mimic human speech in bahasa indonesia using  
speech recognition and speech synthesis

By

Valens Prabagita Ivan Susilo

A Thesis

Submitted to the Faculty of Computing

President University

in Partial Fulfilment of the Requirements

for the Degree of Bachelor of Science

in Information Technology

Cikarang, Bekasi, Indonesia

October 2018

Copyright by

Valens Prabagita Ivan Susilo

2018

mimic human speech in bahasa indonesia using  
Speech recognition and speech synthesis

By

Valens Prabagita Ivan Susilo

Approved:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Tjong Wan Sen, S.T., M.T. Drs. Nur Hadisukmana, M.Sc.

Thesis Advisor Program Head of Information Technology

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ir. Rila Mandala, M.Eng., Ph.D.

Dean of Faculty of Computing

ABSTRACT

Everyday people use Speech recognition and speech synthesis unconsciously. The technologies help them with their activities. With each technology can produce any kinds software related to speech. Combine both of technologies can produce many more. One of the combinations is mimic human speech. This research will discuss about Convolutional Neural Network, Speech Recognition, and Speech Synthesis. The purpose of this research is to develop application to collect, train, and mimic speech in Bahasa Indonesia. User can participate record their speech. The collected speech will be train to be used in the application to recognize the speech. After the collected speech is trained, User can mimic their speech by identify or recognize the speech and generate or synthesis the speech. The application to collect and mimic speech develops in website application.

Keywords: Convolutional Neural Network, Speech Recognition, Speech Synthesis

ACKNOWLEDGMENTS

The author gratefully acknowledges the amazing supports, guidance and advices from:

1. Bebe, Annisa Feryannie Widya.
2. My beloved family, Ayah, Ibu, Mas Risky and Dek Lia.
3. B35TFR13NDZ0NE, Nisa, Marsel, Bora, Fira, Rai, Pindy, Oscar, and Michel.
4. Sangar Team, Mas Andreas, Mba Diana, Nisa, Alfian, and Aci
5. DV 14, Vovo, Gojal, Ariyel, Rino, Willy, Azmi, Damara, Atisah, Risda, Sonya, Dini, and Arizha.
6. My cool thesis lecturer, Mr. Wan Sen.
7. My second family, Van Lith XXII.
8. And you, yes you.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS ............................................................................................. i

TABLE OF CONTENTS .............................................................................................. ii

LIST OF TABLES ....................................................................................................... iv

LIST OF FIGURES ...................................................................................................... v

CHAPTER

I INTRODUCTION ............................................................................................ 1

1.1 Background ........................................................................................... 1  
 1.2 Problem Statement ................................................................................ 2  
 1.3 Research Objective ................................................................................ 2  
 1.4 Scope and Limitation ............................................................................. 2  
 1.5 Thesis Methodology .............................................................................. 3  
 1.6 Thesis Outline ........................................................................................ 4

II LITERATURE STUDY .................................................................................... 6

2.1 Machine Learning .................................................................................. 6  
 2.1.1 Supervised Learning ................................................................. 7  
 2.1.2 Unsupervised Learning ............................................................. 7  
 2.2 Neural Network ..................................................................................... 9  
 2.3 Phonemic ............................................................................................... 9  
 2.4 Speech Recognition ............................................................................. 10  
 2.5 Pre-processing Speech ........................................................................ 10  
 2.5.1 Sound Sampling ...................................................................... 10  
 2.5.2 MFCC ..................................................................................... 11  
 2.6 Speech Synthesis ................................................................................. 12  
 2.7 Related Work ....................................................................................... 12  
 2.7.1 Lyrebird .................................................................................. 12  
 2.7.2 Google Translate ..................................................................... 13

III SYSTEM ANALYSIS .................................................................................... 15  
 3.1 System Overview ................................................................................ 15  
 3.2 Functional Analysis ............................................................................. 15 3.3 Software and System Requirements .................................................... 16

CHAPTER Page  
  
 3.4 System Architecture ............................................................................ 17  
 3.4.1 Use-Case Diagram .................................................................. 17  
 3.4.2 Use-Case Narrative ................................................................. 18  
 3.4.3 Activity Diagram ..................................................................... 21

IV SYSTEM DESIGN ......................................................................................... 23  
 4.1 User Interface Design .......................................................................... 23  
 4.1.1 Collect Application ................................................................. 31  
 4.1.1.1 Home Page .................................................................. 31  
 4.1.1.2 Phonemes Page ........................................................... 32  
 4.1.2 Mimic Application .................................................................. 34  
 4.1.2.1 Home Page .................................................................. 34  
 4.1.2.2 Identify Page ............................................................... 35  
 4.1.2.3 Generate Page ............................................................. 38  
 4.2 Class Diagram ..................................................................................... 26  
 4.2.1 Collect Application ................................................................. 31  
 4.2.1.1 Home ........................................................................... 31  
 4.2.1.2 Phonemes .................................................................... 32  
 4.2.2 Train Application .................................................................... 31  
 4.2.2.1 Model .......................................................................... 31  
 4.2.2.2 Train ............................................................................ 32  
 4.2.3 Mimic Application .................................................................. 34  
 4.2.3.1 Home ........................................................................... 34  
 4.2.3.2 Identify ........................................................................ 35  
 4.2.3.3 Generate ...................................................................... 38  
 4.2.4 Server ...................................................................................... 34  
 4.2.4.1 serverHanlder .............................................................. 34  
 4.2.4.2 createServer ................................................................ 35  
 4.2.5 Database .................................................................................. 34  
 4.2.5.1 connection ................................................................... 34  
 4.2.5.2 model .......................................................................... 35

IV SYSTEM DEVELOPMENT ........................................................................... 31  
 5.1 User Interface Development ................................................................ 31  
 5.1.1 Collect Application ................................................................. 31  
 5.1.1.1 Home Page .................................................................. 31  
 5.1.1.2 Phonemes Page ........................................................... 32  
 5.1.2 Mimic Application .................................................................. 34  
 5.1.2.1 Home Page .................................................................. 34  
 5.1.2.2 Identify Page ............................................................... 35  
 5.1.2.3 Generate Page ............................................................. 38  
 5.2 Application Details .............................................................................. 41  
 5.2.1 Server Code ............................................................................. 41  
 5.2.2 Router Code ............................................................................ 43  
 5.2.3 Database Code ......................................................................... 51  
 5.2.4 Train Application Code ........................................................... 53  
 5.2.5 Library Package Code ............................................................. 58

IV SYSTEM TESTING ....................................................................................... 59  
 6.1 Testing Environment ........................................................................... 59  
 6.2 Testing Scenario .................................................................................. 59  
 6.2.1 Collect Application ................................................................. 60  
 6.2.2 Train Application .................................................................... 61  
 6.2.3 Mimic Application .................................................................. 61

IV CONCLUSIONS AND FUTURE WORK ...................................................... 59  
 7.1 Conclusion ........................................................................................... 59  
 7.2 Future Work ........................................................................................ 59

REFERENCES ............................................................................................................ vi

LIST OF TABLES

TABLE Pag

3.1 Collect application functionality table ............................................................. 20

3.2 Train application functionality table ................................................................ 22

3.3 Mimic application functionality table .............................................................. 22

3.4 Use-case narrative – Access Collect application .............................................. 22

3.5 Use-case narrative – Record Speech ................................................................ 22

3.6 Use-case narrative – Train model ..................................................................... 22

3.7 Use-case narrative – Access Mimic application ............................................... 22

3.8 Use-case narrative – Identify speech ................................................................ 22

3.9 Use-case narrative – Generate speech .............................................................. 23

4.1 Collect Home page description ........................................................................ 28

4.2 Collect Phoneme page description ................................................................... 29

4.3 Mimic Home page description ......................................................................... 30

4.4 Mimic Identify page description ...................................................................... 28

4.5 Mimic Generate page description .................................................................... 29

6.1 Collect application scenarios ........................................................................... 28

6.2 Train application scenarios .............................................................................. 29

6.3 Mimic application scenarios ............................................................................ 30

6.4 Tester 1 speech recognition on identify process results.................................... 24

6.5 Tester 2 speech recognition on identify process results.................................... 24

6.4 Tester 3 speech recognition on identify process results ................................... 24

LIST OF FIGURES

FIGURE Page

1.1 RAD Diagram [3] .............................................................................................. 8

2.1 Neural network to find estimated price from specific input [7] ........................ 9

2.2 Output comparison between neural network that

has no memory (left) with the one that has memory (right) [7] ........................ 8

2.3 The basic neural network where it only recognizes the object

on the center (top), but doesn't on another surface (bottom) [8] ...................... 11

2.3 Sound sampling process ................................................................................... 11

2.4 100 samples in numbers from “Hello” sound with sample rate

of 16kHz (16 samples per seconds) ................................................................. 11

2.5 Screenshot of Lyrebird in the website .............................................................. 13

2.6 Screenshot of Google Translate in website ...................................................... 14

3.1 Use-case diagram ........................................................................................... 18

3.2 Collect application activity diagram ................................................................ 22

3.3 Train application activity diagram .................................................................. 22

3.4 Mimic application activity diagram ................................................................ 22

4.1 Collect Home page ........................................................................................... 28

4.2 Collect Phoneme page ..................................................................................... 29

4.3 Mimic Home page ........................................................................................... 30

4.4 Mimic Identify page ......................................................................................... 28

4.5 Mimic Generate page ....................................................................................... 29

4.6 Collect application class diagram .................................................................... 30

4.7 Train application class diagram ....................................................................... 28

4.8 Mimic application class diagram ..................................................................... 28

4.9 Server class diagram ........................................................................................ 29

4.10 Database class diagram .................................................................................. 30

5.1 Collect Home Page .......................................................................................... 22

5.2 Collect Phonemes page with ‘a’ phoneme ....................................................... 24

5.3 Collect Phonemes page after record process is finish ...................................... 25

5.4 Mimic Home page ........................................................................................... 26

5.5 Mimic Identify page ......................................................................................... 28

5.6 Mimic Identify page alert user indicating identify process

in the server is finish ........................................................................................ 30

FIGURE Page

5.7 Mimic Identify page alert user indicating there is

no value in Input element ................................................................................. 18

5.8 Mimic Generate page ....................................................................................... 22

5.9 Mimic Generate page play the generate speech

after generate process success ......................................................................... 24

5.10 Mimic Generate page alert user indicating there is

no value in Textarea element ......................................................................... 25

5.11 Mimic Generate page alert user indicating there is

error in the generate process .......................................................................... 26

5.12 serverHandler on Server code ........................................................................ 28

5.13 Create server command on Server code ......................................................... 13

5.14 uploadCollect on Collect Router code ............................................................ 14

5.15 router on Collect Router code ........................................................................ 18

5.16 extract on Identify Router code ...................................................................... 22

5.17 uploadSpeech on Identify Router code .......................................................... 24

5.18 identifySpeech on Identify Router code ......................................................... 25

5.19 router on Identify Router code ....................................................................... 26

5.20 loadSpeech on Generate Router code ............................................................. 28

5.21 generateSpeech on Generate Router code ...................................................... 30

5.22 router on Generate Router code ...................................................................... 13

5.23 connection on Database code ......................................................................... 14

5.24 model on Database code ................................................................................ 18

5.25 extractWav on Train application code ............................................................ 22

FIGURE Page

5.26 extractFiles on loadData Train application code ............................................ 24

5.27 load and split command on loadData Train application code ........................ 25

5.28 getData on nextBatch Train application code ................................................ 26

5.29 getLabel on nextBatch Train application code ............................................... 28

5.30 saveModelDB on Train application code ....................................................... 30

5.31 modelML on model Train application code ................................................... 24

5.32 preparation command on train Train application code ................................... 25

5.33 train on train Train application code .............................................................. 26

6.1 Some screenshot from the Collect application scenarios ................................. 26

6.2 Some screenshot from the Train application scenarios .................................... 26

6.3 Some screenshot from the Mimic application scenarios .................................. 26

6.4 Some screenshot from the Tester 1 results ....................................................... 26

6.5 Some screenshot from the Tester 2 results ....................................................... 26

6.6 Some screenshot from the Tester 3 results ....................................................... 26

# INTRODUCTION

## Background

“Ok Google, play some music”. “Siri, what should I eat for lunch?”. Everyday people use their virtual assistance to boost their activities. People very like to use it because they just asked to their device and then in seconds, the wish is granted. It seems like, people are talking to the computer. The truth is, speech recognition takes big role with the help of machine learning. Google Assistance, Apple Siri, Microsoft Cortana, Amazon Alexa, and others have thousands of speech data to be analysed with the machine learning and they easily add data by collecting people speech from the assistance with permission.

If speech recognition is the process to get data by analysed speech, the opposite of speech recognition is speech synthesis, the process to produce artificial speech. Therefore, speech recognition is known as speech-to-text and speech synthesis is known as text-to-speech. “Hey Cortana, read my email” command make virtual assistance generate speech from the email text. With each technology can produce any kinds software related to speech. Combine both of technologies can produce many more. One of the combinations is mimic human speech.

## Problem Statement

This research aims to develop application which can be used to collect speech data, train machine learning model with collected data and mimic speech in Bahasa Indonesia. The application to collect and mimic speech develops in website application. The application can recognize the speech and generate speech from text.

## Research Objective

This research sees an opportunity to implement speech recognition and speech synthesis to create a mimic speech.

## Scope and Limitation

This research focuses on developing an application which will be able to:

1. Perform collecting data.
2. Perform train model.
3. Perform speech recognition.
4. Perform speech synthesis.

The limitations of this application are as following:

1. There are 10 selected phonemes to be used in the application, a, i, t, na, ma, mu, di, ri, and ku.
2. Recorded speech in 1 second, with sample rate 16000 and mono sound.
3. Speech recognition data is taken from recorded speech and in human speech in Bahasa Indonesia.
4. Speech synthesis data is taken from saved speech, result from speech recognition.
5. Application is developed as website application.
6. Collecting, testing, and running the application is used in the same hardware.

## Thesis Methodology

Rapid Application Development (RAD) methodology will be used in the development of this application. RAD method is a term originally used to describe a software development process introduced by James Martin in 1991 [1]. RAD method is a methodology to develop software that requires minimum planning for rapid prototyping. As James Martin says, RAD is a lifecycle used for development of software which provides faster development and also gives high quality software then, by using traditional software development lifecycle. In short, RAD is the process which accelerates the cycle of development of an application [2]. The RAD diagram is depicted in Figure 1.1.

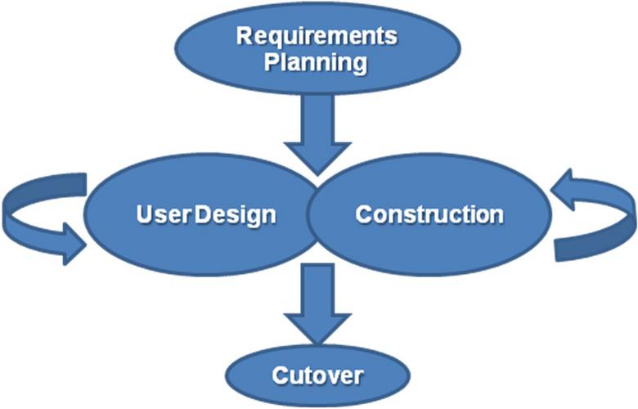


Figure 1.1 RAD Diagram [3].

The RAD model implemented in this thesis will consists of four major phases:

1. Requirement Planning Phase

This phase combines elements of the system planning and systems analysis phases of the Systems Development Life Cycle (SDLC) [3].

1. User Design Phase

During this phase, users interact with systems analysts and develop models and prototypes that represent all system processes, inputs, and outputs. This phase is a continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs [3].

1. Construction Phase

This phase focuses on program and application development task similar to the SDLC. In RAD, however, users continue to participate and can still suggest changes or improvements as actual screens or reports are developed. Its tasks are programming and application development, coding, unit-integration and system testing [3].

1. Cut Over Phase

This final phase resembles the final tasks in the SDLC implementation phase, including data conversion, testing, changeover to the new system, and user training. Compared with traditional methods, the entire process is compressed. As a result, the new system is built, delivered, and placed in operation much sooner [3].

## Thesis Outline

The thesis consists of seven chapters, which are as follow:

1. Chapter I: Introduction

This chapter introduce the research background, problem, and objective. It also explains the research scope and limitation, method to achieve the objective.

1. Chapter II: Literature Study

This chapter contain the literature study that related to the research background.

1. Chapter III: System Analysis

This chapter explains the analysis of the application – both in its function and behaviour, in order to fulfil the prescribed requirements.

1. Chapter IV: System Design

This chapter explains the system design of interfaces and class diagram based on the previous chapter that will be used in the next chapter.

1. Chapter V: System Development

This chapter explains the system development of interfaces and code details on the application.

1. Chapter VI: System Testing

This chapter ensures the application system runs well by evaluating all the features, and making sure the system fulfils its function requirements.

1. Chapter VII: Conclusion and Future Work

This chapter sums up this research and also suggestion for future research work.

# LITERATURE STUDY

## Machine Learning

Machine learning is a form of AI that enables a system to learn from data rather than through explicit programming [4]. Others state that machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI) [5].

In general, there are 2 types of machine learning, Supervised Learning and Unsupervised Learning [4, 5, 6].

### Supervised Learning

In supervised learning, the algorithms are trained using pre-processed examples, and at this point, the performance of the algorithms is evaluated with test data. Occasionally, patterns that are identified in a subset of the data can’t be detected in the larger population of data. If the model is fit to only represent the patterns that exist in the training subset, it creates a problem called overfitting [4].

Overfitting means that the machine learning model is precisely tuned for the training data but may not be applicable for large sets of unknown data. To protect against overfitting, testing needs to be done against unforeseen or unknown labelled data. Using unforeseen data for the test set can help evaluate the accuracy of the model in predicting outcomes and results [4].

### Unsupervised Learning

In unsupervised learning, the algorithms segment data into groups of examples (clusters) or groups of features. The unlabelled data creates the parameter values and classification of the data. In essence, this process adds labels to the data so that it becomes supervised [4].

Unsupervised learning can determine the outcome when there is a massive amount of data. Unsupervised can be used as the first step before passing the data to a supervised learning process if one doesn’t know the context of the data being analysed, and labelling isn’t possible at the stage [4].

## Neural Network

A neural network is an approach to machine learning in which small computational units are connected in a way that is inspired by connections in the brain [6]. Non-linear elements have as their inputs a weighted sum of the outputs of other elements, much like networks of biological neurons do [5].

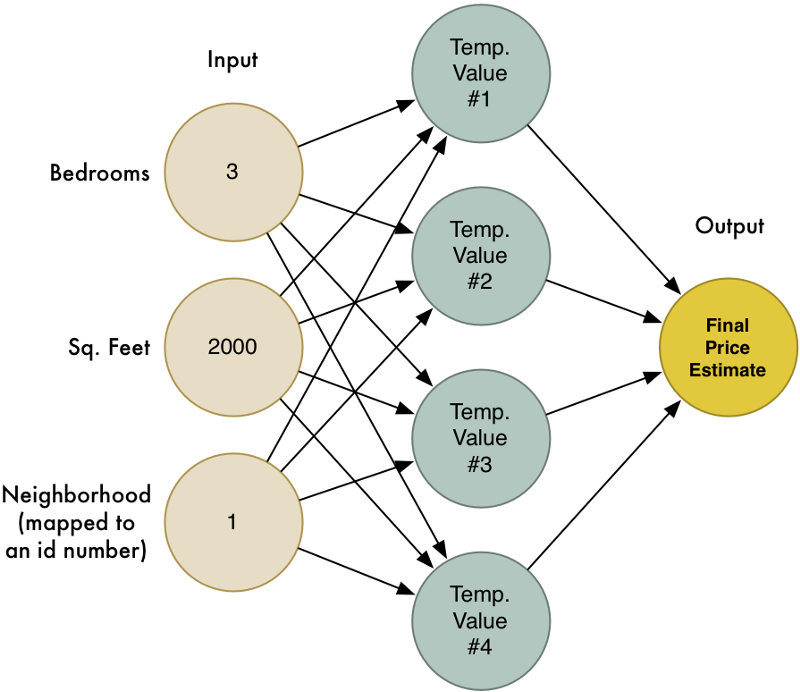


Figure 2.1 Neural network to find estimated price from specific input [7].

Every neural network model is basically a three-layered system, which are Input layer, Hidden Layer and Output Layer [4, 6]. Input layer, is designed to receive information from the outside model. While the other side of the network, an output layer, communicates a decision about the data that has been received. Between these, other layers communicate information about elements of the input to each other, which contribute to the output [6].

There are 2 known variants of neural network, Recursive Neural Network (RNN) and Convolutional Neural Network (CNN or ConvNet) [7, 8].

### Recursive Neural Network

In Recursive Neural Network, the neural network is design to have memory, it saves the model’s state and use it as part of next calculation consideration. The model updates each time it is used. This allows it to update its predictions based on what it saw most recently. It can even model patterns over time as long as the model is given enough memory [7].

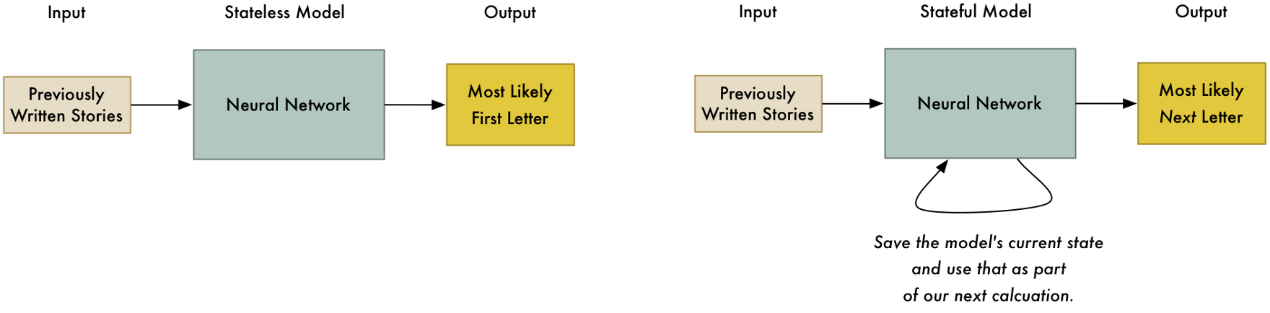


Figure 2.2 Output comparison between neural network that has no memory (left) with the one that has memory (right) [7].

### Convolutional Neural Network

In Convolutional Neural Network, the neural network is designed to recognize an object no matter what surface the object is on. The model doesn’t have to re-learn the idea of child for every possible surface it could appear on [8].

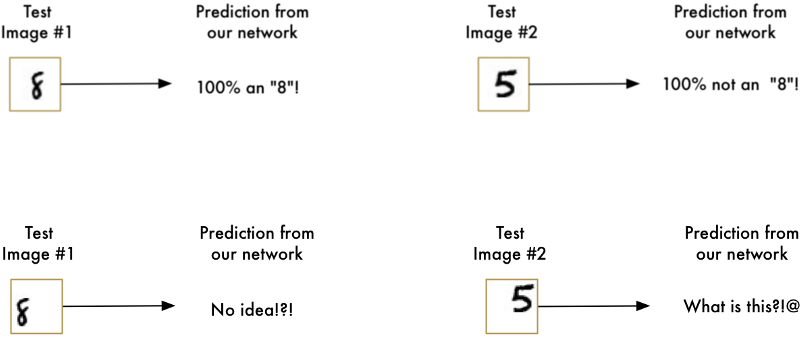


Figure 2.3 The basic neural network where it only recognizes the object on the center (top), but doesn't on another surface (bottom) [8].

## Phonemic

*…Explanation of phonemic…*

A phoneme is the basic unit of phonology. It is the smallest unit of sound that may cause a change of meaning within a language, but that doesn’t have meaning by itself. For example, in the words’ “bake” and “brake,” only one phoneme has been altered, but a change in meaning has been triggered. The phoneme /r/ has no meaning on its own, but by appearing in the word it has completely changed the word’s meaning [10].

In the other hand, grapheme is individual letters and groups of letters that represent single phonemes, like the “s” and the “oo” in “spoon”. Understanding how letters are used to encode speech sounds in written language is crucial in learning to decode unfamiliar words [11].

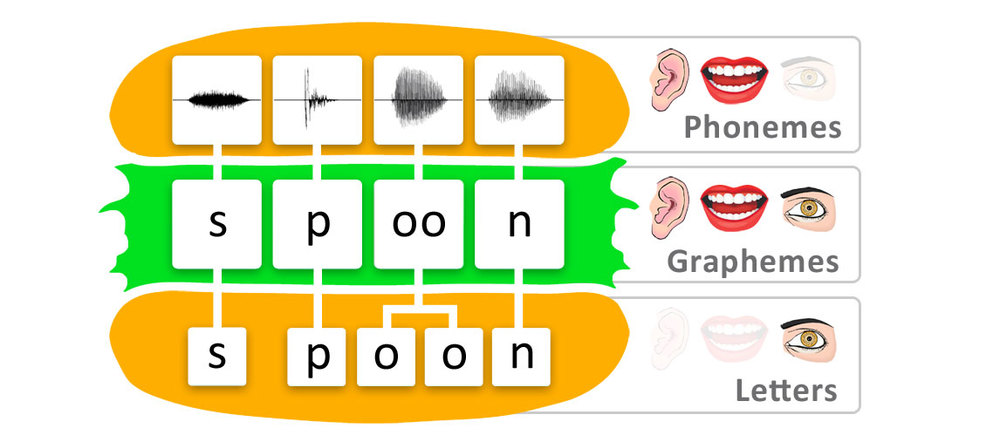


Figure 2.2 Phonemes, graphemes, and letters in the word “spoon” [11].

In Bahasa Indonesia phoneme is distributed into *vokal*, *diftong*, *konsonan*, *gugus konsonan* [12].

*…Explanation of silabel…*

## Speech Recognition

The process of automatically recognizing spoken words of speaker based on information in speech signal is called Speech Recognition [21]. Other definition of speech recognition, also known as Automatic Speech Recognition (ASR), or computer speech recognition, is the process of converting a speech signal to a sequence of words, by means of an algorithm implemented as a computer program [22].

*…Google Convolutional Neural Network for Small-footprint Keyword Spotting…*

## Pre-processing Speech

As sound is transmitted as waves, and computer understand numbers, it’s necessary to pre-processing speech so that computer understand and can be feed as input into neural network.

*…Additional explanation to convert to CNN…*

### Sound Sampling

Sound sampling is taking a reading thousands of times a second and recording a number representing the height of the sound wave at that point in time. Basically, all an uncompressed .wav audio file [23].

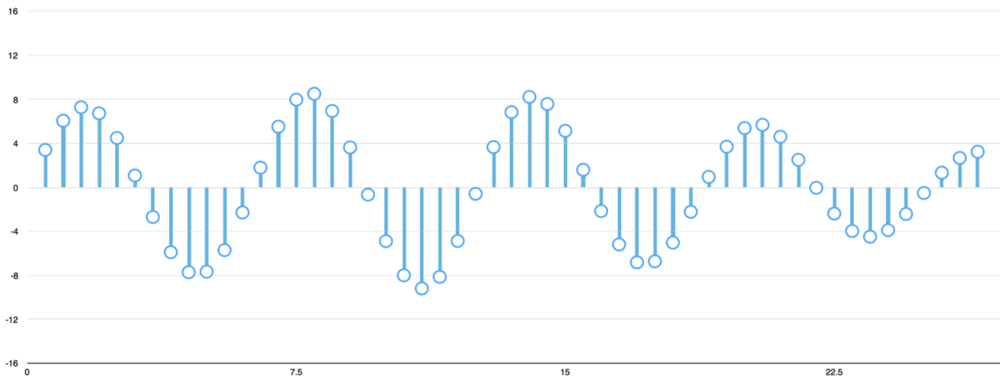
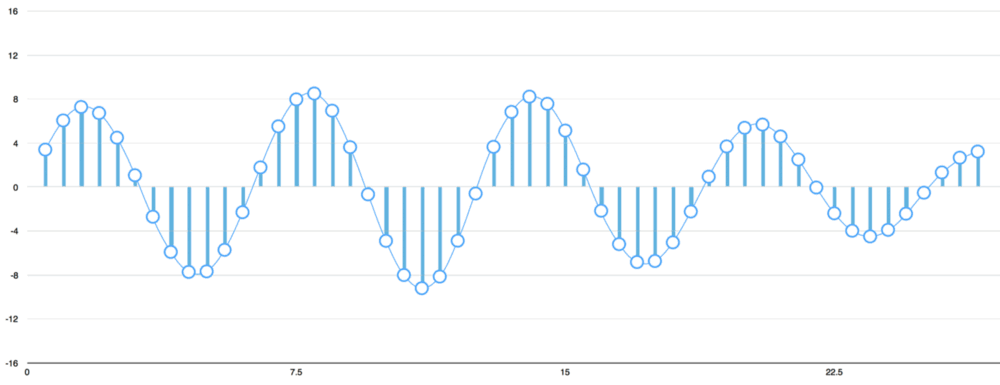
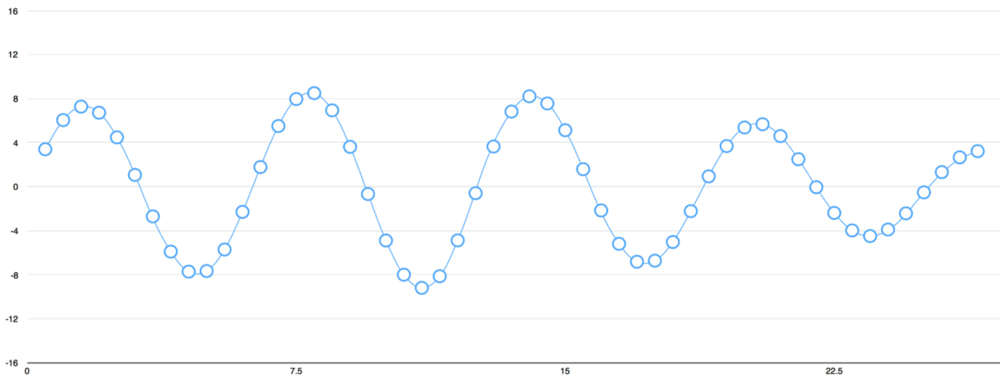
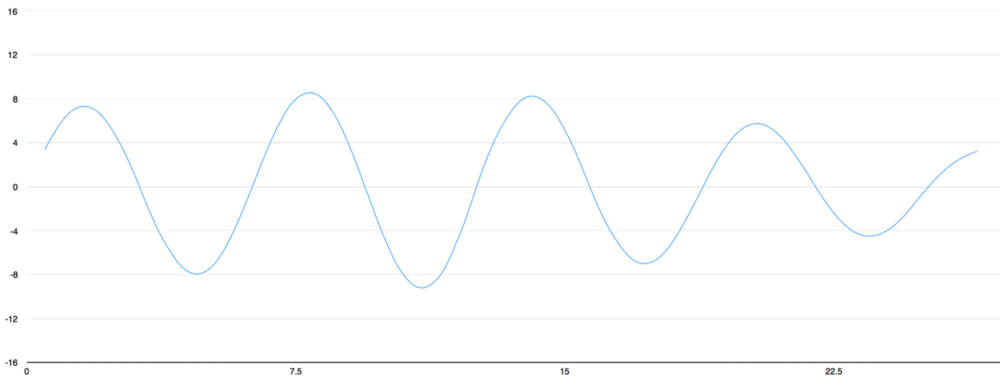


Figure 2.3 Sound sampling process [23].

https://cdn-images-1.medium.com/max/1000/1*BG4iFbx7qhb5v_JTr958PQ.png

Figure 2.4 100 samples in numbers from “Hello” sound with sample rate of 16kHz (16 samples per seconds) [23].

Nyquist sampling theorem [24] provides a prescription for the nominal sampling interval required to avoid aliasing. The sampling frequency should be at least twice the highest frequency contained in the signal. In the case, where one has *fc* = 3 Hz, and so the Nyquist theorem tells that the sampling frequency, *fs*, must be at least 6 Hz [25].

### MFCC

*…Explanation of MFCC…*

## Speech Synthesis

Speech synthesis is the artificial production of human speech [26]. The Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database [27]. From phoneme, text can be analysed by its phonemes and then with the phonemes concatenating speech can be done.

## Related Work

The following are most related work to the research. Have relation to mimic speech, both speech recognition and speech synthesis.

### Lyrebird

Lyrebird is website application, *https://lyrebird.ai*, that has 3 products: Custom Voice, Vocal Avatar, and Vocal Avatar API [28]. Custom voice is a product to create speech based on real people’s speech, it can control the intonation, expression, and the emotion of the speech. Vocal avatar is a product to create own digital speech by read some English sentences, and then generate any sentences with own digital speech. Vocal avatar API is a product to provide API to use user’s own vocal avatar.

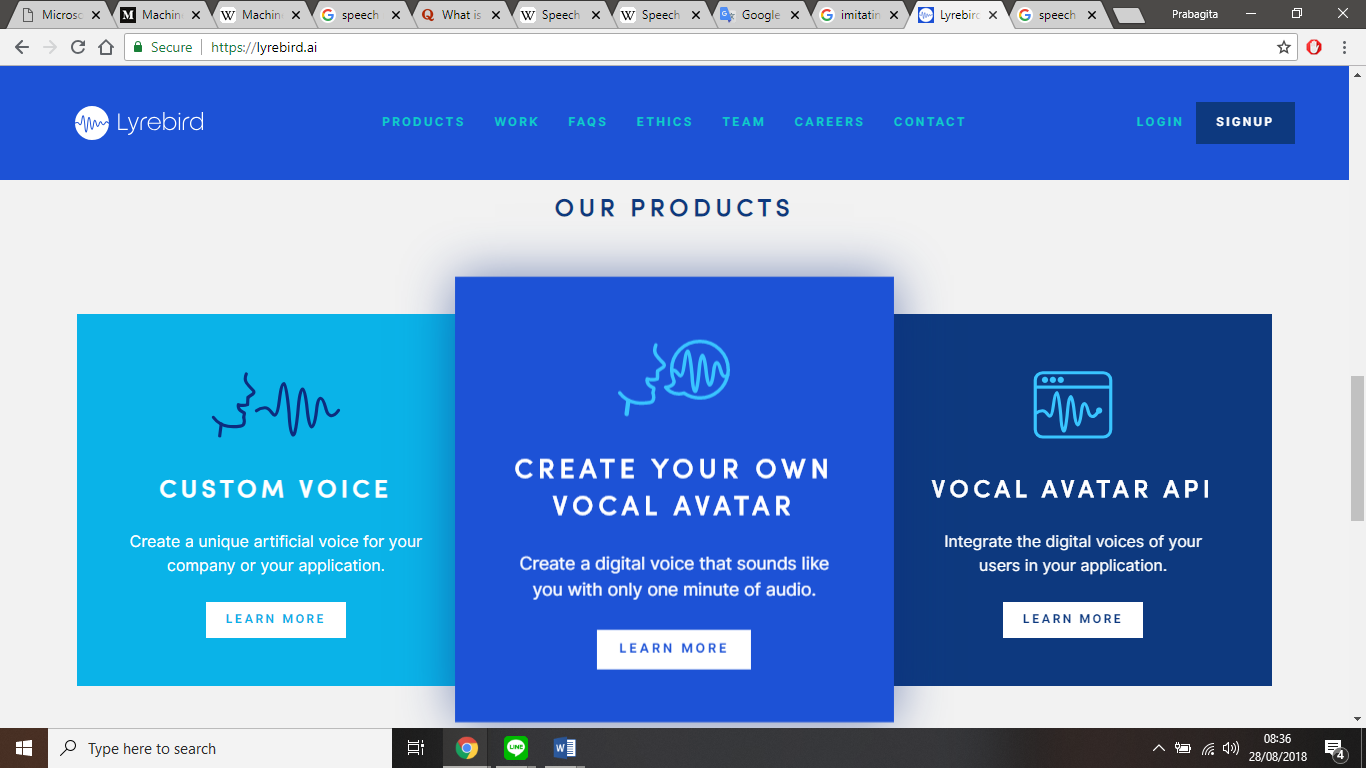
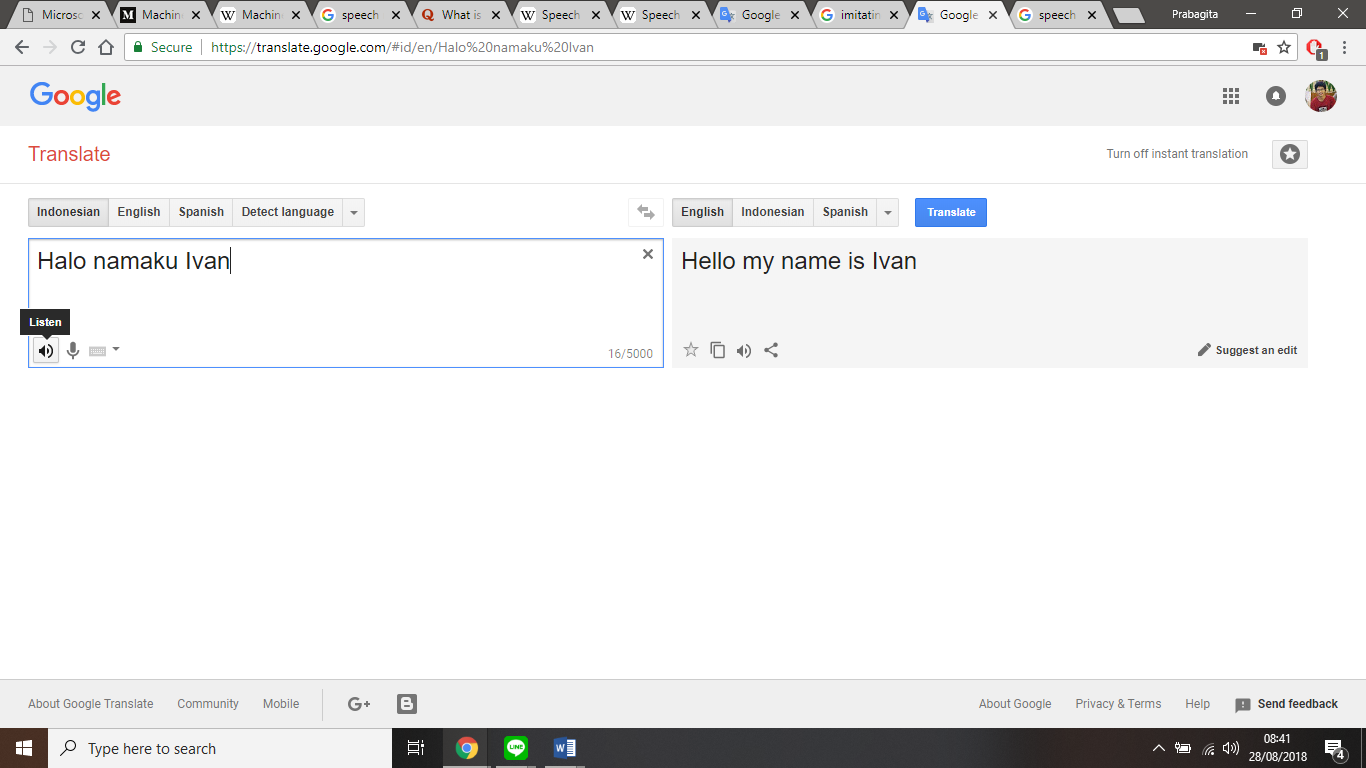


Figure 2.5 Screenshot of Lyrebird in the website [28].

### Google Translate

Google Translate is one of Google products that is an application to translate languages. Google Translate can be access freely through [*https://translate.google.com/*](https://translate.google.com/). It has many features [29] and one of them is Talk feature. Talk has function to input text from speech and generate speech from text in any languages.

Figure 2.6 Screenshot of Google Translate in website [29].

From related work, it can be concluded that Lyrebird can mimic speech with its vocal avatar, but the speech is in English. In the other hand, Google Translate could mimic speech into any languages, but the speech vocal is from the Google Translate itself.

# SYSTEM ANALYSIS

## System Overview

This research is intended to implement speech recognition and speech synthesis into this research. This research consists of 3 application, Collect, Train, and Mimic application. Collect application is used to collect speech data. Train application train model with collected data for speech recognition with convolutional neural network as machine learning model. Mimic application is used to mimic the speech. It identifies what user said with trained model. Saves the speech data if the model recognizes. Then, it generates speech with concatenating saved speech data. Collect application and Mimic application develops in website application.

## Functional Analysis

There are several functions for each application listed in the Table 3.1 up to 3.3.

Table 3.1 Collect application functionality table.

|  |  |
| --- | --- |
| **No** | **Function Description** |
| 1 | Allow user to access Collect application. |
| 2 | Allow user to record user’s speech. |

Table 3.2 Train application functionality table.

|  |  |
| --- | --- |
| **No** | **Function Description** |
| 1 | Allow user to train model with collected data. |

Table 3.3 Mimic application functionality table.

|  |  |
| --- | --- |
| **No** | **Function Description** |
| 1 | Allow user to access Mimic application |
| 2 | Allow user to identify user’s speech. |
| 3 | Allow user to generate speech. |

## Software and System Requirements

This research and application development should be supported by the following list requirement in order to write the research, build and run the application well.

1. Laptop / Personal Computer

Laptop or Personal Computer is used as the tool where operating system is run. In this research, ASUS A455LN is used with Windows 10 as the OS.

1. Browser

Browser is used as development tool when developing website application. In this research, Chrome v70.0.3538.110 is used.

1. Microsoft Office Word

Microsoft Office Word is used to write the research documentation. In this research, Microsoft Office Word 2016 is used.

1. Node.js, JavaScript Run-Time Environment.

Node.js is an open source server environment – Node.js is free – Node.js runs on various platform (Windows, Linux, Unix, Mac OS X, etc) – Node.js uses JavaScript on the server [30]. In this research, Node.js v10.14.1 is used.

1. NoSQL document-oriented database

Document databases pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents [31]. In this research, MongoDB Community Server 4.0.2 as database server and MongoDB Compass 1.15.4 as MongoDB UI.

1. Integrated Development Environment (IDE)

IDE is used as application development environment. In this research, Visual Studio Code 1.30.0 is used.

1. Git

Git is used as version control. The repository is placed on local and cloud as preventive action. In this research, Git 2.18.0.windows.1 and GitLab as cloud repository is used.

## System Architecture

The system architecture explains about the use-case diagram and narrative for this application in both point of view, the actors and the system.

### Use-Case Diagram

Use-Case Diagram defines the functionality of a system and explains it in user point of view. The actors in this research is the application user. The diagram is shown in Figure 3.1.

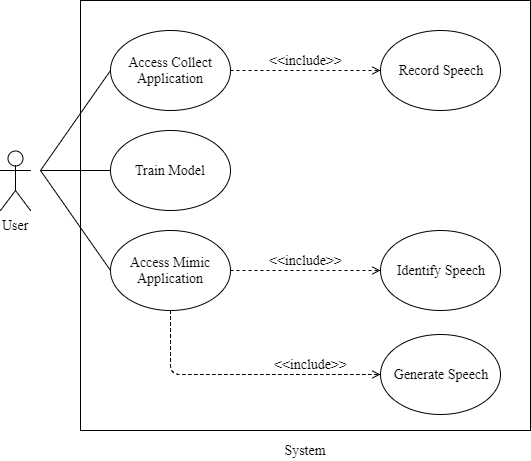


Figure 3.1 Use-case diagram.

### Use-Case Narrative

Use-Case Narrative explains the interaction between the actors and the system. It describes the detail of use-cases such as name, description, pre-condition, post-condition, business rules, and the course of events that happened in the system. The Use-Case Narrative is shown in Table 3.4 up to Table 3.9.

Table 3.4 Use-case narrative – Access Collect application.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Access Collect application** | |
| Use Case ID | UC01 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user opens Collect application through browser. | |
| Precondition | None | |
| Trigger | User opens Collect application. | |
| Typical Course of Event | **Actor Action** | **System Response** |
| Click Start button. | Phonemes page is shown. |
| Alternate Course | None | |
| Post Condition | Home page is shown. | |
| Business Rule | None | |
| Implementation Constraint and Specifications | None | |

Table 3.5 Use-case narrative – Record speech.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Record speech** | |
| Use Case ID | UC02 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user opens Phonemes page. | |
| Precondition | User clicks Start button from Home page. | |
| Trigger | User opens Phonemes page. | |
| Typical Course of Event | **Actor Action** | **System Response** |
| Click Record button and say asked phoneme. | Record the speech and upload to save it. |
| Alternate Course | **Actor Action** | **System Response** |
| Click Next button. | Next Phonemes or Home page is shown. |
| Post Condition | Recorded speech is played on audio element. | |
| Business Rule | None | |
| Implementation Constraint and Specifications | None | |

Table 3.6 Use-case narrative – Train model.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Train model** | |
| Use Case ID | UC03 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user start Train application. | |
| Precondition | None | |
| Trigger | User starts Train application. | |
| Typical Course of Event | **Actor Action** | **System Response** |
|  | Load and extract collected data. |
|  | Split into train, validation, and test data. |
|  | Convert data to model’s input and output. |
|  | Train the model with train data and validate with validation data. |
|  | Test the model with test data and insert the model to MongoDB. |
| Alternate Course | None | |
| Post Condition | Model is trained and inserted in MongoDB. | |
| Business Rule | None. | |
| Implementation Constraint and Specifications | None | |

Table 3.7 Use-case narrative – Access Mimic application.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Access Mimic application** | |
| Use Case ID | UC04 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user opens Mimic application through browser. | |
| Precondition | None | |
| Trigger | User opens Mimic application. | |
| Typical Course of Event | **Actor Action** | **System Response** |
| Click Identify Speech or Generate Speech button. | Identify or Generate page is shown. |
| Alternate Course | None | |
| Post Condition | Home page is shown. | |
| Business Rule | None | |
| Implementation Constraint and Specifications | None | |

Table 3.8 Use-case narrative – Identify speech.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Identify speech** | |
| Use Case ID | UC05 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user opens Identify page. | |
| Precondition | User clicks Identify Speech button from Home page. | |
| Trigger | User opens Identify page. | |
| Typical Course of Event | **Actor Action** | **System Response** |
| Click Record button and say something. | Record the speech and upload to save it. |
|  | Recorded speech is played on audio element. |
| Insert speech ID in input element and click Identify Speech button. | Recorded speech is identified with trained model. |
|  | Speech data is updated to MongoDB if recognized. |
| Alternate Course | **Actor Action** | **System Response** |
| Click Finish button. | Home page is shown. |
| Post Condition | Information whether recognized or not is alerted in the browser. | |
| Business Rule | None | |
| Implementation Constraint and Specifications | None | |

Table 3.9 Use-case narrative – Generate speech.

|  |  |  |
| --- | --- | --- |
| **User Case Name** | **Identify speech** | |
| Use Case ID | UC06 | |
| Priority | High | |
| Primary Business Actor | User | |
| Primary System Actor | System | |
| Another Participating Actor | None | |
| Description | This use-case describes the event when user opens Generate page. | |
| Precondition | User clicks Generate Speech button from Home page. | |
| Trigger | User opens Generate page. | |
| Typical Course of Event | **Actor Action** | **System Response** |
|  | Load speech data from MongoDB. |
| Choose speech ID from datalist element. | Speech ID selected. |
| Insert text in input element and click Generate Speech button. | Speech is generated based selected speech ID and inserted text. |
| Alternate Course | **Actor Action** | **System Response** |
| Click Finish button. | Home page is shown. |
| Post Condition | Generated speech is played on audio element. | |
| Business Rule | None | |
| Implementation Constraint and Specifications | None | |

### Activity Diagram

Activity Diagram presents a series of actions and flow of control in a system. The activity diagram is separated into each application.

#### Collect Application

This activity describes the Collect application which is start when user opens the application. Home page will be shown as the home page. As user clicks Start button, the Phonemes page will be shown. User can record asked phonemes by click Record button. Recorded speech is played in audio element. User can also change the next asked phonemes by click Next button and the next Phonemes page will be shown. If there are no more asked phonemes, Home page will be shown instead the next Phonemes.

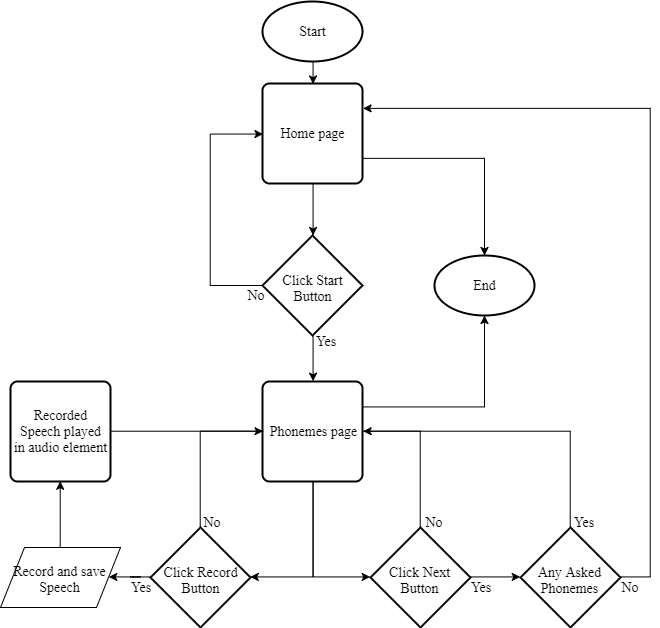


Figure 3.2 Collect application activity diagram.

#### Train Application

This activity describes the Train application which is start when user starts the application. Collected data will be loaded and extracted. Then, data split into train, validation, and test data. Each split data converted to model’s input and output. The model is trained with train data and validated with validation data. Finally, the model is tested with test data and inserted to MongoDB. The model is trained and ready to use by Mimic application.

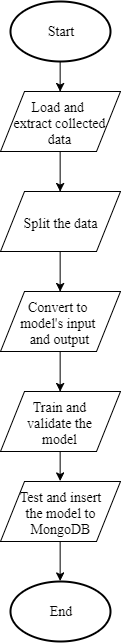


Figure 3.3 Train application activity diagram.

#### Mimic Application

This activity describes the Mimic application which is start when user opens the application. Home page will be shown as the home page. As user clicks Identify Speech button, the Identify page will be shown. User can record speech by click Record button. Then, user inserts speech ID on input element to name the speech data and click Identify Speech button to identify the recorded speech. Recognized speech will update the speech data to MongoDB. Any information regarding to identify process is alerted when finish to the browser. User can also click Finish button and Home page will be shown again. As user clicks Generate Speech button, the Generate page will be shown. Speech ID is loaded as shown as the page is loaded. User can choose speech ID. Inserts text on textarea element. And clicks Generate Speech button to generate speech based on the speech ID and the text. Generated speech is played in audio element. There’s also Finish button in Generate page that also shown Home page when clicked.

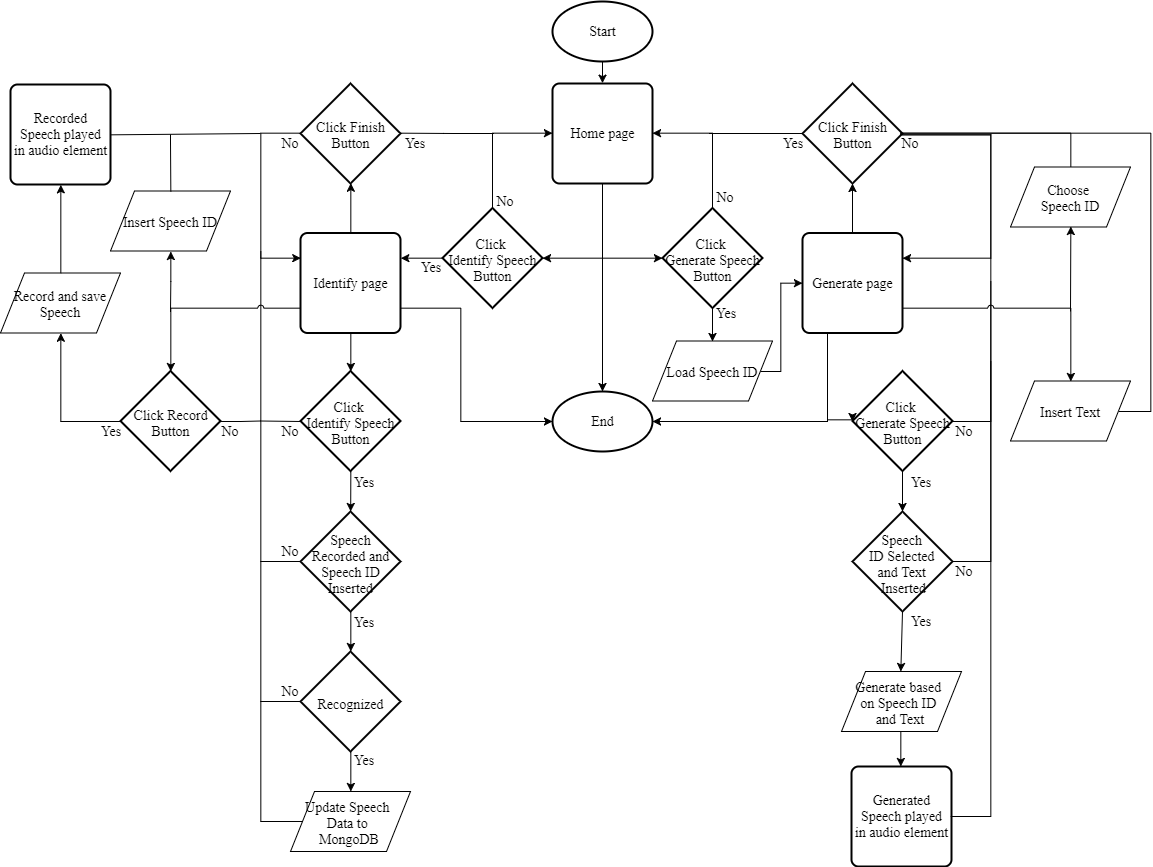


Figure 3.3 Mimic application activity diagram.

# SYSTEM DESIGN

## User Interface Design

The UI is applied on Collect and Mimic application. There are two core pages in Collect application and three core pages in Mimic Application.

### Collect Application

Two core pages in Collect application are Home page and Phonemes page.

#### Home Page

Home page is the home page of the Collect application. When user opens the Collect application or directed from Mimic application or redirect back from Phonemes page, it will show the Home page that consist of title, information regarding to Collect application, and 2 buttons such as Start and Mimic Home.



Figure 4.1 Collect Home page.

Table 4.1 Collect Home page description.

|  |  |
| --- | --- |
| **No** | **Description** |
| 1 | Title |
| 2 | Information |
| 3 | Start Button |
| 4 | Mimic Home Button |

#### Phonemes Page

Phonemes page is the main page of the Collect application. When user click Start in the Home page, it will show the Phonemes page that consist of title regarding to which phoneme should be recorded, audio element, and 2 buttons such as Record and Next.

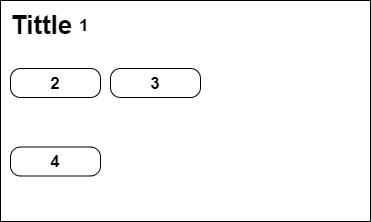


Figure 4.2 Collect Phonemes page.

Table 4.2 Collect Phonemes page description.

|  |  |
| --- | --- |
| **No** | **Description** |
| 1 | Title |
| 2 | Audio Element |
| 3 | Record Button |
| 4 | Next Button |

### Mimic Application

Three core pages in Mimic application are Home page, Identify page, and Generate page.

#### Home Page

Home page is the home page of the Mimic application. When user opens the Mimic application or directed from Collect application or redirect back from Identify or Generate page, it will show the Home page that consist of title, information regarding to Mimic application, and 3 buttons such as Identify Speech, Generate Speech and Collect Home. The description of the design layout is shown in Table 4.3.



Figure 4.3 Mimic Home page.

Table 4.3 Mimic Home page description.

|  |  |
| --- | --- |
| **No** | **Description** |
| 1 | Title |
| 2 | Information |
| 3 | Identify Speech Button |
| 4 | Generate Speech Button |
| 5 | Collect Home Button |

#### Identify Page

Identify page is the one of the main pages of the Mimic application. When user click Identify Speech in the Home page, it will show the Identify page that consist of title, information registered phonemes, 2 elements such as input and audio, and 3 buttons such as Record, Identify Speech, and Finish.

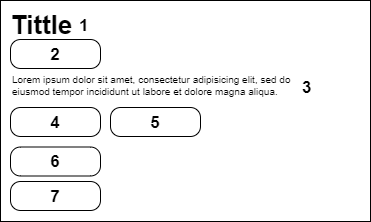


Figure 4.4 Mimic Identify page.

Table 4.4 Mimic Identify page description.

|  |  |
| --- | --- |
| **No** | **Description** |
| 1 | Title |
| 2 | Input Element |
| 3 | Information |
| 4 | Audio Element |
| 5 | Record Button |
| 6 | Identify Button |
| 7 | Finish Button |

#### Generate Page

Generate page is the other main pages of the Mimic application. When user click Generate Speech in the Home page, it will show the Generate page that consist of title, 3 elements such as datalist, textarea, and audio, and 2 buttons such as Generate Speech, and Finish.

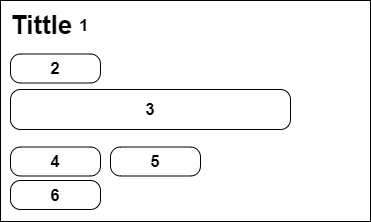


Figure 4.5 Mimic Generate page.

Table 4.5 Mimic Generate page description.

|  |  |
| --- | --- |
| **No** | **Description** |
| 1 | Title |
| 2 | Datalist Element |
| 3 | Textarea Element |
| 4 | Audio Element |
| 5 | Generate Button |
| 6 | Finish Button |

## Class Diagram

The class diagram is the structure of the system used toward this research. The class diagram in this research consist of Collect, Train, and Mimic application, server that handle Collect and Mimic application, and database that handle connection to MongoDB.

### Collect Application

The core classes in the Collect application are home and phonemes. They are the front-end of Collect application.

#### Home

home is HTML that render the Home page. It consists html tags based on the UI design on the previous section and JavaScript to handle the buttons.

#### Phonemes

phonemes is HTML that render the Phonemes page. It consists html tags based on the UI design on the previous section, JavaScript to handle the buttons, and JavaScript to handle the record that will be uploaded.

The upload handler contains file, randomString, and XMLHttpRequest methods. file is used to create wav blob from the record. randomString is used to randomize string for naming the record file and reducing duplicate that will overwrite one file to another. XMLHttpRequest is used to send upload request to the server and receive response from the server.

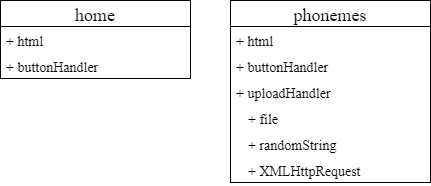


Figure 4.6 Collect application class diagram.

### Train Application

The core classes in the Train application are train and model. They are the code to train the machine learning model with collected data.

#### Model

model is JavaScript that define model to be Convolutional Neural Network model that will be used by train. It contains add method. add is used to add layer to the model.

#### Train

train is JavaScript that setup and train the model based on model. It contains loadData, nextBatch, saveModelDB, fit, and test methods. loadData is used to load collected data from Collect application, split to train, validation, and test data, and extract all of them. nextBatch is used to get portion of loadData result and convert it to model input output. saveModelDB is used to insert the model to MongoDB. fit is used to train the model with train data and validate with validate data. test is used to test the model with test data.

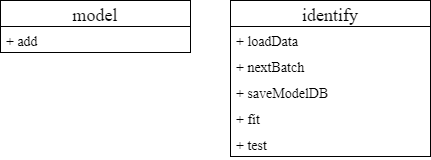


Figure 4.7 Train application class diagram.

### Mimic Application

The core classes in the Mimic application are home, identify, and generate. They are the front-end of Mimic application.

#### Home

home is HTML that render the Home page. It consists html tags based on the UI design on the previous section and JavaScript to handle the buttons.

#### Identify

identify is HTML that render the Identify page. It consists html tags based on the UI design on the previous section, JavaScript to handle the elements and buttons, JavaScript to handle the record that will be uploaded, and JavaScript to handle the record that will be identified.

The upload handler contains file, randomString, and XMLHttpRequest methods. file is used to create wav blob from the record. randomString is used to randomize string for naming the record file and reducing duplicate that will overwrite one file to another. XMLHttpRequest is used to send upload request to the server and receive response from the server.

The identify handler contains XMLHttpRequest methods. XMLHttpRequest is used to send identify request to the server and receive response from the server.

#### Generate

generate is HTML that render the Generate page. It consists html tags based on the UI design on the previous section, JavaScript to handle the elements and buttons, JavaScript to handle the speech data that will be loaded from database, and JavaScript to handle the speech ID and text that will be generated to audio.

The load handler contains XMLHttpRequest methods. XMLHttpRequest is used to send load request to the server and receive response from the server.

The generate handler contains XMLHttpRequest methods. XMLHttpRequest is used to send generate request to the server and receive response from the server.

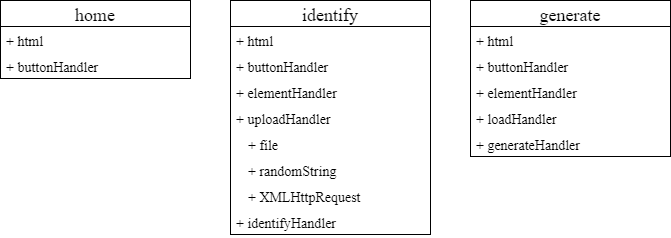


Figure 4.8 Mimic application class diagram.

### Server

The core classes in the server are serverHandler and createServer. They are the back-end of Collect and Mimic application.

#### serverHandler

serverHandler is JavaScript that define server listener that will be used by createServer. It consists of JavaScript to handle upload from Collect application, JavaScript to handle upload and identify from Mimic Identify page, JavaScript to handle load and generate from Mimic Generate page, and JavaScript to handle URL that will display and load in the browser.

The collect handler contains router and upload methods. router is used to check whether the request is point to collect handler or not. If yes, then upload will be executed. upload is used to upload file based on the request. Then, convert its upload path location to URL. It returns the URL as response.

The identify handler contains router, upload, and identify methods. router is used to check whether the request is point to identify handler or not. If yes, then upload or identify will be executed. upload is used to upload recorded file based on the request, then convert its upload path location to URL. It returns the URL and upload path as response. identify is used to extract recorded file based on the request and load the latest trained machine learning model from database. Then, the extract result is test with the model. If the test result show high accuracy the file path is inserted to MongoDB based on the speech ID in the request and the identify result. It returns the information whether the model can identify or not as response.

The generate handler contains router, load, and generate methods. router is used to check whether the request is point to generate handler or not. If yes, then load or generate will be executed. load is used to load all the speech ID from speech data in the MongoDB. It returns the result as response. generate is used to extract text based on the request and load the speech data from speech ID based on the request. Then, the extract result is check to the loaded speech data. If unmatched is found, it returns error as response. If all matched, the speech data corresponding to extract result is combined to 1 audio file. Finally, the file path is converted to URL and returns the result as response

#### createServer

createServer is JavaScript that create the server. It contains listen method. listen is used to listen any request that comes to the server. The request will be proceeded to serverHandler.

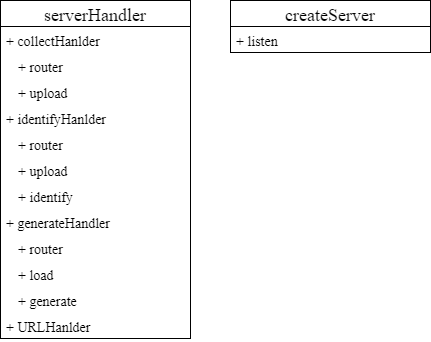


Figure 4.9 Server class diagram.

### Database

The core classes in the database are connection and model. They are the code that create connection to MongoDB and collection model scheme of the database on MongoDB.

#### connection

connection is JavaScript that handle the connection with MongoDB. It contains connect and disconnect methods. connect is used to establish the connection to MongoDB. disconnect is used to close the established connection.

#### model

model is JavaScript that create the collection model scheme of the database on MongoDB. It contains speechDatas and models schemes. speechDatas is used to define model for speech data collection scheme that is used on Mimic application. Each collection record speech ID and phonemes. models is used to define model for machine learning model collection scheme that is used on Train and Mimic application. Each collection record model’s path and the date the model is saved.

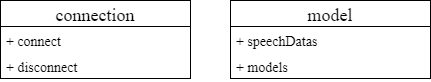


Figure 4.10 Database class diagram.

# SYSTEM DEVELOPMENT

## User Interface Development

The UI is applied on Collect and Mimic application. There are two core pages in Collect application and three core pages in Mimic Application.

### Collect Application

Two core pages in Collect application are Home page and Phonemes page.

#### Home Page

Home page is the home page of the Collect application. It shows information about how the collecting process works and information regarding to the collecting process. There are also 2 buttons, Start and Mimic Home button.

Start button will direct user to Phonemes page to start collecting. Mimic Home button will direct user to Mimic application Home page.

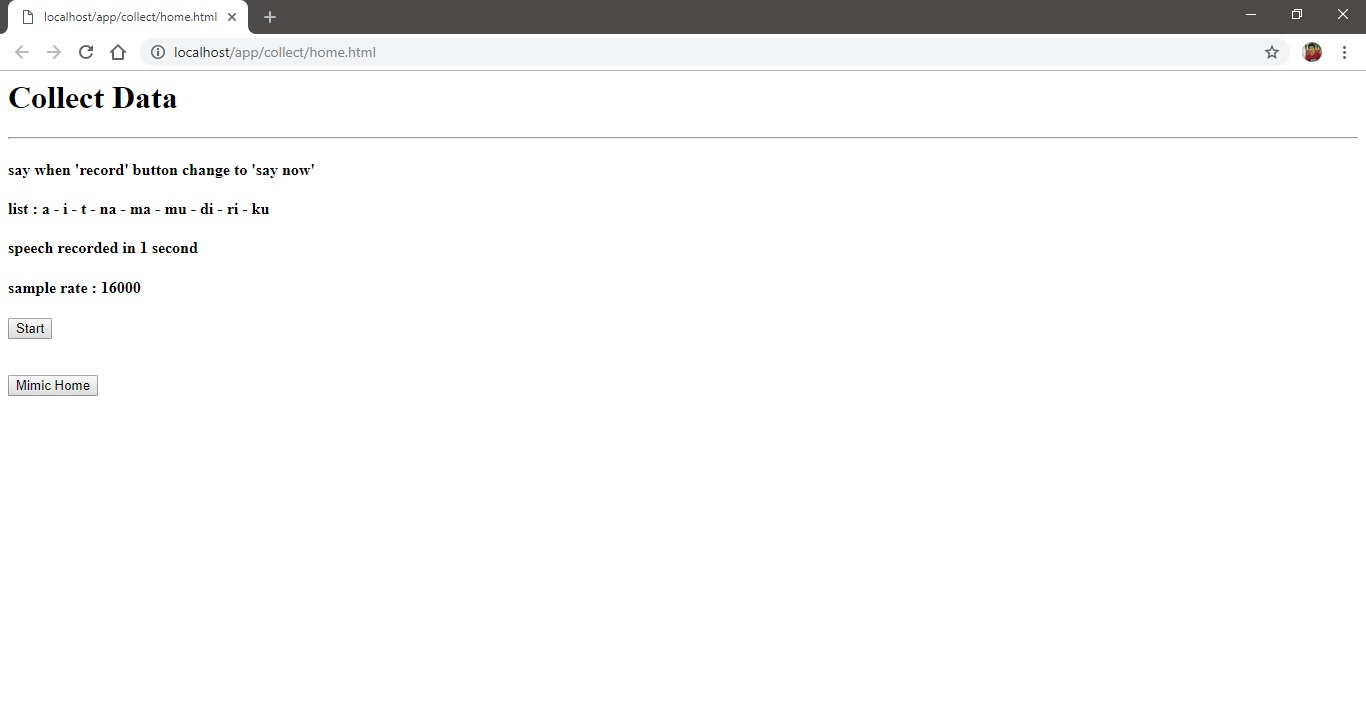


Figure 5.1 Collect Home page.

#### Phonemes Page

Phonemes page is the main page of the Collect application. It shows information about which phoneme should be recorded and replayed the record. There are also 2 buttons, Record and Next button and an Audio element. Any response from the server is also printed in the console.

Record button will start the record process. When the record process is starting, Record button will be renamed to ‘Say Now’ which tell user that the record process is starting and indicate user to say the relevant phoneme. When the record process is finish, the recorded blob is uploaded to the server and Record button will be renamed to ‘Record Again’ which tell user that the record process is finish and user can do the next record. Next button will direct user to the next Phonemes page. In this application there are 10 Phonemes. On the last phoneme, Next button will direct user back to Home page.

Audio element will be shown when the record process is finish with an audio from the record. User can hear the replay of the record from the Audio element.

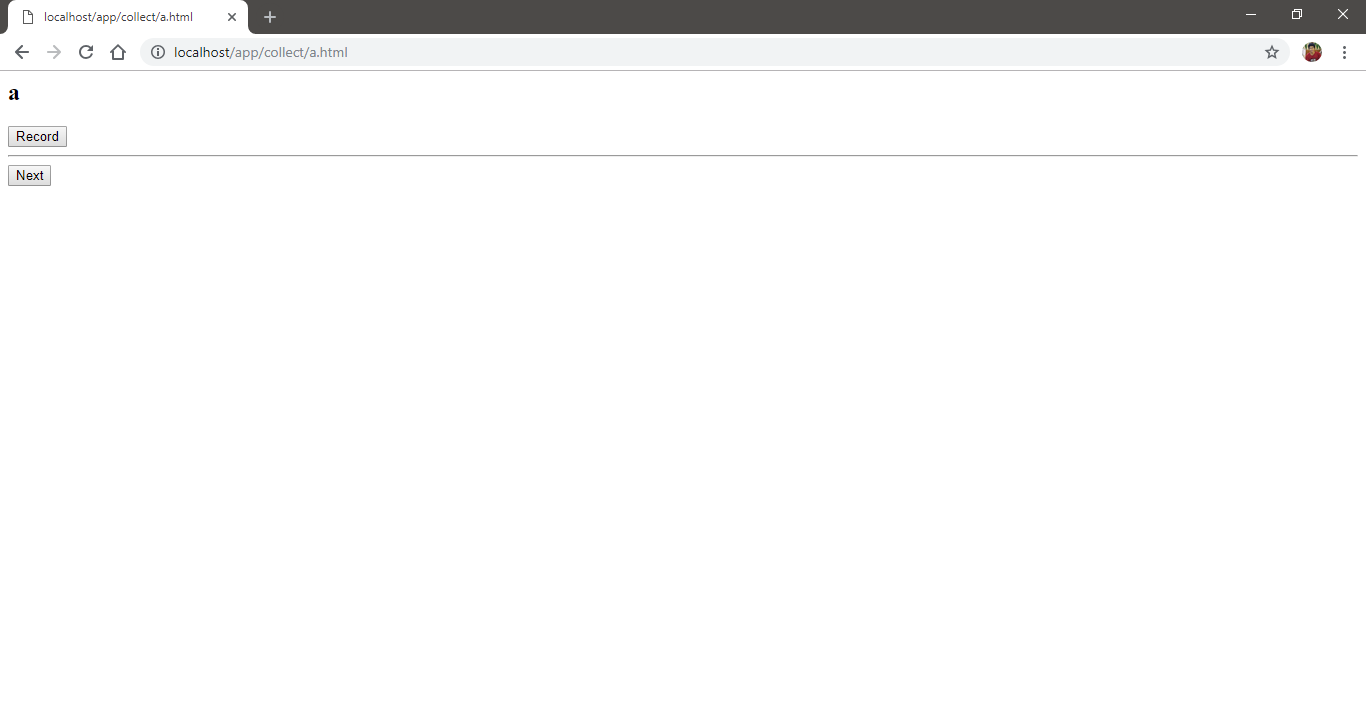


Figure 5.2 Collect Phonemes page with ‘a’ phoneme.

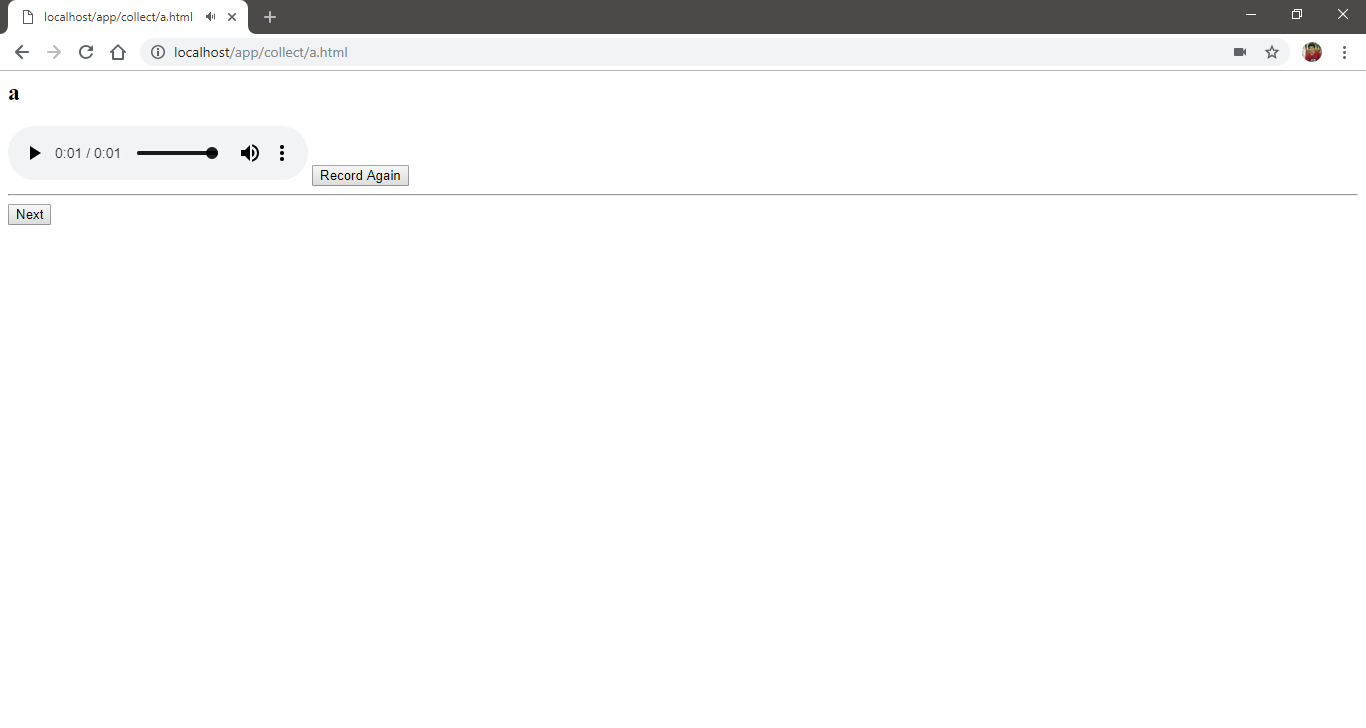


Figure 5.3 Collect Phonemes page after record process is finish.

### Mimic Application

Three core pages in Mimic application are Home page, Identify page, and Generate page.

#### Home Page

Home page is the home page of the Mimic application. It shows information about how the mimic process works and information regarding to the identify and generate process. There are also 3 buttons, Identify, Generate, and Collect Home button.

Identify button will direct user to Identify page to start identifying speech. Generate button will direct user to Generate page to start generating speech. Collect Home button will direct user to Collect application Home page.

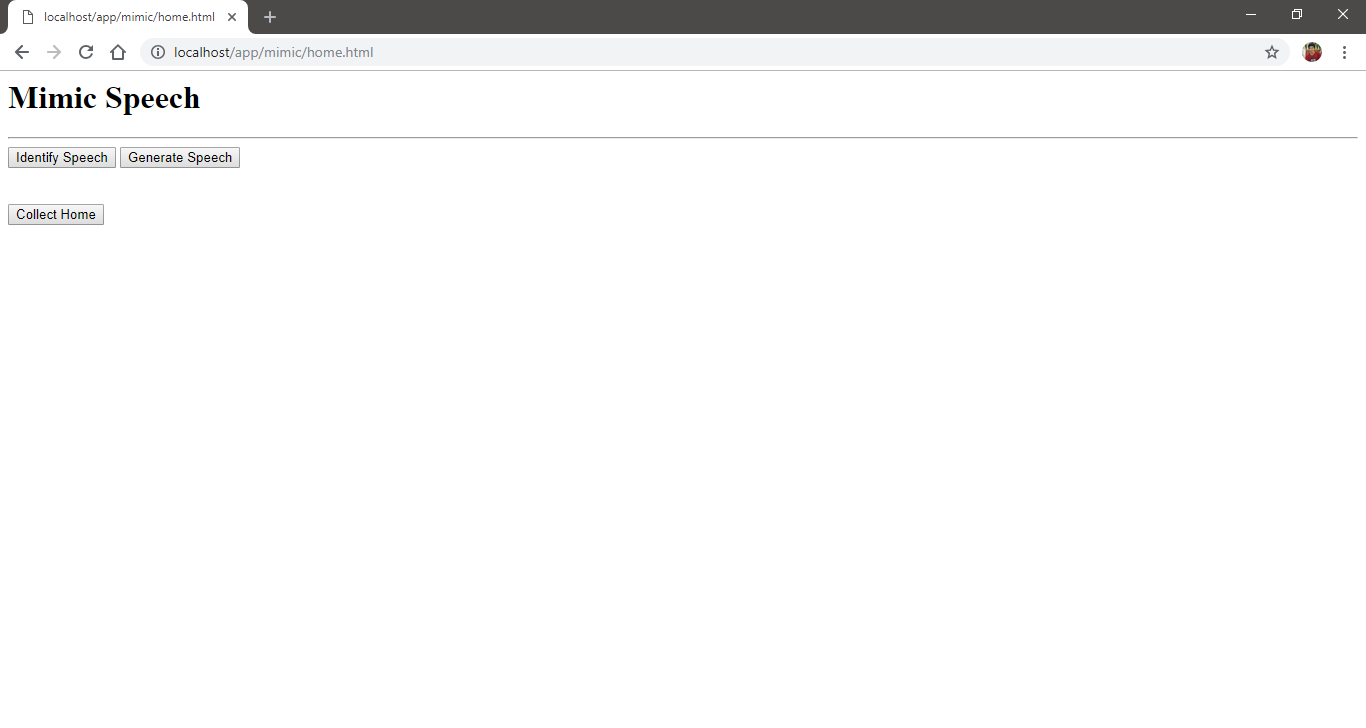


Figure 5.4 Mimic Home page.

#### Identify Page

Identify page is the one of the main pages of the Mimic application. It shows information about which phoneme is recognizable and form of input and audio that will be used to identify the speech and update it to database. There are also 3 buttons, Record, Identify Speech and Finish button, and an Input text and an Audio element. Any response from the server is also printed in the console.

Record button will start the record process. When the record process is starting, Record button will be renamed to ‘Say Now’ which tell user that the record process is starting and indicate user to say the phoneme. When the record process is finish, the recorded blob is uploaded to the server Record button will be renamed to ‘Record Again’ which tell user that the record process is finish and user can do the next record. Identify Speech button will take the value of Input element and file source of Audio element, then, send them to the server to be identify. The indication of identify process in the server is success or error is there is alert on the browser that show any information regarding to identify process in the server. If there is no value of Input element or file source of Audio element, Identify Speech button will alert user that there is not any value. Finish button will direct user back to the Home page.

Audio element will be shown when the record process is finish with an audio from the record. User can hear the replay of the record from the Audio element. Input element is used as input of user speech ID value.

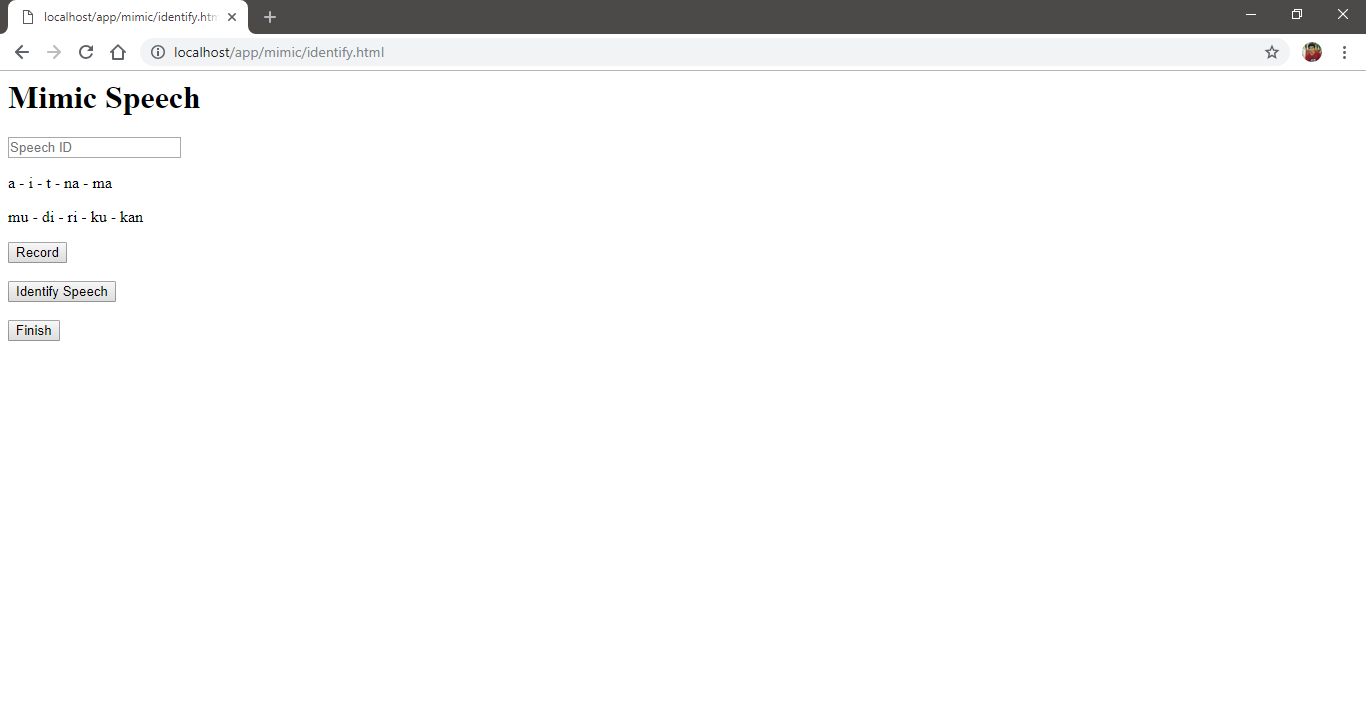


Figure 5.5 Mimic Identify page.

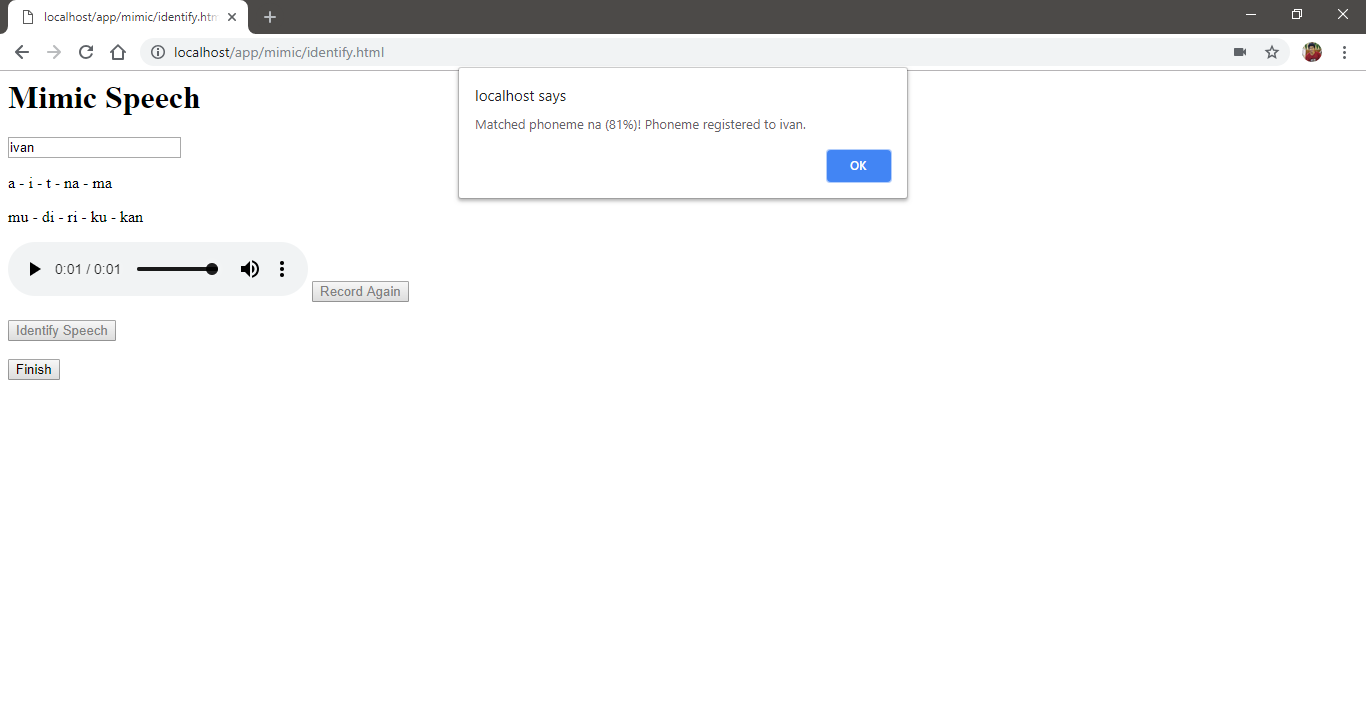


Figure 5.6 Mimic Identify page alert user indicating identify process in the server is finish.

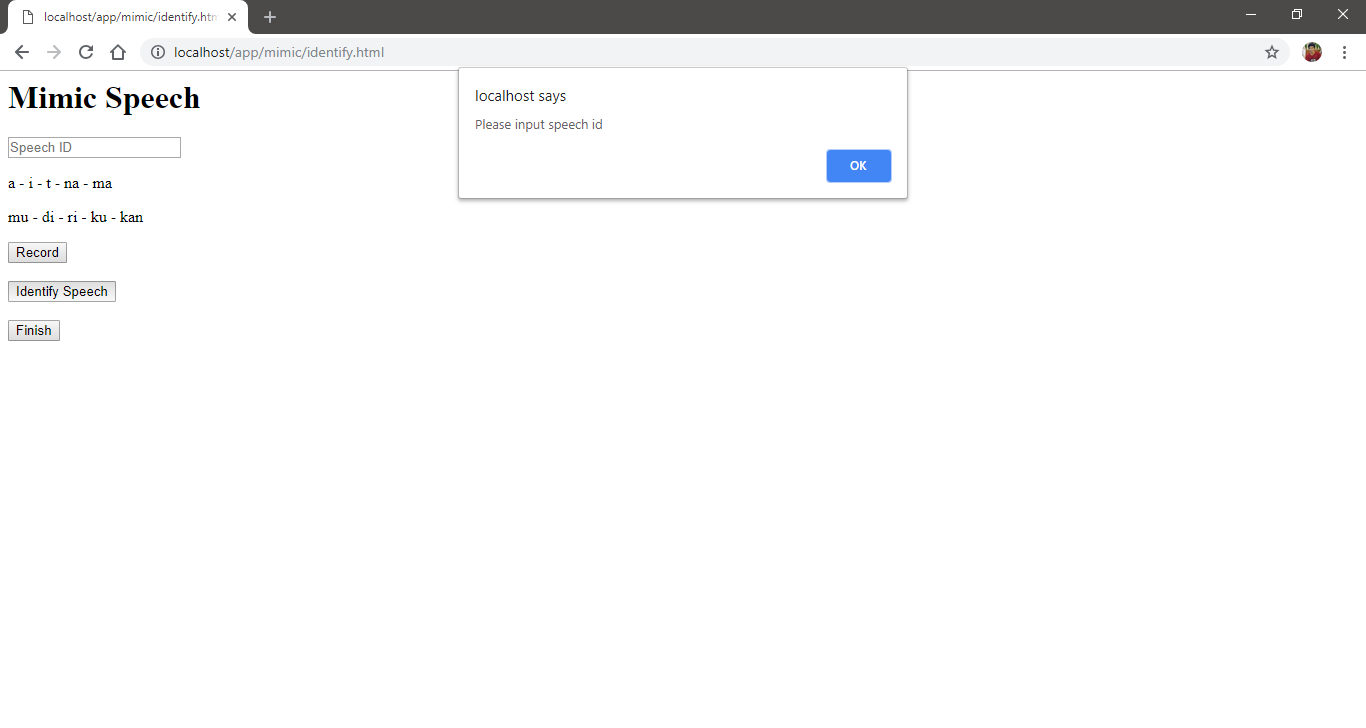


Figure 5.7 Mimic Identify page alert user indicating there is no value in Input element.

#### Generate Page

Generate page is the one of the main pages of the Mimic application. It shows form of data list and text area that will be used to generate the speech. When the page is loading, it also loads speech data from the database and then store it on data list. If there is no speech data found, it will alert user and direct user back to Home page. There are also 2 buttons, Generate Speech and Finish button, and a Datalist element, a Textarea element and an Audio element. Any response from the server is also printed in the console.

Generate Speech button will take the value of Datalist element and Textarea element, then, send them to the server to be generate. The indication of generate process in the server is error is there is alert on the browser that show any error information regarding to generate process in the server. The indication of generate process in the server is success is the Audio element is shown with an audio from the generated speech. If there’s no value of Datalist element or Textarea element, Identify Speech button will alert user that there is not any value. Finish button will direct user back to the Home page.

Audio element will be shown when the generate process is success with an audio from the generated speech. User can hear the generated speech from the Audio element. Datalist element is shows all speech data from database and it is used as input of user speech ID value. Textarea element is used as input of words that user want to generate.

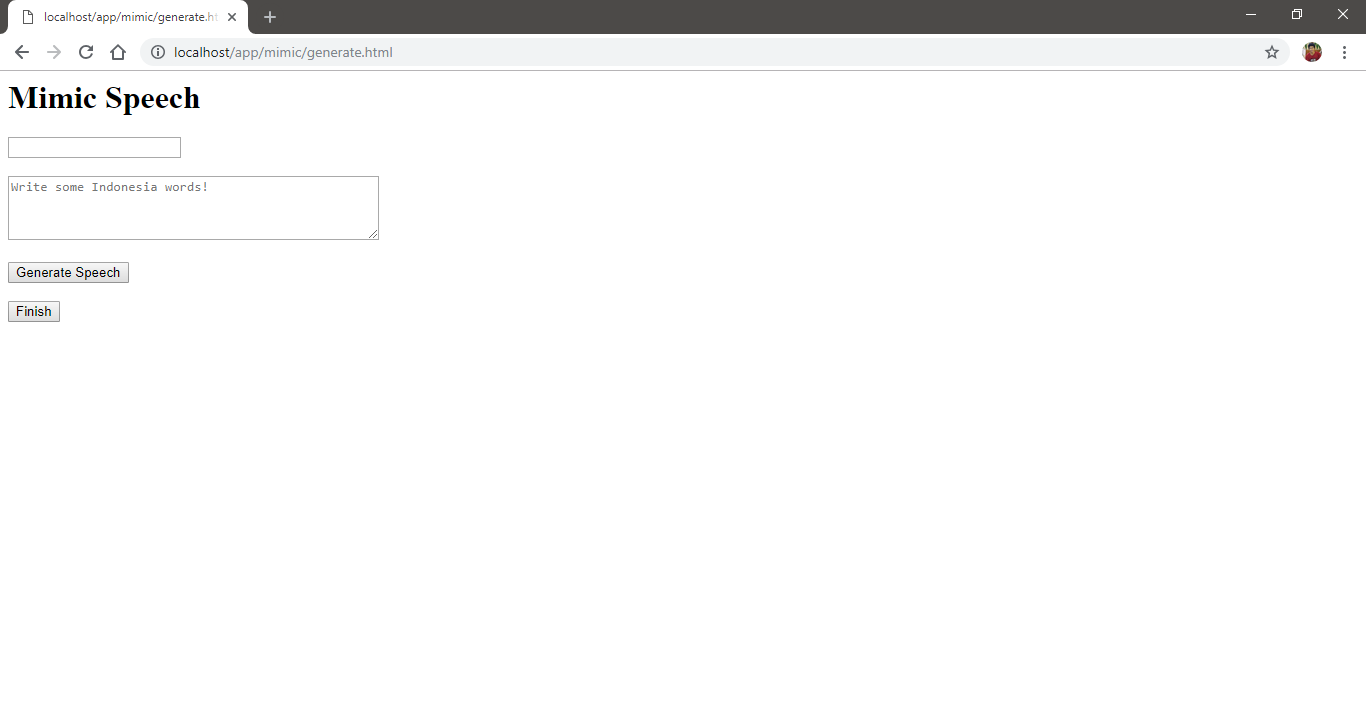


Figure 5.8 Mimic Generate page.

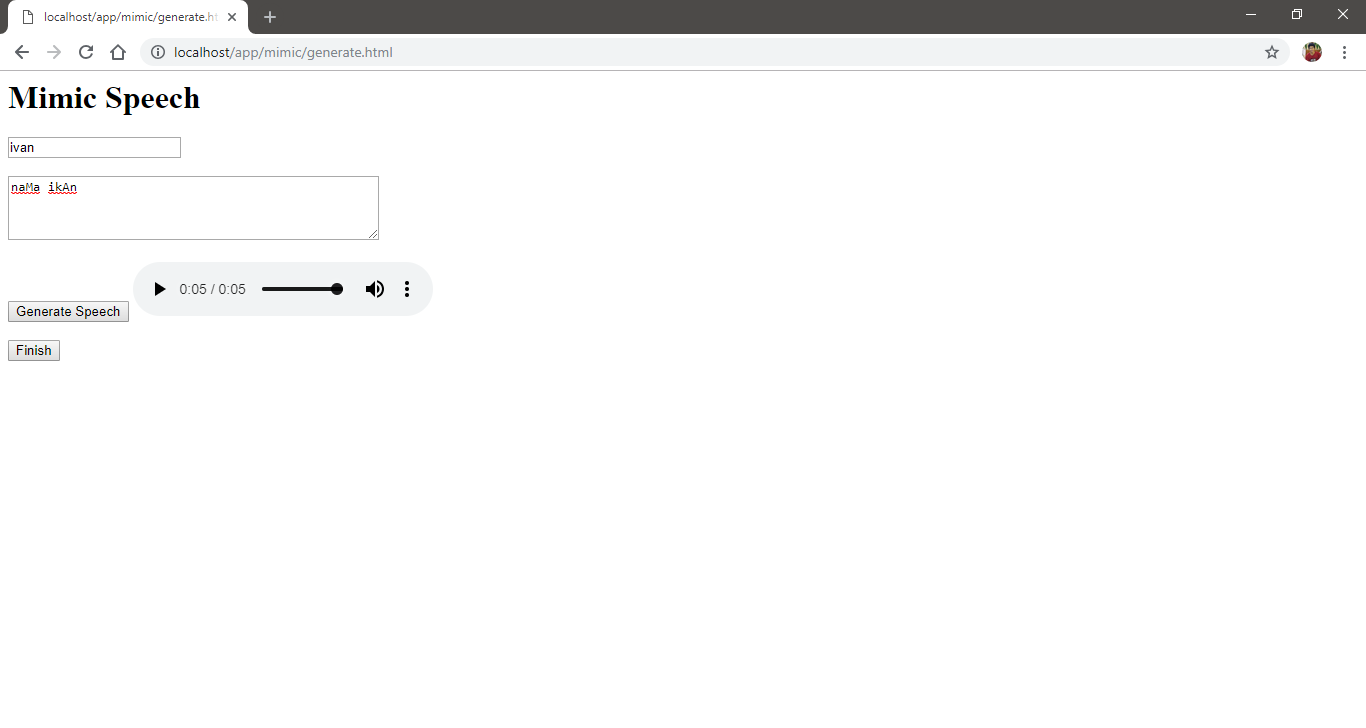


Figure 5.9 Mimic Generate page play the generate speech after generate process success.

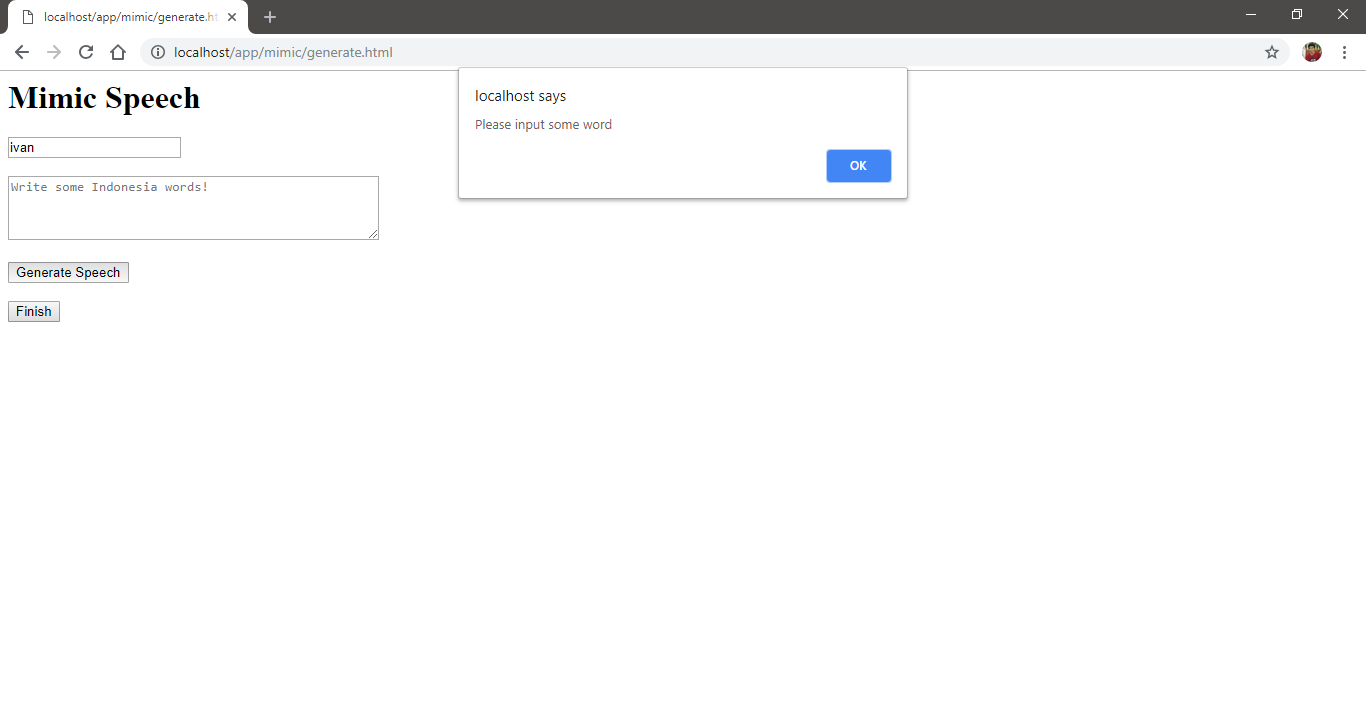


Figure 5.10 Mimic Generate page alert user indicating there is no value in Textarea element.

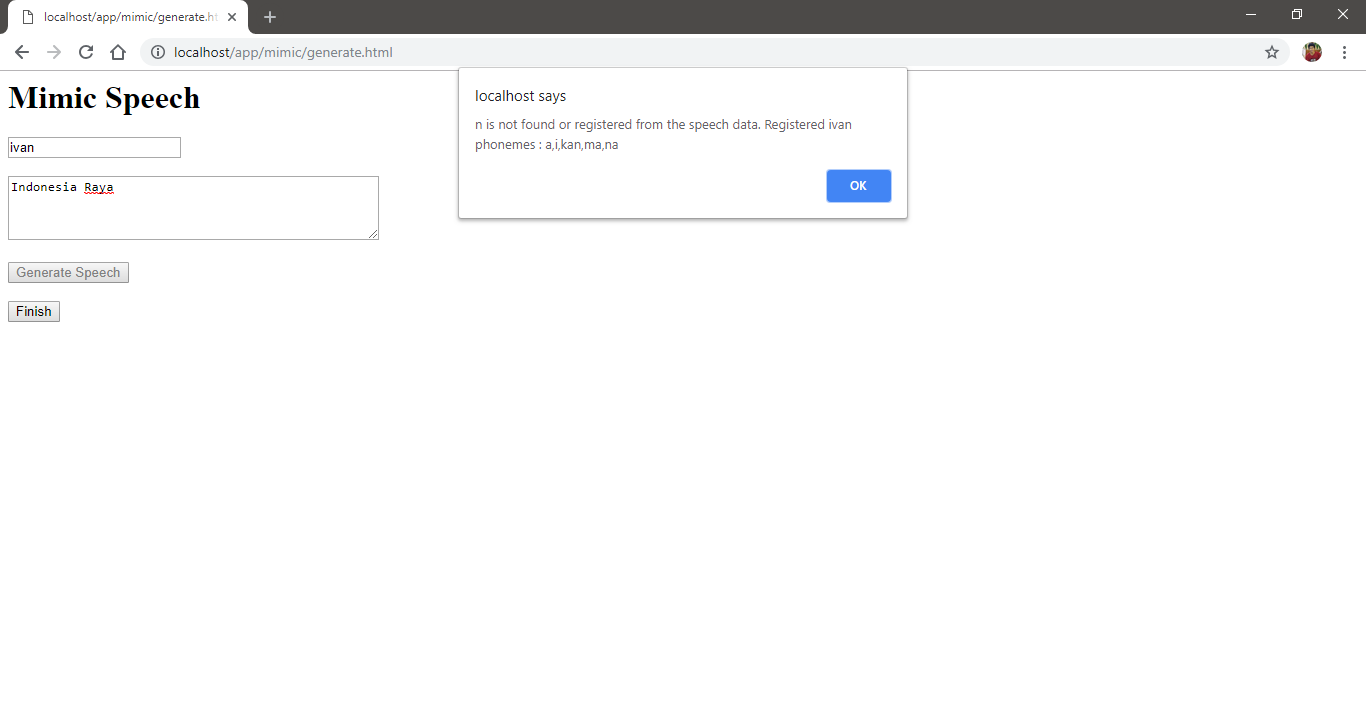


Figure 5.11 Mimic Generate page alert user indicating there is error in the generate process.

## Application Details

The application details explain on server code, router code, database code, Train application code and library package that is used in the application. The figures that show the code also included comment that help explanation for corresponding line.

### Server Code

The server code is used to create http server in the computer. It contains serverHandler and create server command itself.

serverHandler handle all request to the server each response corresponding to each request. There are 3 type main router collect, identify, generate to handle any request corresponding to them. When the request match it returns 200 alongside with the response corresponding to the router. When the request doesn't match any of those, it checks if the file URL as filename is existed in working directory or not. When the filename not found it return 404 Not Found. When found it read the file and return it with 200 so that it can be displayed by the browser. It returns 500 if error occurred when reading the file. used to create http server in the computer core pages in Mimic application are Home page, Identify page, and Generate page.

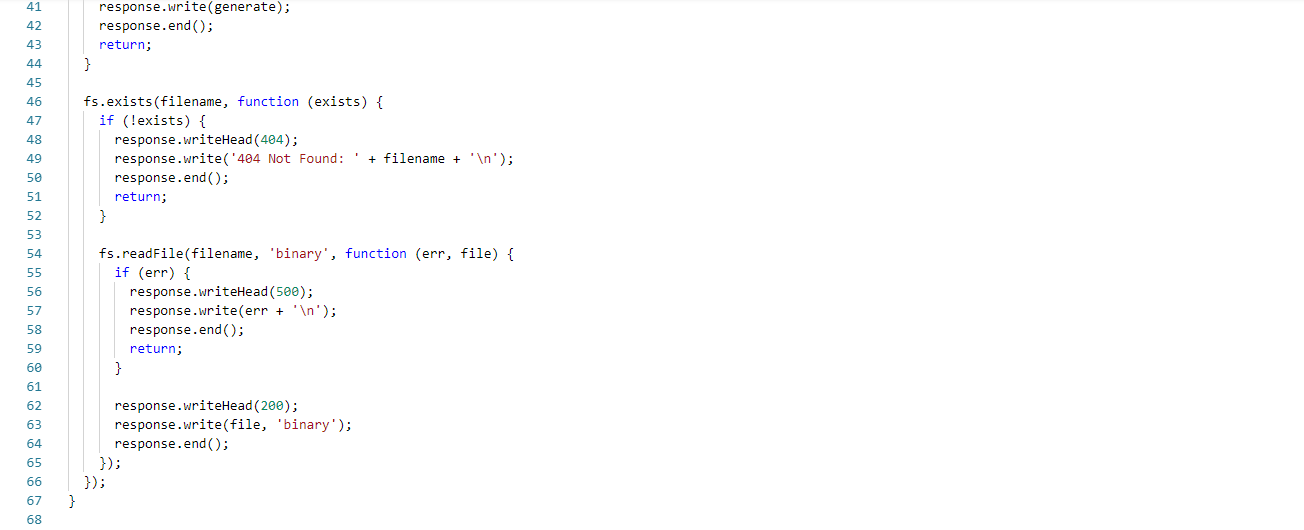
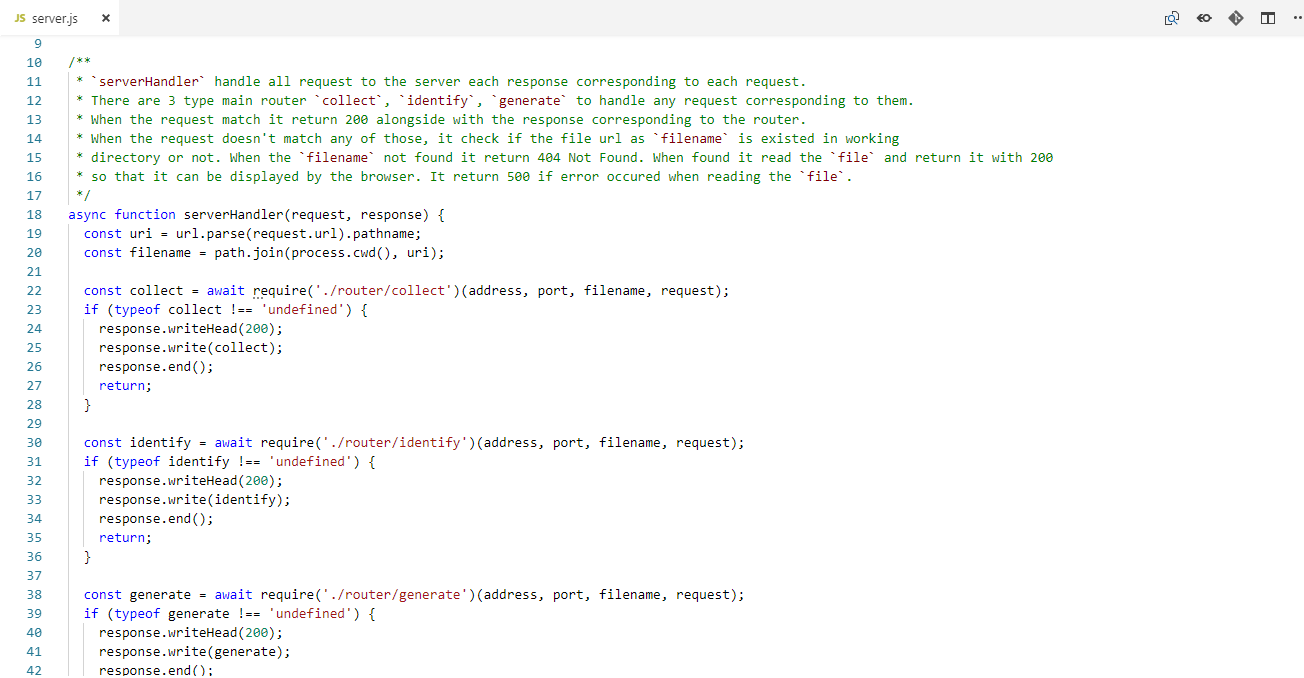


Figure 5.12 serverHandler on Server code.

Create server command accept serverHandler as the parameter. It will listen on defined variable localhost and port. When the server established, it prints information in console.

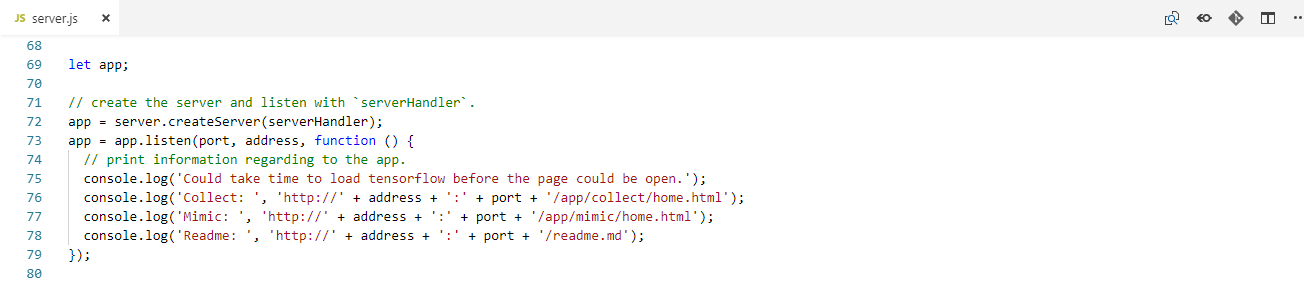


Figure 5.13 Create server command on Server code.

### Router Code

The router is used to handle categorized request from the server. It split into 3 categorized files, collect, identify, and generate.

Collect router handle all collect app request, each response corresponding to each request. There are a lot of request categorized by collect app but all of them is basically uploadCollect. When the request doesn't match anything, it returns nothing. uploadCollect upload file in the request based on defined directory and then convert its path location to URL. It returns the URL as fileURL.

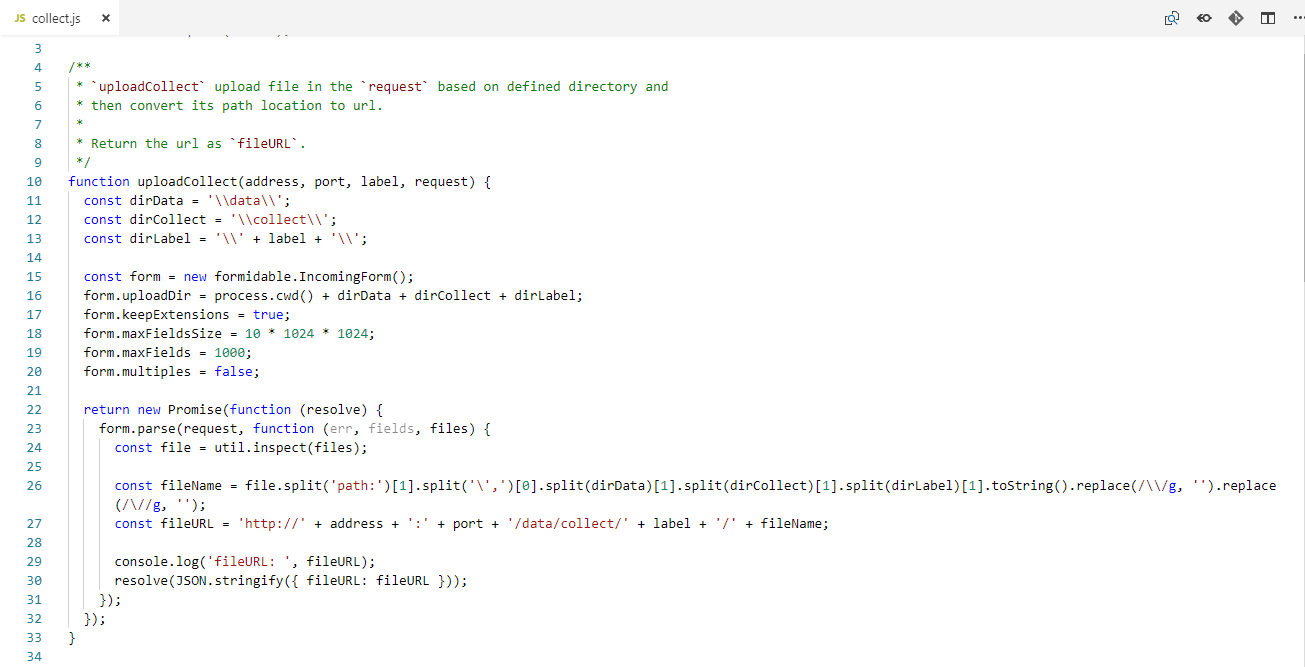


Figure 5.14 uploadCollect on Collect Router code.

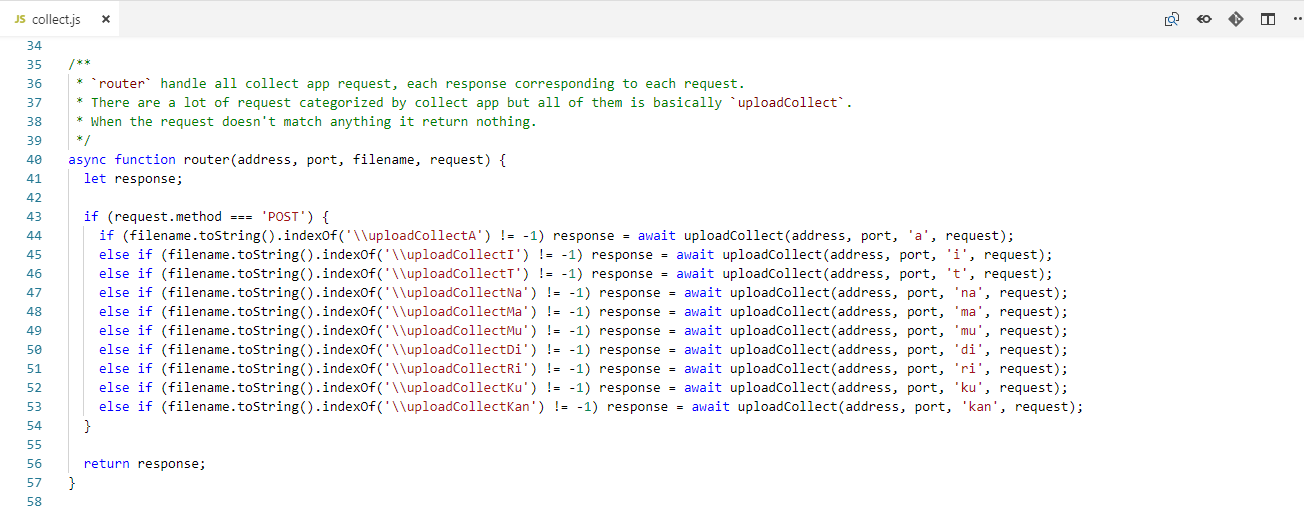


Figure 5.15 router on Collect Router code.

Identify router handle all mimic app identify section request, each response corresponding to each request. There are 2 type requests categorized by mimic app identify section, uploadSpeech and identifySpeech. When the request doesn't match anything, it returns nothing. uploadSpeech upload file in the request based on defined directory and then convert its path location to URL. It returns the URL and path location as fileURL and filePath. identifySpeech load latest inserted trained model from MongoDB models collection. Use it to identify the speech file from filePath by extract then feed it to model. Matched speech will register the phoneme to corresponding name and update it on MongoDB. Error connection will return message. Not found model will return message. Unsatisfied identification, results below 0.75 (75%) from model will return message. None of those is occurred will update the MongoDB speechDatas collection on its name and identified phoneme filePath and return message. The returned message whether just information or even error is return as status. extract is support function decode wav based on filePath, frame it, extract by FFT and MFCC and convert it to TensorFlow input shape. It returns 4d tensor.

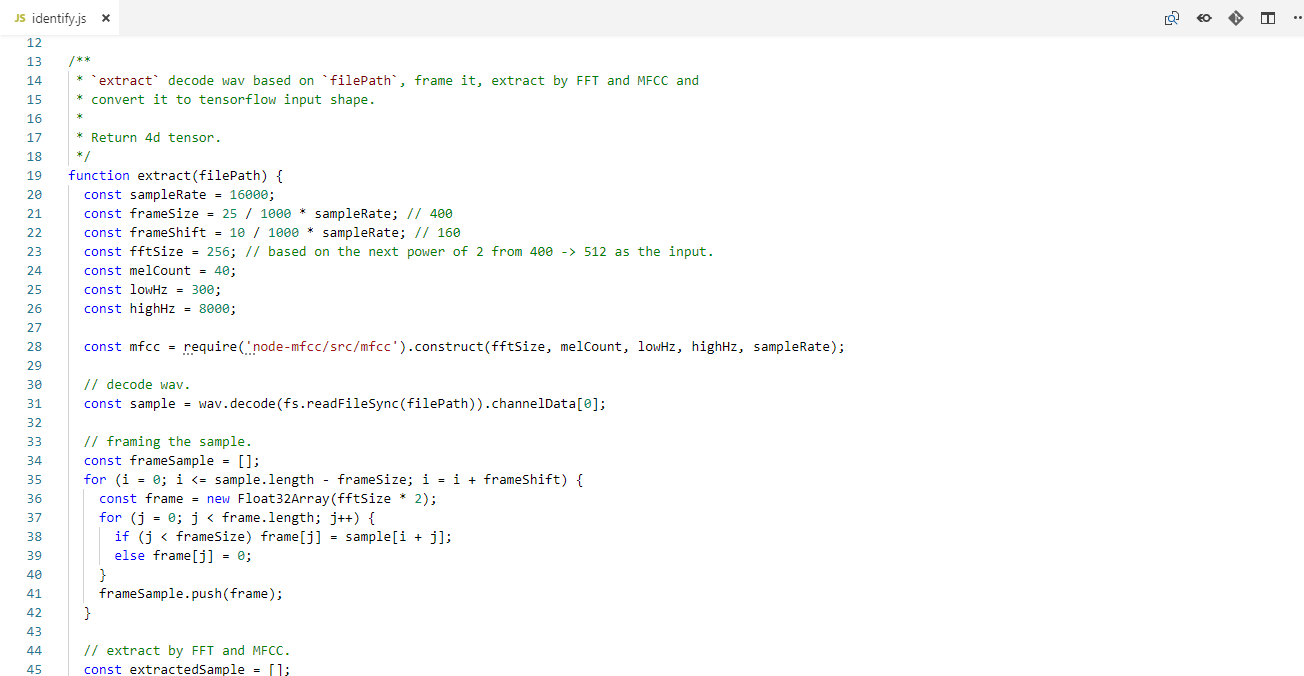




Figure 5.16 extract on Identify Router code.



Figure 5.17 uploadSpeech on Identify Router code.

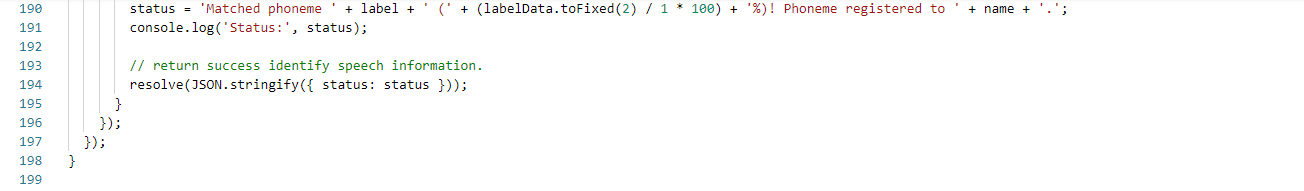
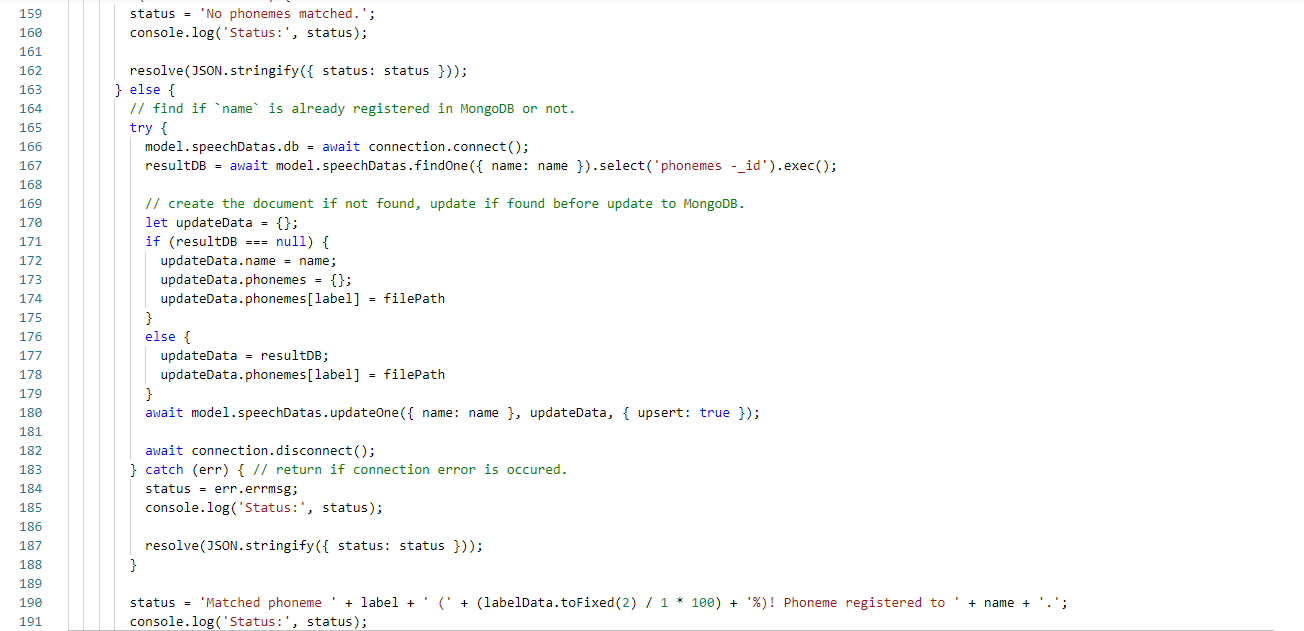
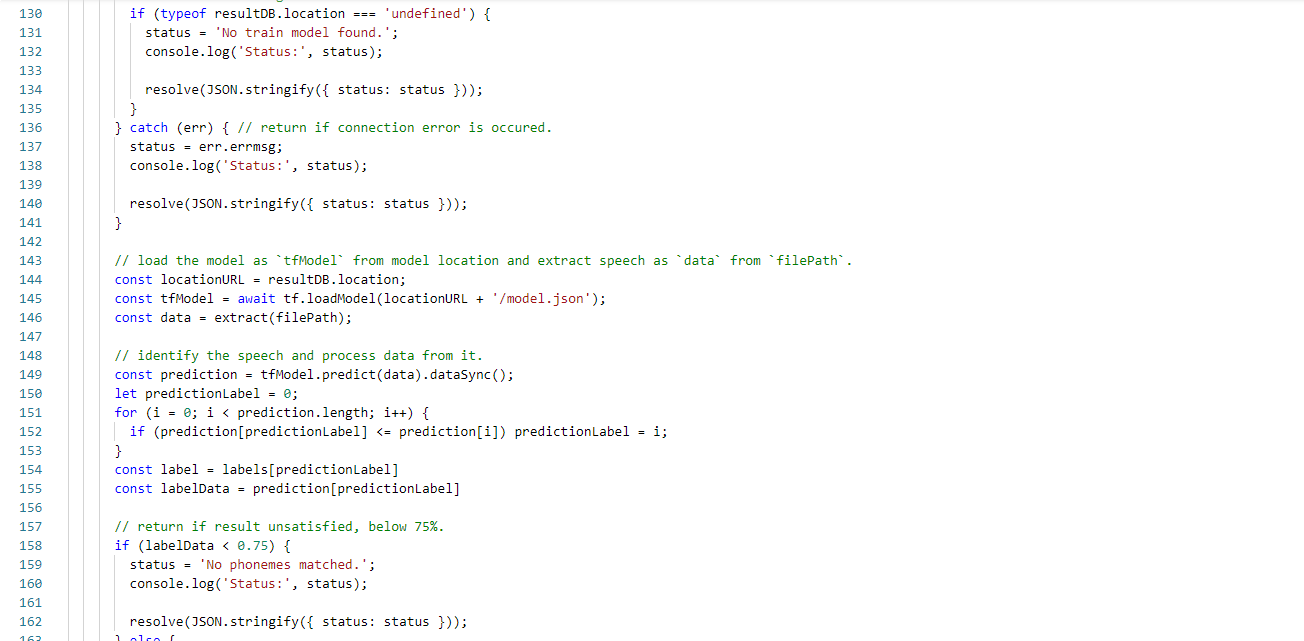
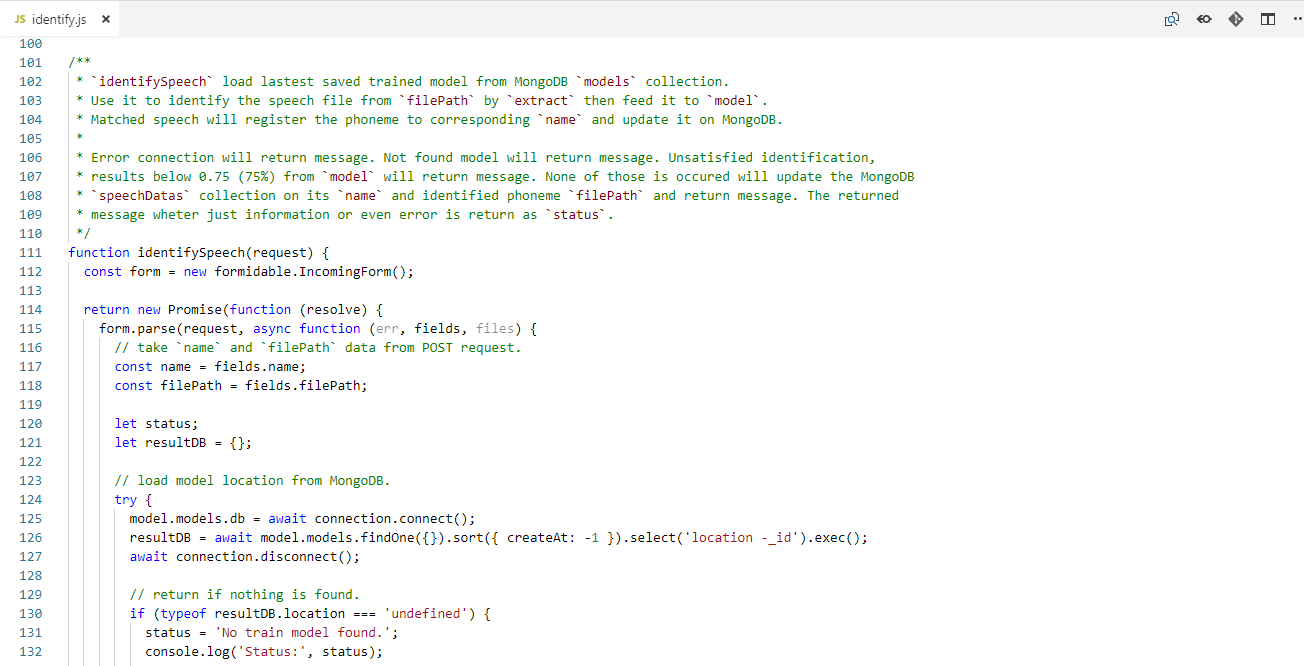


Figure 5.18 identifySpeech on Identify Router code.

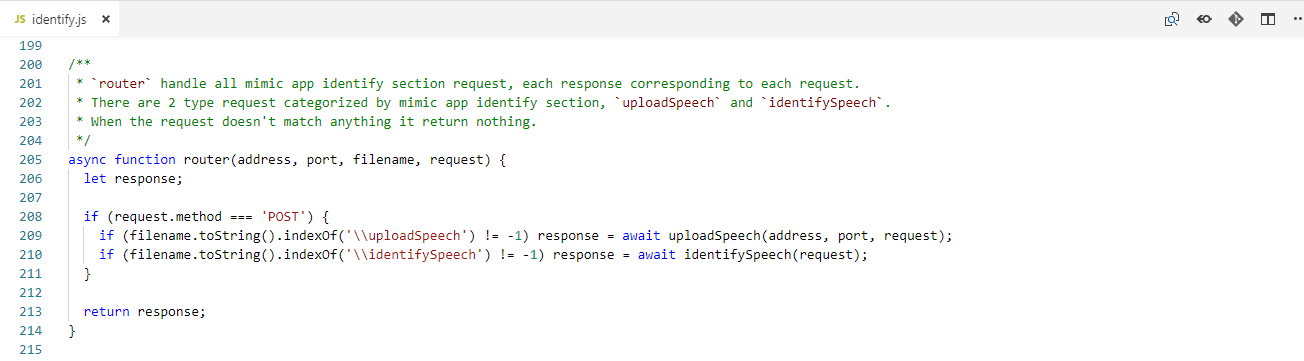


Figure 5.19 router on Identify Router code.

generate router handle all mimic app generate section request, each response corresponding to each request. There are 2 type requests categorized by mimic app identify section, loadSpeech and generateSpeech. When the request doesn't match anything, it returns nothing. loadSpeech load all registered name from MongoDB speechDatas collection. It returns the array of name as data and error message if nothing is found or error. generateSpeech load registered phonemes from MongoDB speechDatas collection based on name. Extract words as extractWord. Decode all coresponding extractWord based on the database result. them and encode to wav file and take its fileURL location. It returns the URL as fileURL and error message if connection error or unregistered phoneme is found.

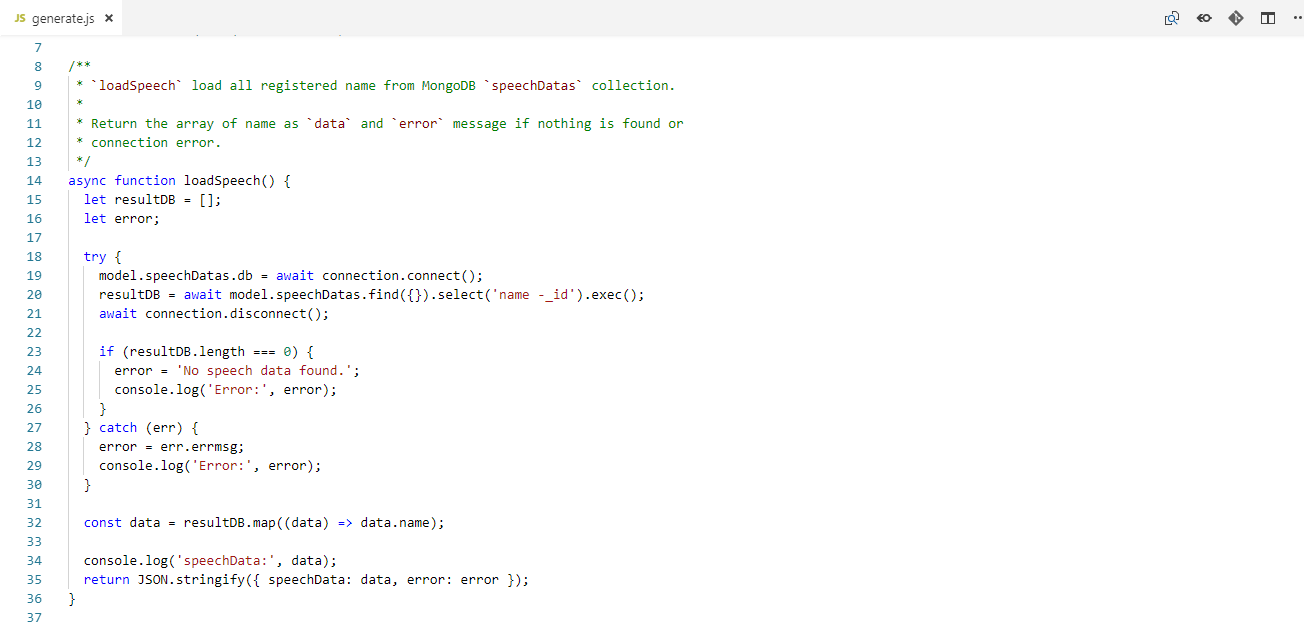


Figure 5.20 loadSpeech on Generate Router code.

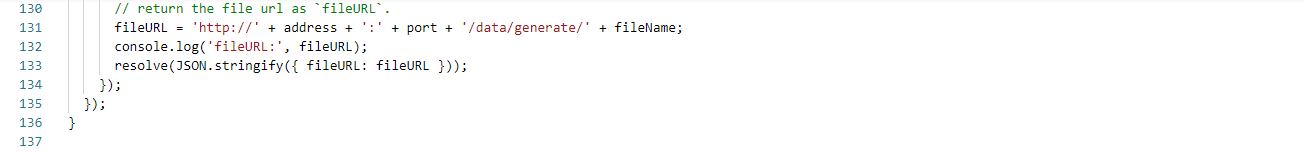
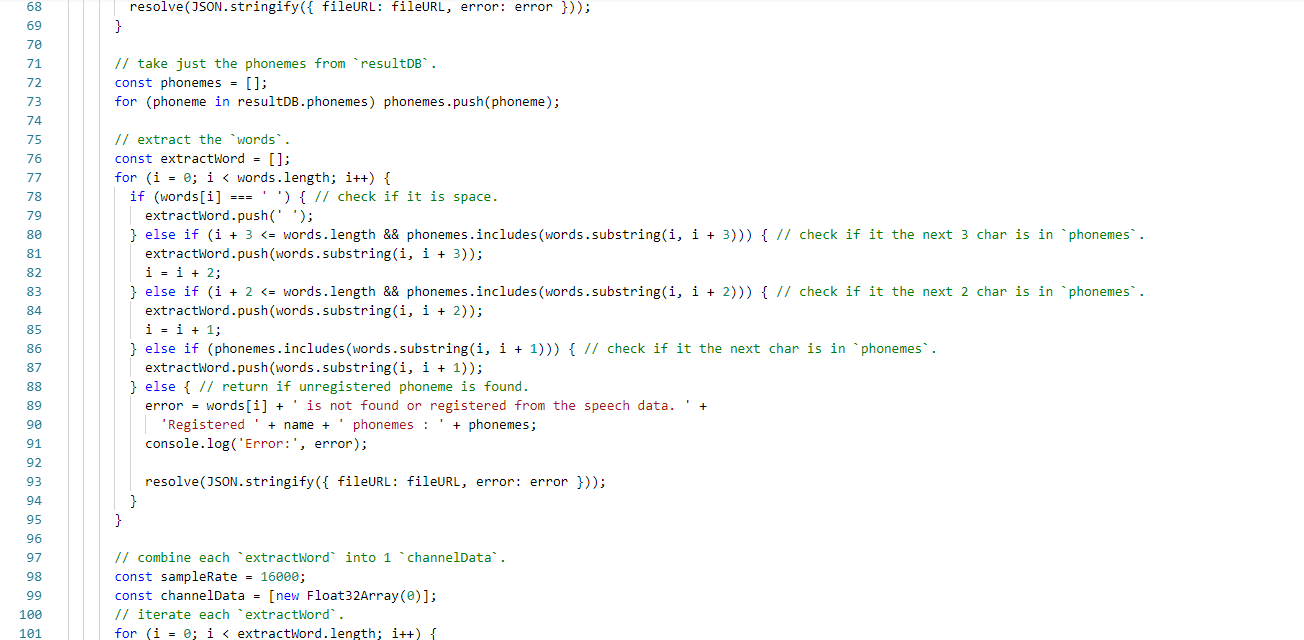
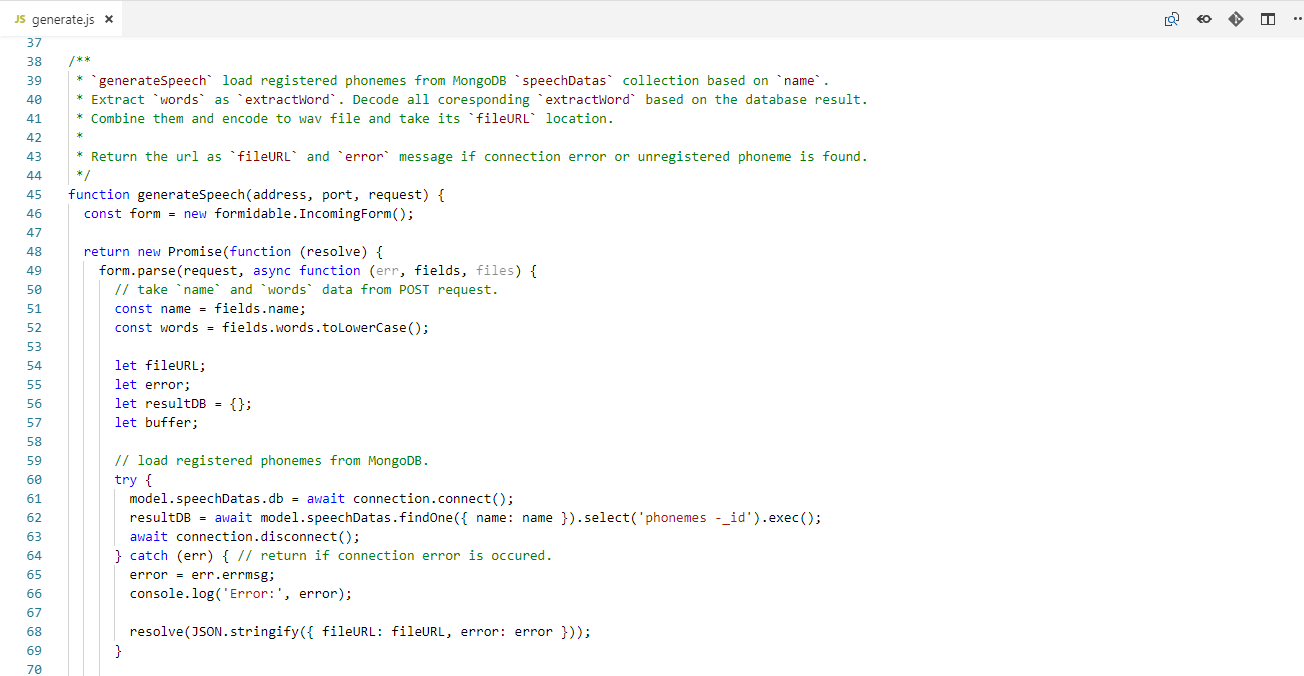


Figure 5.21 generateSpeech on Generate Router code.



Figure 5.22 router on Generate Router code.

### Database Code

The database code is used to create connection to MongoDB and create MongoDB collection model scheme. It contains connection and model.

connection define 2 functions connect and disconnect. connect establish connection to MongoDB mimic\_speech database on localhost. Established connection is saved in connection. disconnect close established connection to MongoDB.

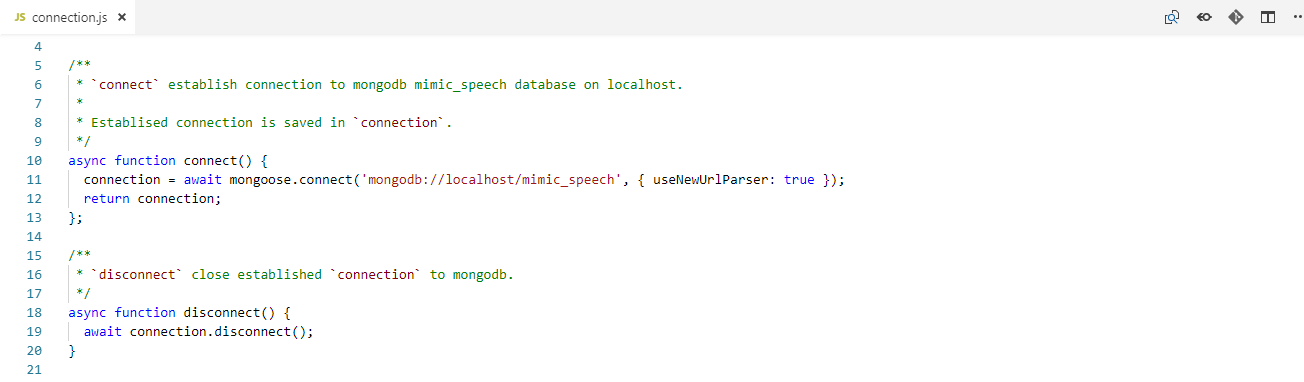


Figure 5.23 connection on Database code.

model define 2 schemes variable speechDatas and models. speechDatas define scheme with name as string and phonemes as object. models define scheme with location as string and createAt as date with current date being default value when create or update collection when createAt is not defined.

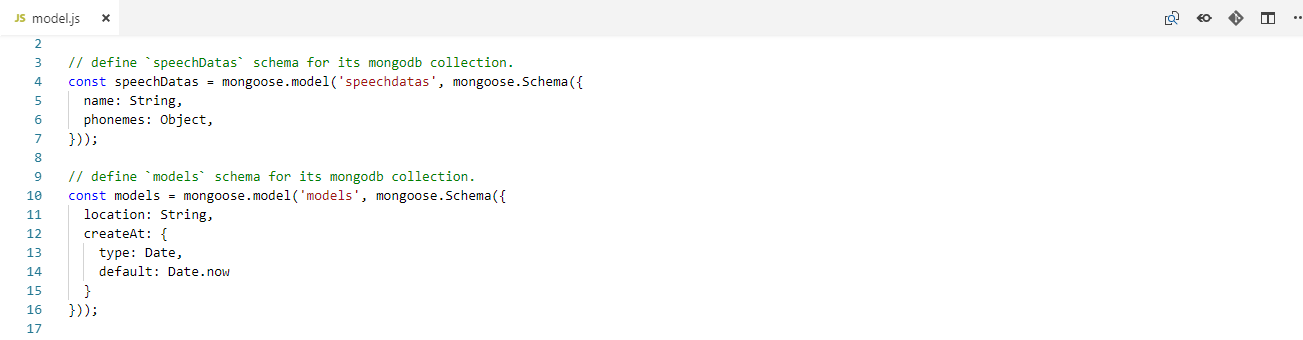


Figure 5.24 model on Database code.

### Train Application Code

Train application code is used to train model with collected data and save the model on n iteration and last iteration so that later can be used to identify the speech. It contains extractWav, loadData, nextBatch, saveModelDB as support code and model and train as the main code.

extractWav decode wav based on file and label location, frame it and extract by FFT and MFCC. It returns JSON of extracted sample as data and label.

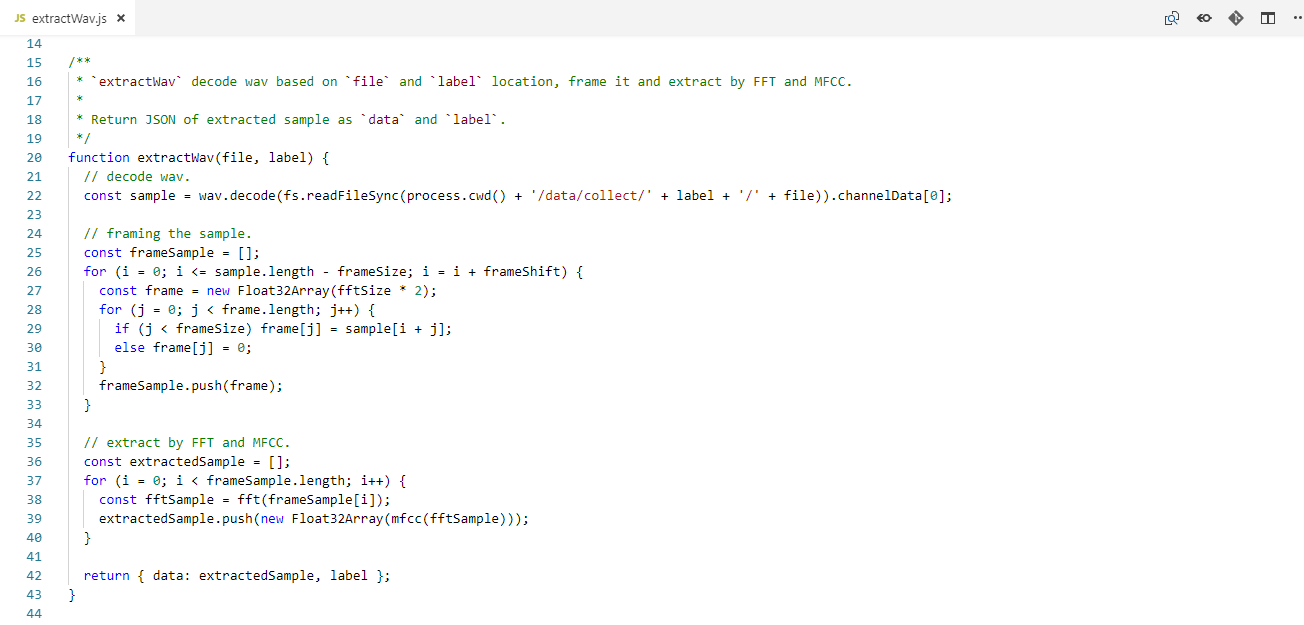


Figure 5.25 extractWav on Train application code.

loadData is used to load all collected data, then, extract them and split them into train data, validation data, and test data. http server in the computer. It contains extractFiles and load and split command itself. extractFiles extract wav from loaded files as load, assigned to corresponding data. It returns array of extracted waves. Every iteration when extracting the loaded files is printed. If inconsistent data is met, the process stops and exit.

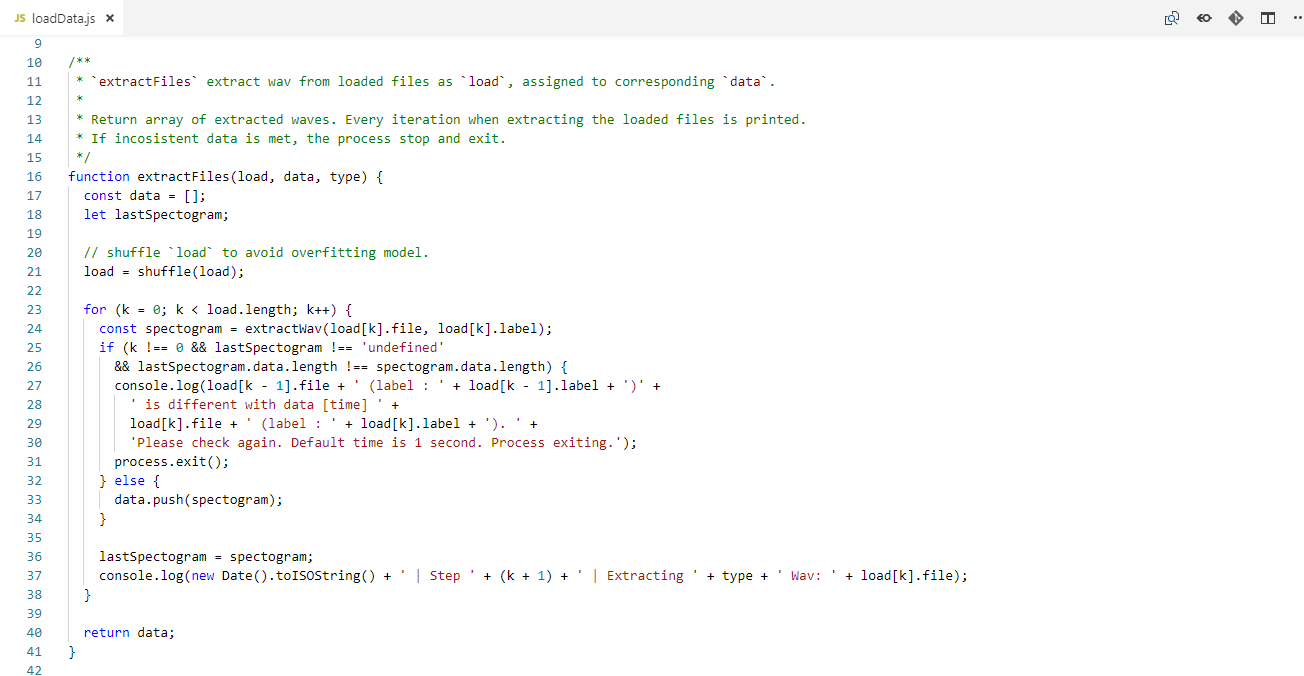


Figure 5.26 extractFiles on loadData Train application code.

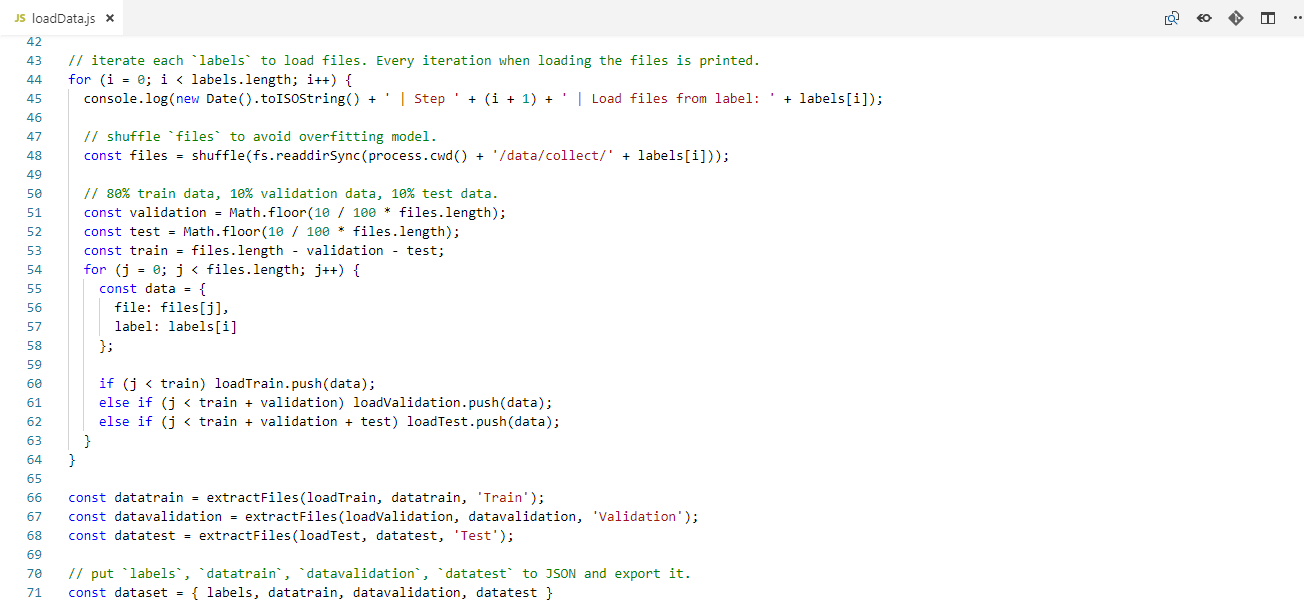


Figure 5.27 load and split command on loadData Train application code.

nextBatch is used to get the next batch data and then convert it to train model input or output. It contains getData and getLabel. getData convert corresponding dataset data based on iteration, batch, time, freq to TensorFlow input. It defines TensorFlow input shape from batch, time, and freq. It defines TensorFlow input data convert it by rearrange the data inside dataset based on batch, iteration, time, and freq. Then it returns as 4d tensor. getLabel convert corresponding dataset label based on iteration, batch to TensorFlow output. It defines TensorFlow output shape from batch, and labels.length. It defines TensorFlow output data convert it by define array from labels based on batch, iteration, and dataset. Then it returns as 2d tensor.

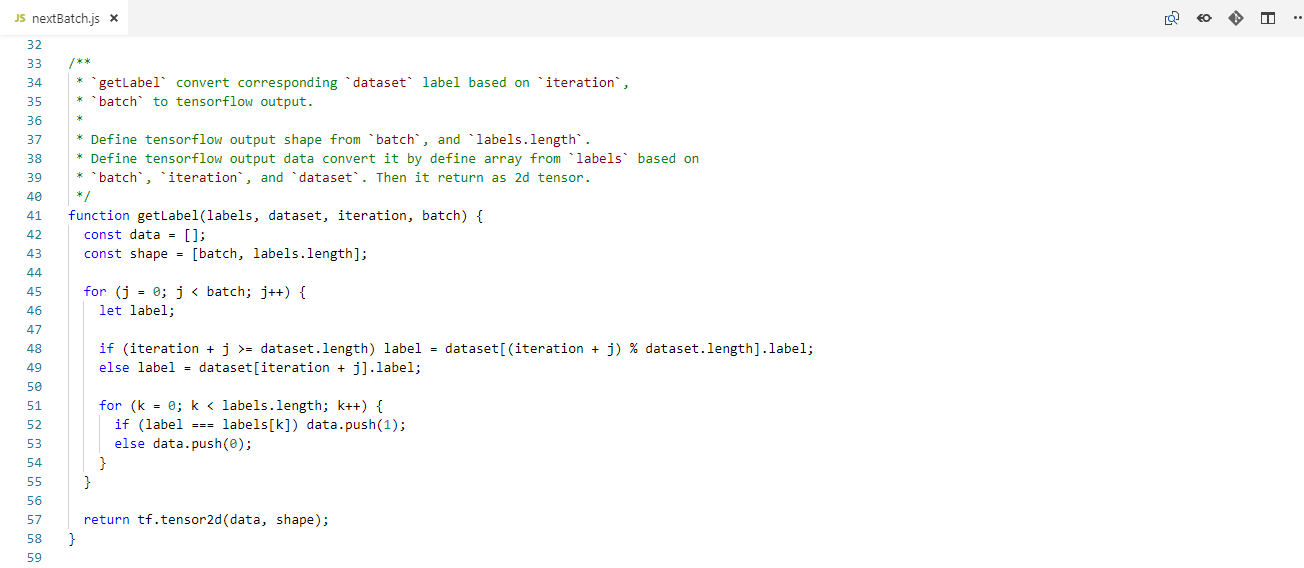


Figure 5.28 getData on nextBatch Train application code.

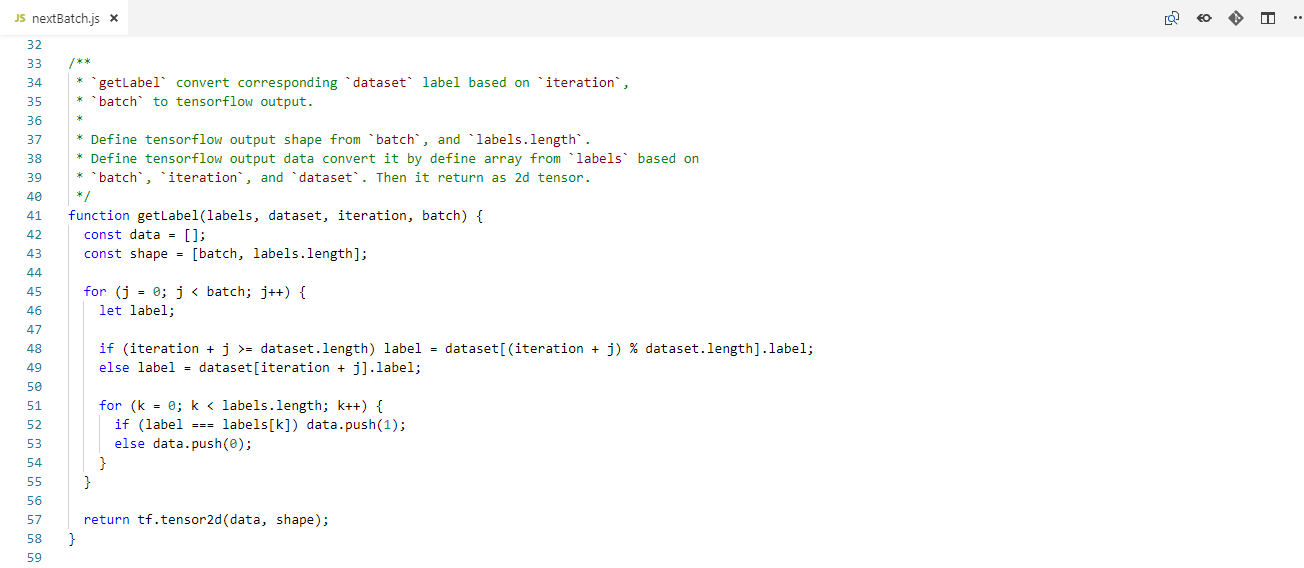


Figure 5.29 getLabel on nextBatch Train application code.

saveModelDB save location to models collection in mimic\_speech database in MongoDB.

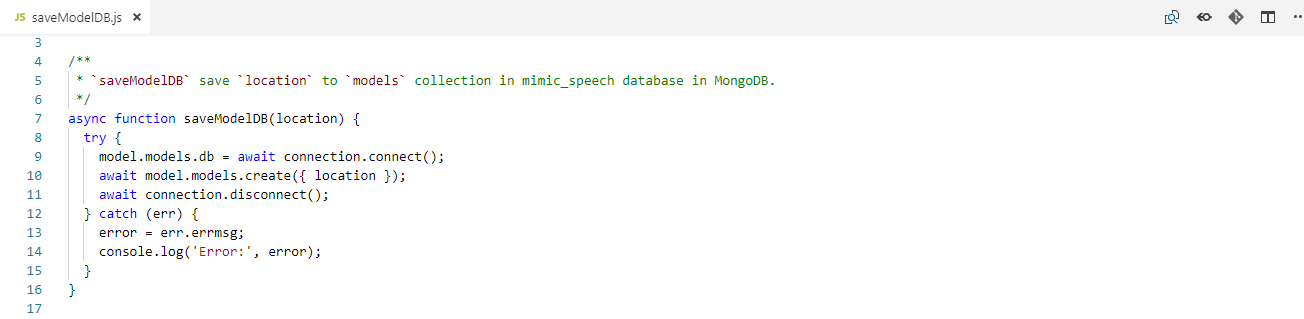


Figure 5.30 saveModelDB on Train application code.

model as modelML model the TensorFlow neural network. It is CNN model with cross entropy loss, and SGD (Stochastic Gradient Descent) optimizer.

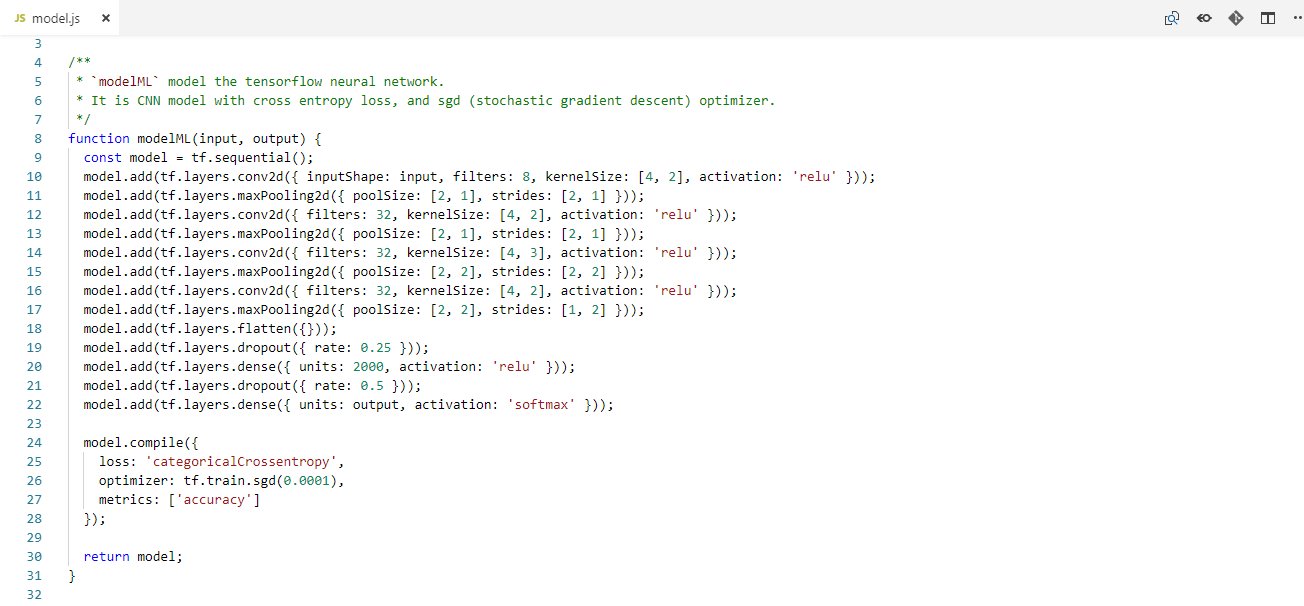


Figure 5.31 modelML on model Train application code.

train contains preparation command for train and train itself. train train the model to fit the dataset.datatrain and validate the model on dataset.datavalidation data by its loss and accuracy. The train iterate trainIteration times and test every testIteration in trainIteration. Every iteration loss and accuracy are printed. Every test is occurred and the last iteration, confusionMatrix is printed, model is saved, the saved model location is inserted to MongoDB, loss and accuracy is saved as csv.

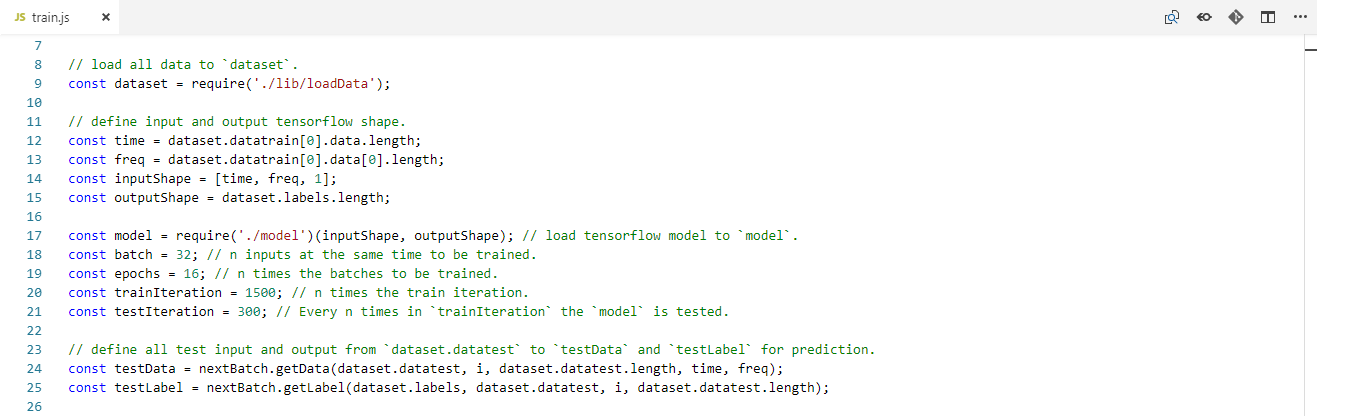


Figure 5.32 preparation command on train Train application code.

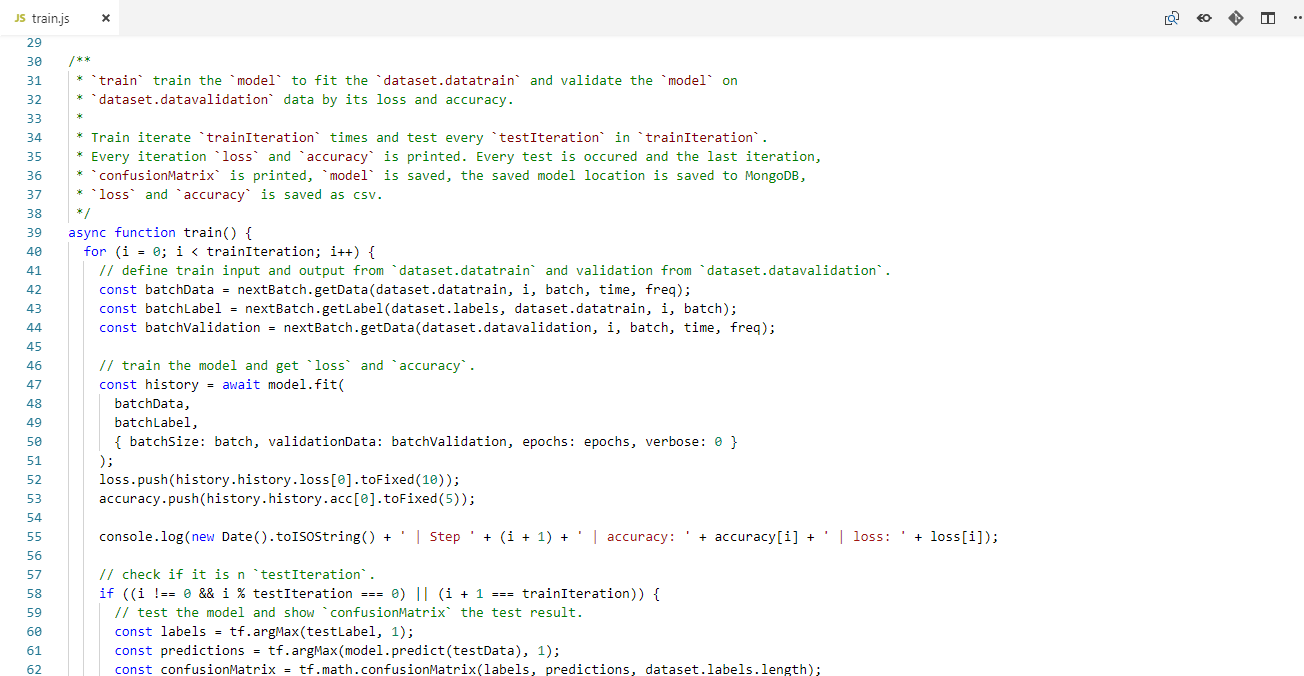


Figure 5.33 train on train Train application code.

### Library Package Code

# SYSTEM TESTING

## Testing Environment

The testing environment specify the environment during testing. The environment specification are as follows:

1. Windows 10.
2. Chrome browser.

There is additional environment only on the identify process. User that the records is trained by the application is called Tester 1. Male user that the record hasn’t trained by the application called Tester 2. Female user that the record hasn’t trained by the application called Tester 3. Every tester is test in each the following environment:

1. Noisy background (Loud music or people chit-chat).
2. Semi noisy background (AC noise or sound from the other rooms).
3. No noisy background.

## Testing Scenario

The testing scenario conducted by evaluating all the features with a set of cases or scenario in the application based on its functionality requirement and defined testing environment.

The testing scenarios of the application is categorized based on Collect application, Train application, and Mimic application.

### Collect Application

The Collect application has 3 subcategorized scenarios, Collect server, Home page and Phonemes page. The Collect server has scenarios to allow user to access Home page, Phonemes page in the browser and process the upload record request. The Home page has scenarios to allow user to direct to Phonemes page to start the Collect application and direct to Mimic application Home page to change to Mimic application. The Phonemes page has scenarios to allow user to upload the record to the server and direct to the next Phonemes page or direct back to Home page. Figures 6.1 shows some screenshot of expected result from the scenarios.

Table 6.1 Collect application scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Scenario** | **Expected Result** | **Result** |
| 1 | Access Home page | Home page is shown | As expected |
| 2 | Access Phonemes page | Phonemes page is shown | As expected |
| 3 | Process upload record request | Printed process results information on the console | As expected |
| 4 | Direct to Phonemes page | Directed to Phonemes page | As expected |
| 5 | Direct to Mimic application Home page | Directed to Mimic application Home page | As expected |
| 6 | Upload the record | Audio element is shown with the user record | As expected |
| 7 | Direct to the next Phonemes page or direct back to Home page | Directed to the next Phonemes page or Directed back to Home Page | As expected |

Figures 6.1 Some screenshot from the Collect application scenarios.

### Train Application

The Train application has scenario to allows user to train model with collected data. When train is run, it loads and splits collected data, extracts the data, gets the corresponding batch data, trains the TensorFlow model to fit the batch train data, tests the TensorFlow model from test data, and saves the TensorFlow model to database. Figures 6.2 shows some screenshot of expected result from the scenarios.

Table 6.2 Train application scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Scenario** | **Expected Result** | **Result** |
| 1 | Load and split collected data | Printed information on the console | As expected |
| 2 | Extract the data | Printed information on the console | As expected |
| 3 | Get corresponding batch data | Printed information on the console | As expected |
| 4 | Train the TensorFlow model to fit batch train data | Printed information on the console | As expected |
| 5 | Test the TensorFlow model from test data | Printed information on the console | As expected |
| 6 | Save the TensorFlow model to database | Recorded document in mimic\_speech database models collection | As expected |

Figures 6.2 Some screenshot from the Train application scenarios.

### Mimic Application

The Mimic application has 4 subcategorized scenarios, Mimic server, Home page, Identify page, and Generate page. The Mimic server has scenarios to allow user to access Home page, Identify page, and Generate page in the browser and process the upload record and identify speech on Identify request and load speech data and generate speech on Generate request. The Home page has scenarios to allow user to direct to Identify page to start the identify speech application, direct to Generate page to start the generate speech application, and direct to Collect application Home page to change to Collect application. The Identify page has scenarios to allow user to upload the record to the server, send form request for identify speech to the server, and direct back to Home page. The Generate page has scenarios to allow user to load speech data from the server, send form request for generate speech to the server, and direct back to Home page. Figures 6.3 shows some screenshot of expected result from the scenarios.

Table 6.3 Mimic application scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Scenario** | **Expected Result** | **Result** |
| 1 | Access Home page | Home page is shown | As expected |
| 2 | Access Identify page | Identify page is shown | As expected |
| 3 | Access Generate page | Generate page is shown | As expected |
| 3 | Process upload record on Identify request | Printed process results information on the console | As expected |
| 4 | Process identify speech on Identify request | Printed process results information on the console | As expected |
| 5 | Process load speech data on Generate request | Printed process results information on the console | As expected |
| 6 | Process generate speech on Generate request | Printed process results information on the console | As expected |
| 7 | Direct to Identify page | Directed to Identify page | As expected |
| 8 | Direct to Generate page | Directed to Generate page | As expected |
| 9 | Direct to Collect application Home page | Directed to Collect application Home page | As expected |
| 10 | Upload the record | Audio element is shown with the user record | As expected |
| 11 | Send form for identify speech | Alert information despite success or error in the process | As expected |
| 12 | Direct back to Home page from Identify page | Directed back to Home Page | As expected |
| 13 | Load speech data | Datalist element store the speech data or alert if no speech data is found | As expected |
| 14 | Send form for generate speech | Audio element is shown with the generated speech or alert if error is occurred | As expected |
| 15 | Direct back to Home page from Generate page | Directed back to Home Page | As expected |

Figures 6.3 Some screenshot from the Mimic application scenarios.

The dataset during the speech recognition testing on identify process is 1000 male speech data on no noisy background. Corrected result shows from the more than 75% of model accuracy. The following tests table results below and figures that shows some screenshot of identify process results:

Table 6.4 Tester 1 speech recognition on identify process results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Phoneme** | **Noisy Background** | **Semi Noisy Background** | **No Noisy Background** | **Result** |
| 1 | a |  |  |  | /3 |
| 2 | i |  |  |  | /3 |
| 3 | t |  |  |  | /3 |
| 3 | na |  |  |  | /3 |
| 4 | ma |  |  |  | /3 |
| 5 | mu |  |  |  | /3 |
| 6 | di |  |  |  | /3 |
| 7 | ri |  |  |  | /3 |
| 8 | ku |  |  |  | /3 |
| 9 | kan |  |  |  | /3 |
| 10 | Unknown 1 |  |  |  | /3 |
| 11 | Unknown 2 |  |  |  | /3 |
| 12 | Unknown 3 |  |  |  | /3 |

Figures 6.4 Some screenshot from the Tester 1 results.

Table 6.5 Tester 2 speech recognition on identify process results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Phoneme** | **Noisy Background** | **Semi Noisy Background** | **No Noisy Background** | **Result** |
| 1 | a |  |  |  | /3 |
| 2 | i |  |  |  | /3 |
| 3 | t |  |  |  | /3 |
| 3 | na |  |  |  | /3 |
| 4 | ma |  |  |  | /3 |
| 5 | mu |  |  |  | /3 |
| 6 | di |  |  |  | /3 |
| 7 | ri |  |  |  | /3 |
| 8 | ku |  |  |  | /3 |
| 9 | kan |  |  |  | /3 |
| 10 | Unknown 1 |  |  |  | /3 |
| 11 | Unknown 2 |  |  |  | /3 |
| 12 | Unknown 3 |  |  |  | /3 |

Figures 6.5 Some screenshot from the Tester 2 results.

Table 6.6 Tester 3 speech recognition on identify process results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Phoneme** | **Noisy Background** | **Semi Noisy Background** | **No Noisy Background** | **Result** |
| 1 | a |  |  |  | /3 |
| 2 | i |  |  |  | /3 |
| 3 | t |  |  |  | /3 |
| 3 | na |  |  |  | /3 |
| 4 | ma |  |  |  | /3 |
| 5 | mu |  |  |  | /3 |
| 6 | di |  |  |  | /3 |
| 7 | ri |  |  |  | /3 |
| 8 | ku |  |  |  | /3 |
| 9 | kan |  |  |  | /3 |
| 10 | Unknown 1 |  |  |  | /3 |
| 11 | Unknown 2 |  |  |  | /3 |
| 12 | Unknown 3 |  |  |  | /3 |

Figures 6.6 Some screenshot from the Tester 3 results.

# CONCLUSIONS AND FUTURE WORK

## Conclusion

The following list sums up that the application is achieved based on this research objective:

1. This application enables to recognize speech in Bahasa Indonesia speech from record audio.
2. This application enables to generate speech in Bahasa Indonesia speech from text.
3. This application enables to mimic speech through the website.
4. This application enables to collect speech data through the website.
5. This application enables to train model with collected speech data through the console.

## Future Work

The following suggestion for further development and improvements of the research or application:

1. User Interface

Improvement on UI will always help the user experience. With some colourful theme, clear button or inputs, the application will comfortable to be used.

1. Speech Recognition

Improvement on machine learning model can be made. When there is no right or wrong in modelling the machine learning model, there is always optimal model to get the best accuracy. A research to find the optimal model or a research developing application with different machine learning model are big improvement in the application or even in Speech Recognition itself.

1. Speech Synthesis

Improvement on Speech Synthesis is when generating the speech. By removing some of silence or unused part of the speech and also reducing background will make generated speech more fluently and good to hear.

REFERENCES

1. Hurwitz, J., & Kirsch, D. (2018). *Machine Learning For Dummies®, IBM Limited Edition.* New Jersey: John Wiley & Sons, Inc.
2. Nilsson, N. J., & Laboratory, R. (2005). *INTRODUCTION TO MACHINE LEARNING.* California: Nils J. Nilsson.
3. dictionary.cambridge.org. (2018). *MACHINE*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/machine
4. dictionary.cambridge.org. (2018). *LEARNING*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/learning
5. Society, T. R. (2017). *Machine learning: the power and promise.* London: The Royal Society.
6. edureka.com. (2018, October 18). *What is Machine Learning?* Retrieved from edureka: https://www.edureka.co/blog/what-is-machine-learning/
7. Geitgey, A. (2016, January 3). *Machine Learning is Fun! Part 2*. Retrieved from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3
8. Magdi Zakaria, M. A.-S. (2014). Artificial Neural Network : A Brief Overview. *Int. Journal of Engineering Research and Applications*, 6.
9. A.D.Dongare, R. A. (2012). Introduction to Artificial Neural Network. *International Journal of Engineering and Innovative Technology*, 6.
10. courses.lumenlearning.com. (n.d.). *Introduction to Language*. Retrieved from lumen: https://courses.lumenlearning.com/boundless-psychology/chapter/introduction-to-language/
11. readingdoctor.com.au. (2016). *Phonemes, Graphemes and Letters: The Word Burger*. Retrieved from Reading Doctor: http://www.readingdoctor.com.au/phonemes-graphemes-letters-word-burger/
12. Yanti, N. T. (n.d.). FONEM BAHASA INDONESIA. *Academia.edu*, 10.
13. dictionary.cambridge.org. (2018). *VOWEL*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/vowel
14. puebi.readthedocs.io. (n.d.). *Huruf Vokal*. Retrieved from PUEBI Daring: https://puebi.readthedocs.io/en/latest/huruf/huruf-vokal/
15. dictionary.cambridge.org. (2018). *DIPHTHONG*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/diphthong
16. puebi.readthedocs.io. (n.d.). *Huruf Diftong*. Retrieved from PUEBI Daring: https://puebi.readthedocs.io/en/latest/huruf/huruf-diftong/
17. dictionary.cambridge.org. (2018). *CONSONANT*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/consonant
18. puebi.readthedocs.io. (n.d.). *Huruf Konsonan*. Retrieved from PUEBI Daring: https://puebi.readthedocs.io/en/latest/huruf/huruf-konsonan/
19. dictionary.cambridge.org. (2018). *CLUSTER*. Retrieved from Cambridge English Dictionary: https://dictionary.cambridge.org/dictionary/english/cluster
20. puebi.readthedocs.io. (n.d.). *Gabungan Huruf Konsonan*. Retrieved from PUEBI Daring: https://puebi.readthedocs.io/en/latest/huruf/gabungan-huruf-konsonan/
21. Dave, B., & Pipalia, P. D. (2014). SPEECH RECOGNITION: A REVIEW. *International Journal of Advance Engineering and Research*, 7.
22. M.A.Anusuya, & S.K.Katti. (2009). Speech Recognition by Machine: A Review. *International Journal of Computer Science and Information Security*, 25.
23. Geitgey, A. (2016, December 24). *Machine Learning is Fun Part 6*. Retrieved from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-part-6-how-to-do-speech-recognition-with-deep-learning-28293c162f7a
24. en.wikipedia.org. (2018, October). *Nyquist–Shannon sampling theorem*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon\_sampling\_theorem
25. Olshausen, B. A. (2000). Aliasing. *Redwood Center for Theoretical Neuroscience*, 6.
26. Rabiner, L. R., & Schafer, R. W. (1978). *Digital Processing of Speech Signals.* Prentice Hall: New Jersey.
27. Hande, S. S. (2014). A Review on Speech Synthesis an Artificial Voice Production. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 8.
28. lyrebird.ai. (2018). *Our Voice Products*. Retrieved from Lyrebird: https://lyrebird.ai/products
29. translate.google.com. (2018). *Languages*. Retrieved from Google Translate: https://translate.google.com/intl/en/about/languages/
30. w3school.com. (2018). *Node.js Introduction*. Retrieved from W3School: https://www.w3schools.com/nodejs/nodejs\_intro.asp
31. mongodb.com. (2018). *NoSQL Databases Explained*. Retrieved from MongoDb: https://www.mongodb.com/nosql-explained