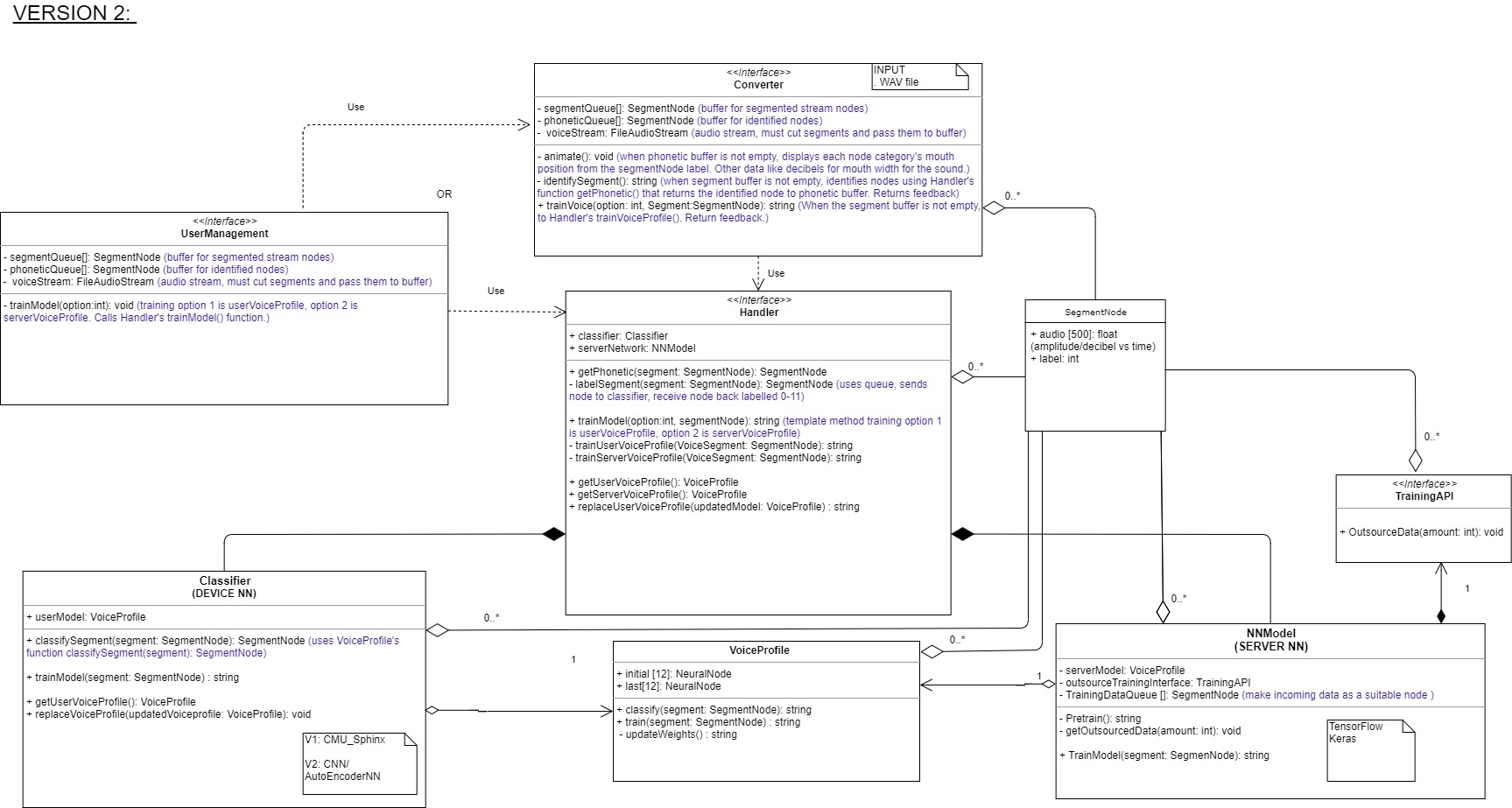
**Omega -** Neural Network

# Integration Interface Document

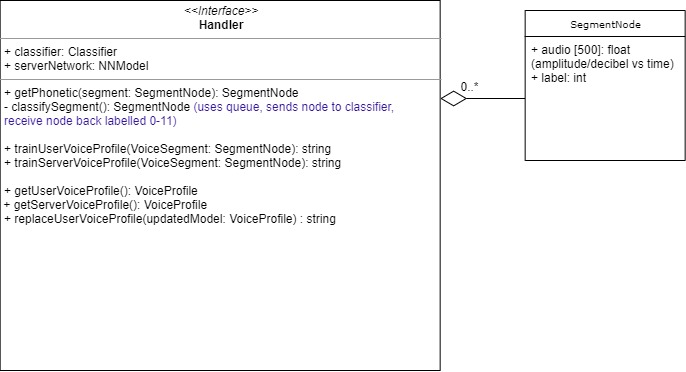
[last updated: 03 April 2020]

NN Module UML Diagram

****

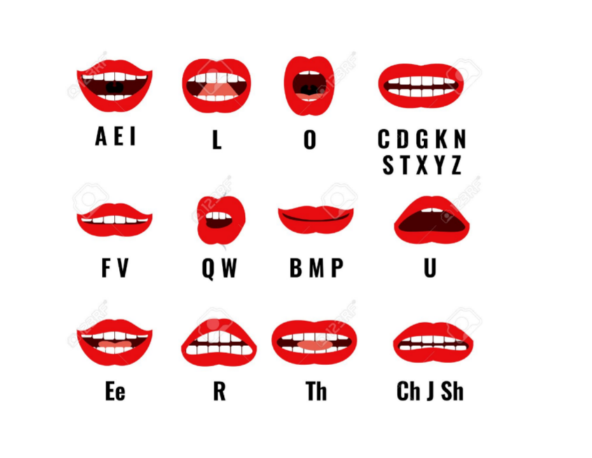
This is the **Handler Class,** it is written in Java. It is responsible for providing an interface from the NN module to the Converter and User Management Modules.

The **SegmentNode Class**contains an array with *audio* data of *20 milliseconds*. This is a graph of *amplitude vs time*. Time is represented by the array’s index and the value will be represent the amplitude. The *label* is an integer from 0-11, indicating which mouth position category the piece of audio falls under. This class is a data structure that is moved between subsystems.



## Speech Recognition – (converter team)

To determine a phonetic for a mouth position from audio, a *segment* of that audio is sent to **Handler** as a parameter in the *getPhonetic()*function. The **Handler** will return the piece of audio with an integer *label* for what phonetic the audio belongs to. These phonetics labels will fall into 12 categories, from integer 0 to 11 in the figure below:



This piece of audio and label is part of an object from the **SegmentNode Class**. The **Converter Module** will create this object then pass it to the **Handler Class**’ function like, *getPhonetic(segment: SegmentNode): SegmentNode*. The function returns the object with an updated *label* for the converter team to use to determine what to change the mouth position to.

## 

## Training and Testing – (user management)

NB: SUBJECT TO CHANGE UPON PROJECT PROGRESSION

1. **NNmodel Class** (SERVER NN) is a server that holds the neural network. This class is responsible for speech recognition and requires training to learn how to identify sounds. This model is pre-trained by outsourcing training data via the **TrainingAPI Class**.
2. **Classifier Class** (DEVICE NN) holds the default model (server neural network) as the user **VoiceProfile Class**. This class is responsible for training and filtering the model so that the output is more accurate to the specific user upon calibrating the voice profile.

To send testing data to these models, use a function in the **Handler Class** called *trainModel(option:int, segment: SegmentNode): string*. To train the server model, first parameter *option* must be set to 0. To train the device model, it must be set to 1. The **SegmentNode** objectcontains the audio’s amplitude vs time graph and has the label populated to what phonetics the audio is - movement of this node is like communication between the NN module and the converter module explained on the previous page. The function returns a string entailing any errors, information or warnings about the process. It is a by-product and is not necessarily needed. 