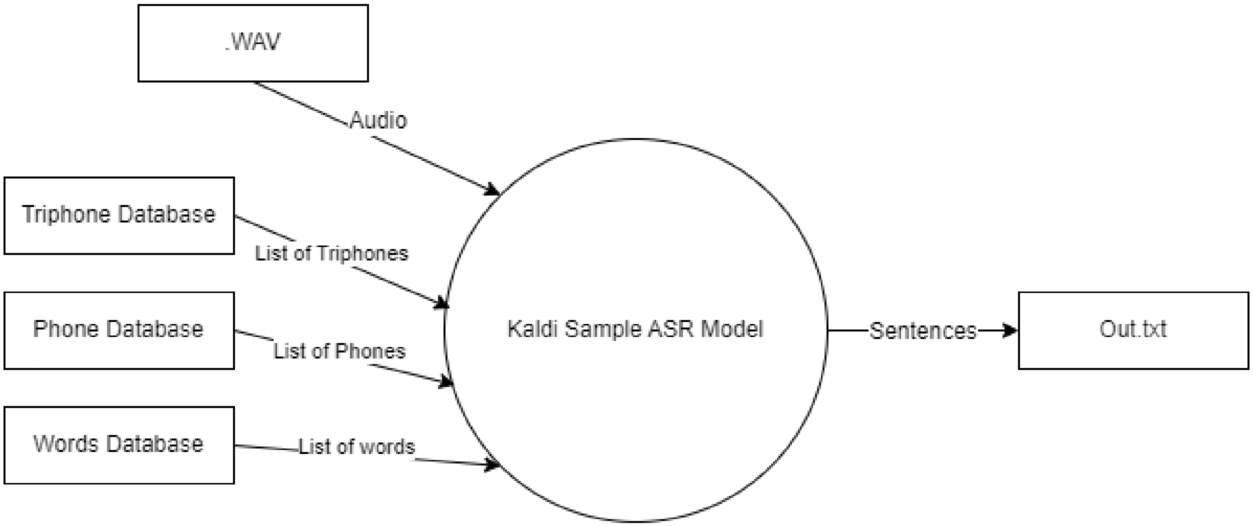
Use Case Diagram ASR Model V2.2.1

11/19


FFT
- Audio audiofile
- Frame frame
- Frame[] frames
+ Frame convertAudioToFrame(Audio)
+ Frame[] constructFrameArray(Frame)
+ Frame convertFrameToFrequency(Frame[])
MFC
- Frame[] frames
- MFCC[]..37...[] MFC vector
+ MFCC[] createArray(Frame[], takeFirstDerivative(Frame[]), takeSecondDerivative(Frame[]))
+ Frame[] takeFirstDerivative(Frame[])
+ Frame[] takeSecondDerivative(Frame[])
HMM
- int MFCstate
- int HMMstate
- int phonemes
+ int compareMFCstatetophonemestate(MFCstate, phonemes)
+ int getphonemes(File phone library)
GMM
- MFCC[]...37...[] vector
- int MFCstate
- int HMMstate
- double distance
+ int findMFCstate(MFC []...37...[] MFC vector)
+ double calculatedistancebetweenMFCandHMMstate(MFC state, HMM state)
WFST
- int[] phonemeChain
- int[] triphone
- int monophone
- int word
- string sentence
- int[] Monophones
+ int[] gettriphone(phoneme chain, File phone library)
+ int convertfromtritomono(triphone)
+ int[] createarrayofmonophones(monophones)
+ int convertfrommonotoword(Monophones[], File lexicon)
+ string compilesentence(word)
Viterbi Decoder
- double distance
- int [] phonemeChain
+int [] calculatephonemechain(distance)

Class Diagram V3.3

11/22



Context Diagram V3.1