**Voice Authentication and Bike Command Classification using Machine Learning**

This voice authentication and bike command classification works by at first registering users using 27 sec voice clip during registration process. It also collect 2 sec audio clip of “Hey TVS” as a wake word from the user during registration phase. During login phase it collects 10 sec audio clip from the user. It then finds the similarity score of the login user with all the registered users one by one. The same person who had previously registered and is now trying to login will have the maximum similarity score and that score should also be more than 0.7. For registration purpose we used speech brain a python package and an open-source Python toolkit designed for developing, training, and deploying speech and audio processing systems. It’s built on top of PyTorch and is used a lot in research and real-world applications.

**Data Collection:**

**[Part 1 – User\_Voice\_Authentication\_DataSet]**

Registration - For user registration 27 sec audio clip is recorded and stored in Registration Folder for the person who want to do registration.

Login – 10 second audio clip of a person who is a registered user and wants to login is stored in the folder named Login.

Since we are using speech brain which is a pretrained model we do not need to train it externally in order to compute similarity score of the speech. For the demonstration purposes we are providing 3 users sample here each for Registration, Login, HeyTVS wake word and Command folder here.

**[Part 2 – NLP\_Bike\_Command\_DataSet]**

This folder contains a csv file. This csv file contains 1495 bike commands and 1495 non-bike commands. Each bike commands has a label 1 and non bike commands has a label 0. This dataset has been built from scratch. So a total of 2990 data points are there.

Each bike commands can be further be subclassified as EDGE, CLOUD and UPDATE.

So once a user is registered and then he/she does login, then he/she can use its own voice commands to control the bike. It will lead to enhanced user experience.

For this purpose we have built 4 LSTM models.

1. 1st model is the MAIN model which classifies a command as a bike or non bike command and into its subclasses as EDGE, CLOUD or UPDATE.
2. If it is a bike command and it will fall in one of these subclasses it is further classified into its respective subclass category.

So according to the commands appropriate response can be generated and bike can be controlled accordingly.

Example – “Show me the shortest route to goa”

Classification – It is a Bike Command and it is of subclass type CLOUD and subclass category Traffic Maps.

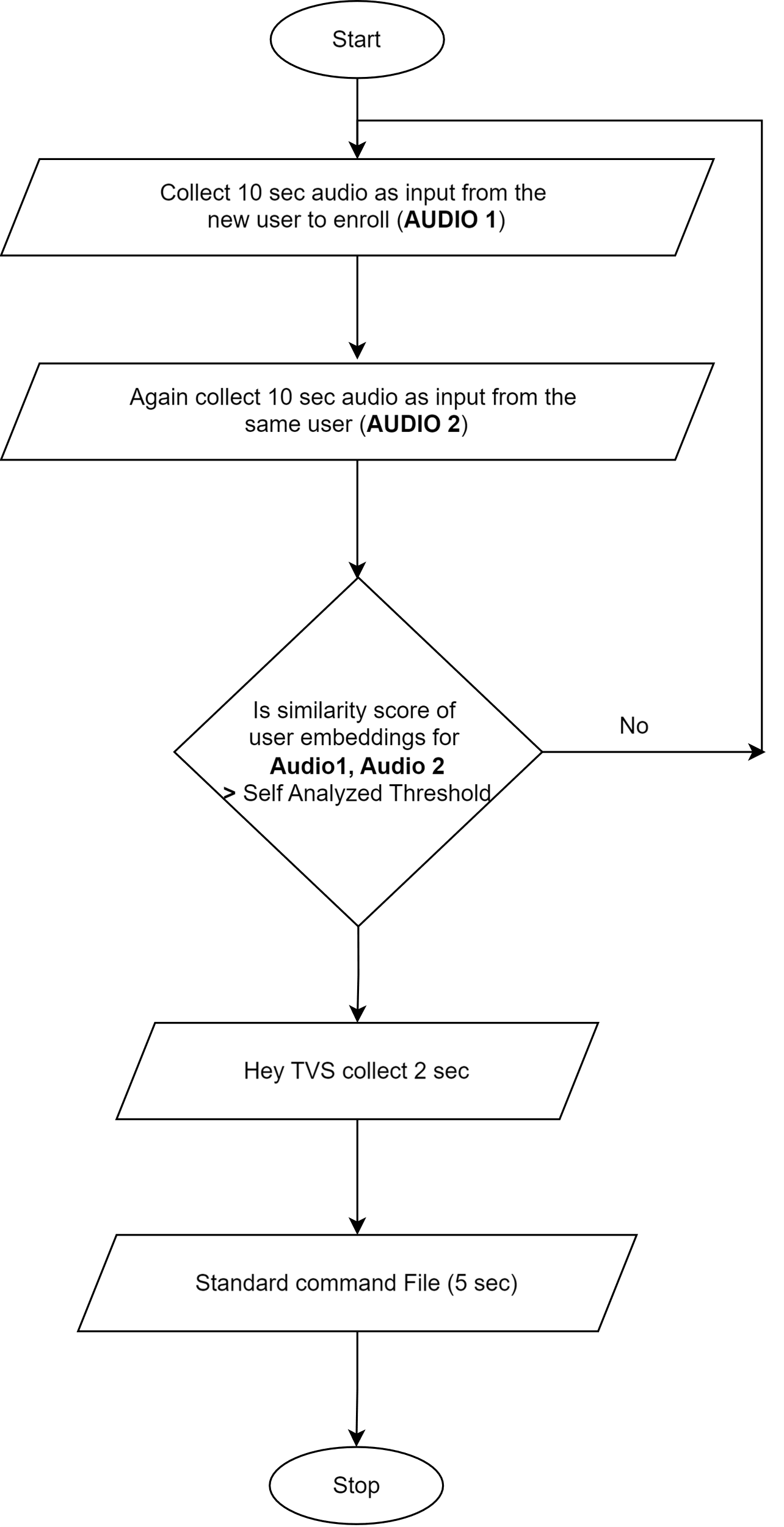
**Dataset Information [Bike vs Non Bike Command DataSet]**

|  |  |  |
| --- | --- | --- |
| Command | No. of data-points  [Subclass] | No. of Data-points  [SubclassCategory] |
| Bike | Edge: 551 | Basic : 287  Battery Fuel : 144  Tyres :120 |
|  | Cloud: 450 | Traffic/Maps : 145  Songs/Media : 102  News/Notifications : 102  Weather : 101 |
|  | Update: 319 | Check : 120  Perform : 103  Cancel : 100 |
| Non-bike | Non-Bike: 1495 | N.A |
| **Total** | **2990** | **2990** |

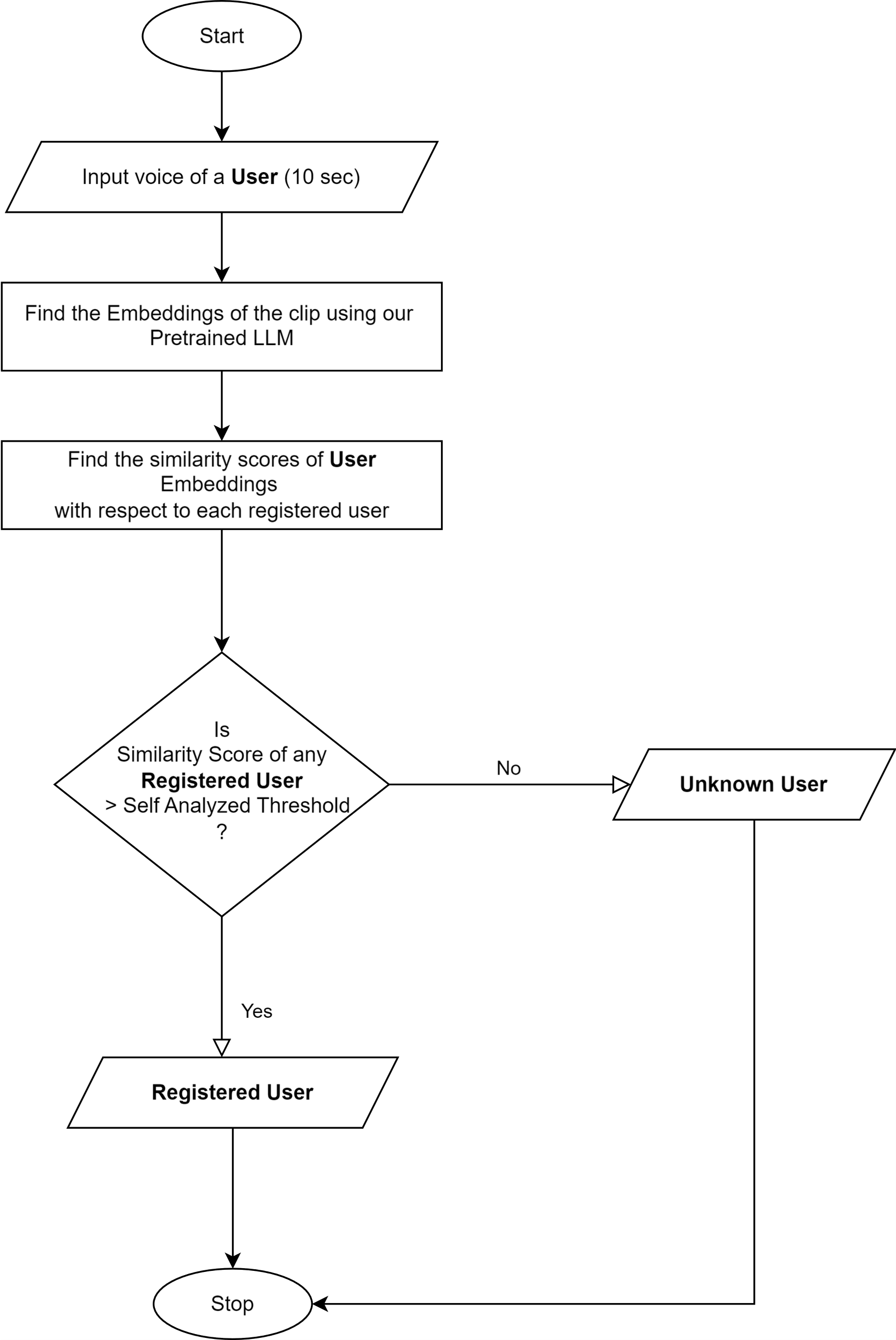
Deployment – All LSTM models are converted to tflite with tf.default optimization which does quantization of models to int8 and we get an approx. of 10x compression reduction.

So total size of the original models was 40 MBs and after model compression and quantization its size is reduced to approximately 4.1 MB i.e approximately 10x reduction in size.

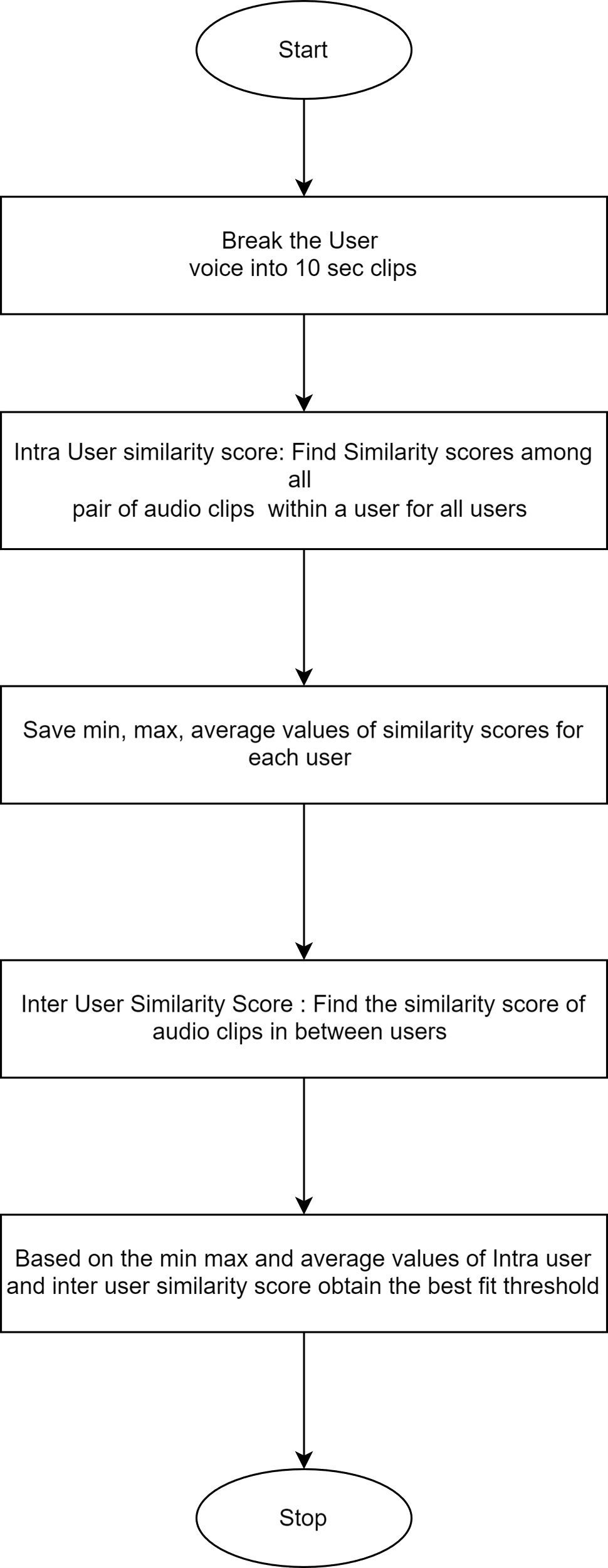
1. Speaker Registration Flowchart



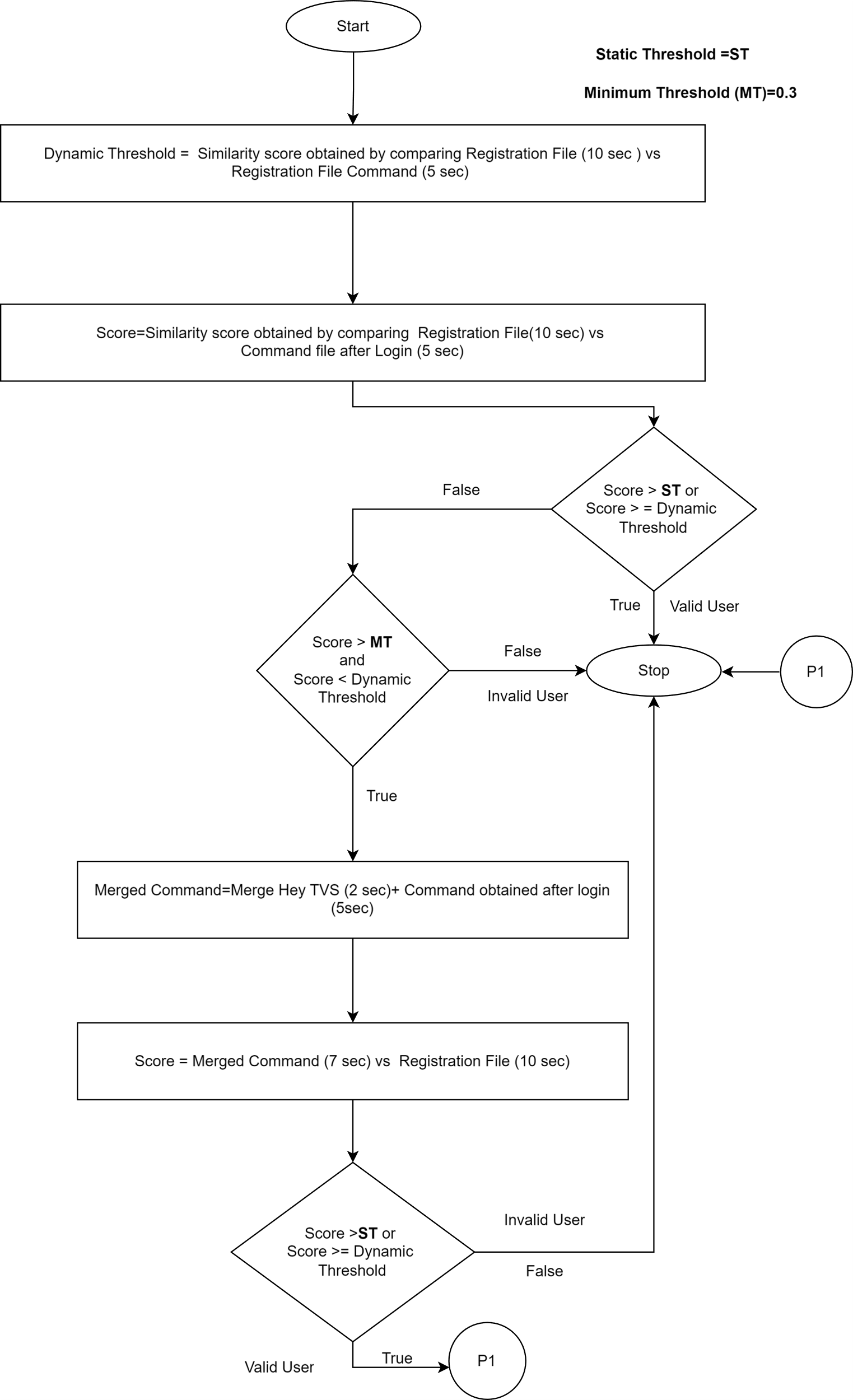
2. Speaker Login Flowchart



1. Threshold Optimization Flowchart



1. Speaker Validation Flowchart



1. NLP MAIN MODEL FLOWCHART

A diagram of a flowchart

AI-generated content may be incorrect.

**Deployed Code**

The deployed code responsible for handling user registration, login, and NLP-based voice command classification is contained in the file **RegandloginNLP.py**, located at **./Deployed\_Code/Server/**. This script runs directly on the edge device — the Raspberry Pi 5 — and is optimized for low-latency, resource-constrained environments. It includes logic for verifying registered users via voice authentication and classifying input commands as either bike-related or irrelevant (non-bike). The code ensures real-time response, noise filtering, and user validation using quantized NLP models.

**Training Code**

The model training pipeline for voice command classification is implemented in the Jupyter Notebook file **RNN\_text\_classification\_Multiclass\_LSTM\_Final.ipynb**, located in the **Model\_Training\_code/** directory. This notebook contains the entire workflow for preparing the dataset, training an LSTM-based RNN model to differentiate between bike and non-bike commands, and exporting the trained model for deployment. The training process includes data preprocessing, sequence padding, multi-class label encoding, and evaluation using accuracy metrics. The final model is later quantized and transferred to the edge device for real-time inference.