System Design Document

For

Speech Recognition for Air Traffic Control

Team members: Braeden Burnett, Jakob Haehre, Kira McFadden, Tyler Carr

Version/Author	Date
1/Tyler Kira	10/5/22
1.2/Braeden, Jakob	10/7/22
2.1/Braeden, Kira	11/7/22
2.2/ Braeden, Jakob, Tyler	11/8/22
3.1/Braeden	12/3/22
3.2/Braeden	3/10/23
4/Tyler, Jakob, Braeden	4/27/23

TABLE OF CONTENTS

SYSTEM DESIGN DOCUMENT

Overview

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of the system is to show live transcriptions of ATC communications overlayed on an interactive map so the user does not have to seek out the ATC data themselves and listen to it but instead has a live transcription of live ATC data on the interactive map.

1.2 Project Executive Summary

When browsing an interactive map of flights, a user would have to research the flight information in order to find the live ATC communication data. When they find it, they won't have the transcribed text of what they are saying or the history of what has already been said. This system solves these problems by displaying an interactive map that the user can navigate around in and overlays live aircraft data that updates location in real time. The live ATC communications from the aircraft are transcribed and displayed on the screen in addition to flight information for any aircraft that is selected.

1.2.1 System Overview

The system consists of two main parts.

Web-Based GUI ASR App

This app will display both an interactive map and a VFR Aeronautical map with the ability to switch back and forth in-between them. Both versions of the map will include buttons to zoom in and zoom out. Both maps must also give the user the ability to click and drag to pan around the map. Overlayed on top of both of the maps, icons representing airplanes will be displayed for every aircraft that is able to be tracked. The icons are updated to move the position of where they have traveled to in real time. After clicking on an aircraft, a live transcription of the aircraft's communication with the ATC is overlaid on the screen in a moveable box. In addition to this, basic flight tracking information is also included, if available.

Audio Speech Recognition

A database of LiveATC data will be used to train a Nvidia NeMo model to recognize ATC and aircraft communication and transcribe it into text. The training is done with a dataset of audio that has already been transcribed in order to teach the model what good transcriptions look like. After the model is trained, it will be used to transcribe live audio data into text. This is the text that will be overlayed on the website in real time. This is necessary because ATC uses jargon that is not in normal language, so it needs to be specifically trained on this speech.

High-level system architecture diagram

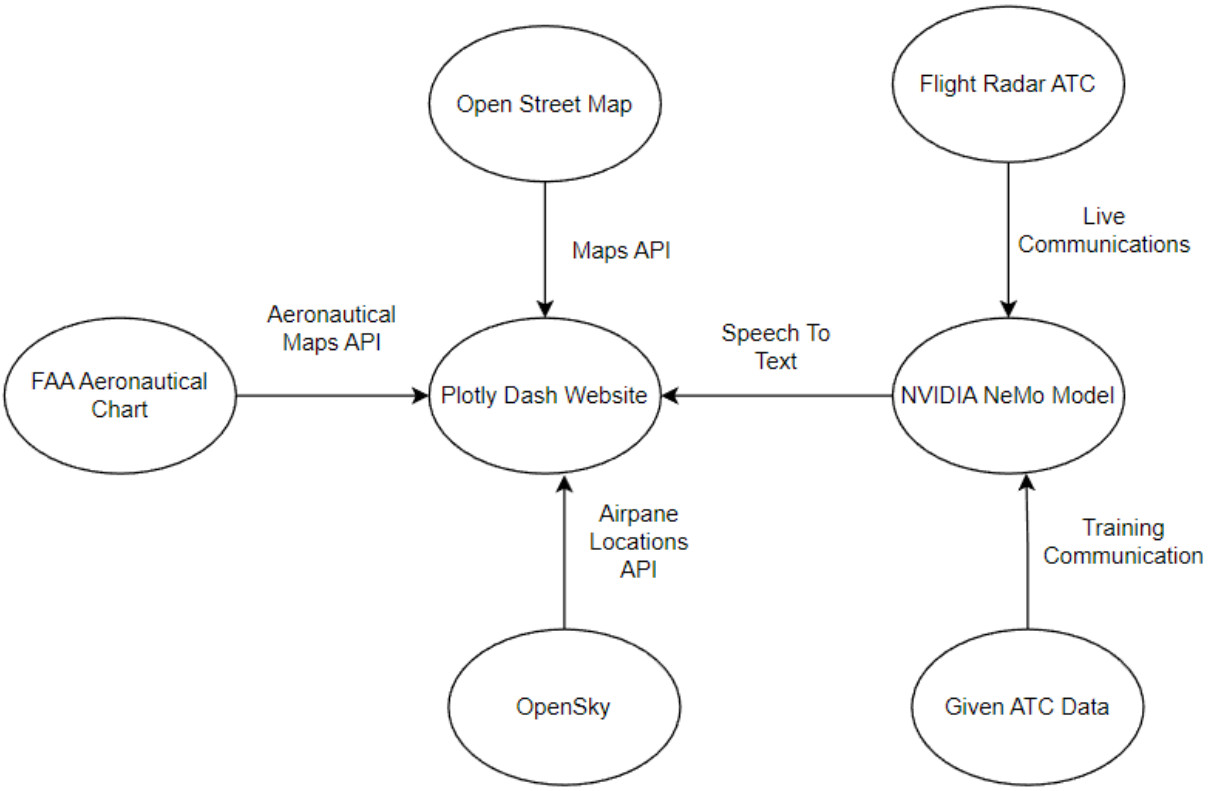

Fig 1. The above figure shows how the given and Live ATC data is passed NeMo model creating the speech to text for the Plotly Dash website and the API's which integrate with the website.

1.2.2 Design Constraints

The development team is restricted to using Plotly Dash to run the website's server. There aren't many resources available for sectional aeronautical charts besides what is provided by the FAA's website, so this restricted the capability of the interactivity of the sectional map within the website to what could be provided by a PDF.

1.2.3 Future Contingencies

A future contingency that could arise throughout this project include our plane coordinate api not working, the plan to account for this is to display the last known location of the plane until the api comes back. Another contingency is that the model outputs a poor transcription, and the plan to account for this is to deploy an additional website for closed crowd-sourced model transcription validation. Another future contingency that could arise is that the website is offline, and the plan to account for this is to display an error message in place of the website displaying a message that the website is currently unavailable. A final contingency that could arise throughout the project includes the model being unavailable and the plan to account for this is to temporarily pause the transcriptions and write transcriptions temporarily unavailable.

1.3 Document Organization

The System Design Document begins with an introductory section that provides an in-depth overview of the project. It then moves on to defining terms and references for external resources that were commonly used on the project. After that, the document talks about the system architecture, the human-machine interface, detailed design, external interfaces, and system integrity controls.

1.4 Project References

1. System Requirements Specification Document

2. System Test Plan Document

3. Nvidia NeMo: https://developer.nvidia.com/nvidia-nemo - The NVIDIA NeMo website detailing how to use Nvidia Neural Modeling

4. Plotly Dash: https://dash.plotly.com/layout - The Plotly Dash Website detailing how to use Plotly Dash to build a dynamic website

5. Leaflet (leaflet.js interactive maps): https://github.com/thedirtyfew/dash-leaflet - The Dash-Leaflet GitHub detailing how to use an interactive map in a Plotly Dash website

6. https://www.earthdatascience.org/tutorials/introduction-to-leaflet-animated-maps/- The tutorial for using leaflet and folium to create dynamic maps using an API in Plotly Dash

7. https://medium.com/@shachiakyaagba_41915/integrating-folium-with-dash-5338604e7c56

   -The tutorial for using leaflet for automated dynamic maps in Plotly Dash

8. Aeronautical VFR charts:

https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/vfr The Federal Aviation Administration website containing the Aeronautical charts

1.5 Glossary

ASR (Automated Speech Recognition) - Allows users to input information via speech rather than inputting information using a keyboard.

ATC (Air Traffic Control) – Traffic controlling facility used in the United States for the purpose of detecting air traffic.

API (Application Programming Interface) -Software intermediary which allows multiple applications to communicate.

GUI (Graphical User Interface)- Multimedia interface user interacts with to use a program.
NeMo (Neural Modeling)- A NVIDIA toolkit for building AI models with ASR, NLP, and TTS models.

NLP (Natural Language Processing)- enables machines to understand and respond to text or voice data.
TTS (Text To Speech)- a type of assistive technology that reads digital text aloud.

VFR (Visual Flight Rules)- A set of regulations that an aircraft can use to operate under clear, sunny weather conditions.

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

To train the neural models for the project we are using two desktop machines and a GPU Laptop all equipped with Nvidia Rtx 3090 GPUs. These GPUs are essential as they enable us to train the Nvidia Nemo ASR models. The Laptop will act as the host for the web API. There is no interaction between hardware in our system as all resources needed are either pulled from the web or stored locally on the machine. The system will be deployed on a single machine that runs in parallel the website and the neural model. This system will need to be connected to the internet allowing it to host the website.

2.2 System Software Architecture

The system software is made up of two separate subsystems both using python. First system being the speech recognition system made using Nvidia Nemo ASR models and second system being a web API using Plotly Dash.

Nvidia Nemo is a set of neural models specifically designed for Automatic Speech Recognition written in python. It utilizes several other python machine learning modules and wraps them up into a single toolkit that uses hardware acceleration to train the models. This hardware acceleration enables improved training times. Meaning we can more effectively test all of the models included in the toolkit to figure out which one is best able to meet our needs. The software architecture for the nemo system can be seen in Fig 2.

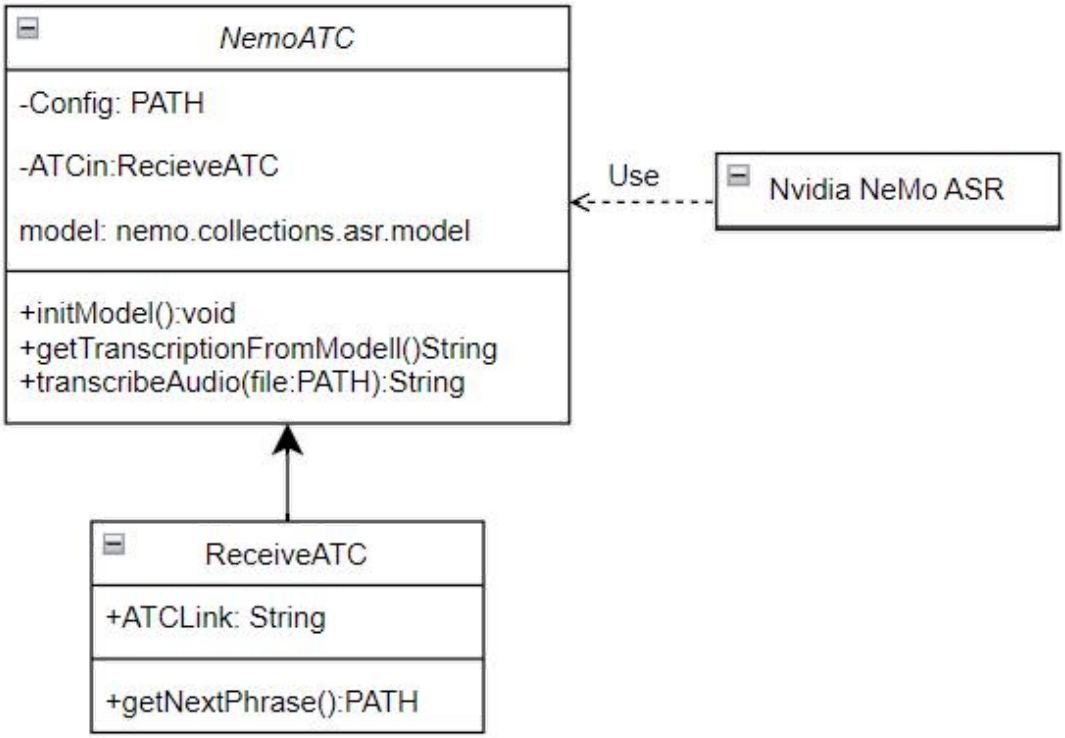

Fig 2. The above figure shows how the Neural Model ATC class uses the Nvidia Nemo ASR model and the Receive ATC from FlightRadar24.net to get the Audio and transcribe the audio to speech

Plotly Dash is a python based web design API that enables us to create dynamic websites. It being python based enables us to more effectively integrate the Nvidia Nemo models into the website and ATC data from relevant source as can be seen in fig 3 below.

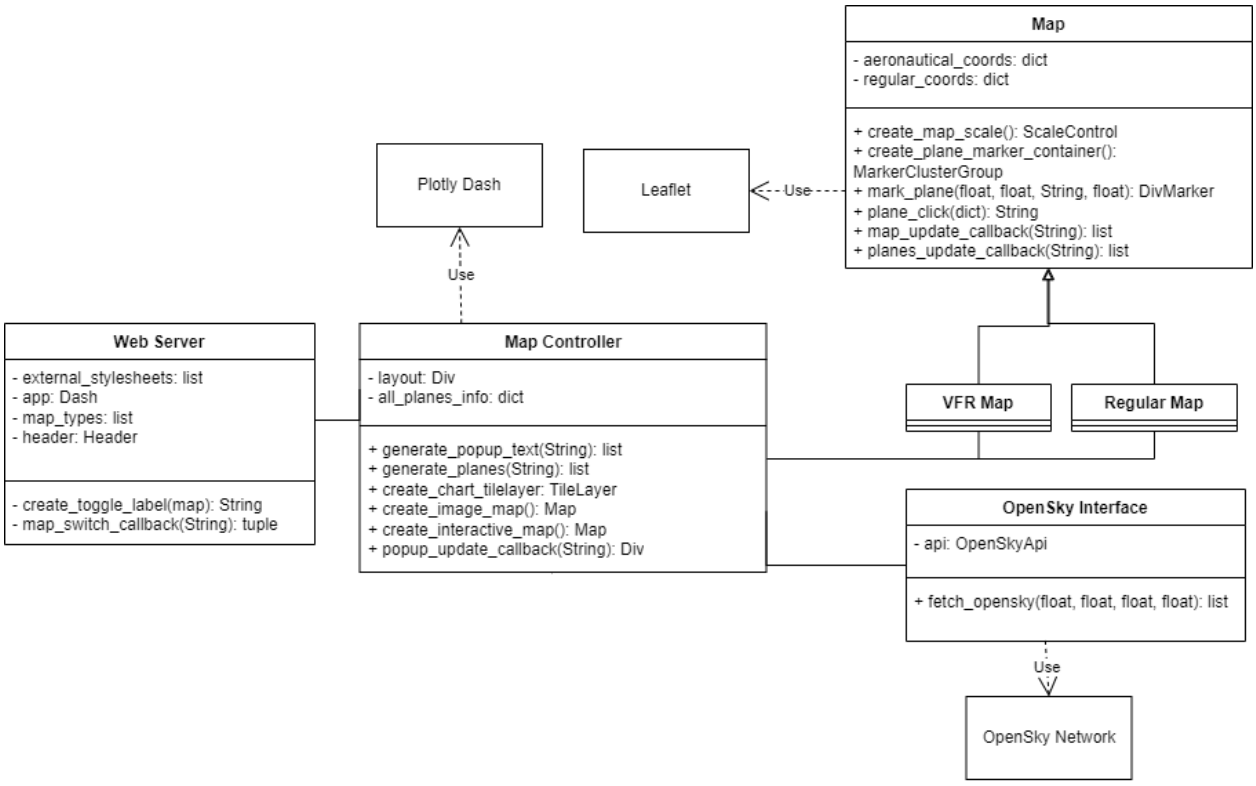

Fig 3. The above figure shows how the website is integrated with the different Map API's and how these API's are used for the Aeronautical and Street Map views

2.3 Internal Communications Architecture

Communication within the system deals with Live ATC data drawn from an online source giving the Live ATC transmissions as .wav files and is fed into the neural model(s) which have been trained with previous ATC communications data and will output the live text using the previous training data along with the current stream of communication to determine what the output of the audio is from the Live ATC data and transcribes this audio to text. The transcription of the

speech as text will then be passed to the website using an API which takes the transcription of the ATC speech and displays this to the website text box for the user. The communication between the website and the street map and aeronautical chart is by the passing of the maps onto the website using an API and displaying the desired map choice based upon the user clicking the toggle switch on the website giving either the street map display or the aeronautical chart display. Finally, the communication between the website and the plane locations will be based on the OpenSky API which interacts with the website through an API which plots the locations of the airplanes on both the street map and aeronautical chart on the website given the user choice between the two maps displays and is updated every 15 seconds.

3 HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs.

3.1 Inputs

The main form of operator input into the system is the live ATC feed and datasets. The datasets are given in .json object files which point to audio files which are in .wav format and hold the given audio transcriptions for the models to train on. They are a dataset specifically designed to train models on ATC speech. Using these files, we will train the neural models to transcribe live ATC data. The live ATC data will be drawn from online resources.

The main form of user input into the system will be through an interactive map the user can use and a toggle to change the map type. The map toggle will change the view of the map between a regular civilian map which is drawn from the Map box API and an aeronautical map drawn from the FAA website at https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/vfr/. Users will also be able to click on a plane icon to display additional information regarding the craft, as well as the ATC transcription.

3.2 Outputs

The output of the system is a website containing an updated street map or aeronautical map based on a toggle and displaying aircraft locations and headings over each of the map, and a moving text box which outputs the ATC transcriptions from the neural models. When an aircraft is clicked on the website will display additional information regarding that aircraft.

4 DETAILED DESIGN

4.1 Hardware Detailed Design

There was no hardware designed by the group for this project as this project required only the use of software based designs. All hardware components were provided through the laptop and computer which contained an NVIDIA Geforce series graphics card used to train the Nemo models and transcribe the speech on the Plotly Dash website.

4.2 Software Detailed Design

The software required for the Website includes Plotly Dash, OpenStreetMap, OpenSky Network, and FAA Aeronautical maps, along with the NeMo links.

The software required for the NeMo ASR includes Python3.8 and Pytorch 1.10. The software will stream ATC audio to the nemo model(s) and then transcribe them. These transcriptions will then be streamed to the website and displayed to the user.

4.3 Internal Communications Detailed Design

The internal communications detailed design includes the Plotly Dash created website interfacing with the OpenStreetMap API to update the map, interfacing with updating flight data for website airplanes based on live flight data, live ATC communications based on FAA radio communications given by FlightRadar24, and a transcription of these communications by the NVIDIA NeMo modeling. All of these are given to the website to be displayed to the user. The user will be able to request specific information. This information is stored by the website after it is received from the previously discussed sources.

5 EXTERNAL INTERFACES

5.1 Interface Architecture

The main interfaces of the system will be with file systems on the machines and online resources. Both Nvidia Nemo 1.8.2 and Plotly Dash 2.6.1 are python modules and therefore run using regular python interfaces. Online resources will also be accessed by using python modules through the API on the Plotly Dash website as seen in figure 4 and this will happen on the machine the website is currently running on. However, these online resources require scripts to change the data from its original format to a format demanded by the Nemo models and Plotly Dash which is seen in the speech to text transmission for the website.

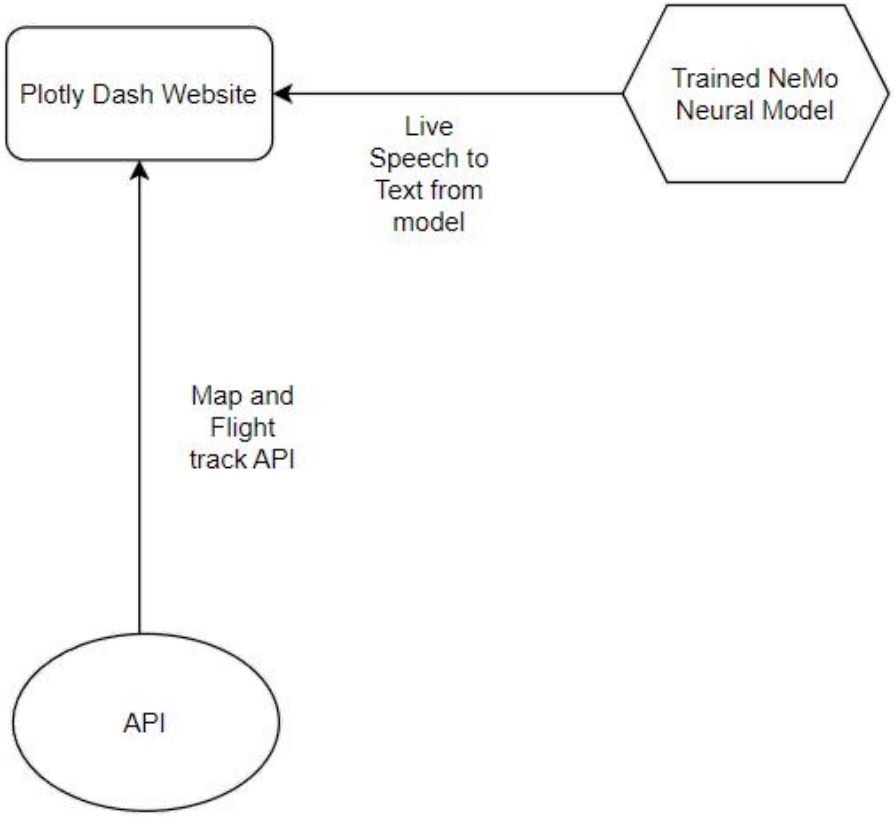

Fig 4. The above figure shows the connection between the trained NeMo model with the Live ATC speech and the map and flight API interaction with the Plotly Dash website.

5.2 Interface Detailed Design

File system interfaces will be handled using standard python file system functions as well as standard .json objects and .wav functions. The ATC training data that is being used to train the networks as seen in Figure 5 is made up of .sph files holding the audio and .txt holding the transcriptions. These files cannot be interpreted by the nemo neural models and thus need to be combined then converted to a .json object file that points to a .wav file. The live ATC will be drawn from the online resource website LiveATC.net and be input into the neural models as a live stream of the data directly from the LiveATC.net website over HTTP protocol into the trained Neural Model through the downloading and interfacing of python modules. The neural models will interact with the website using python scripts that will populate the text box on the website with the transcription from the neural model.

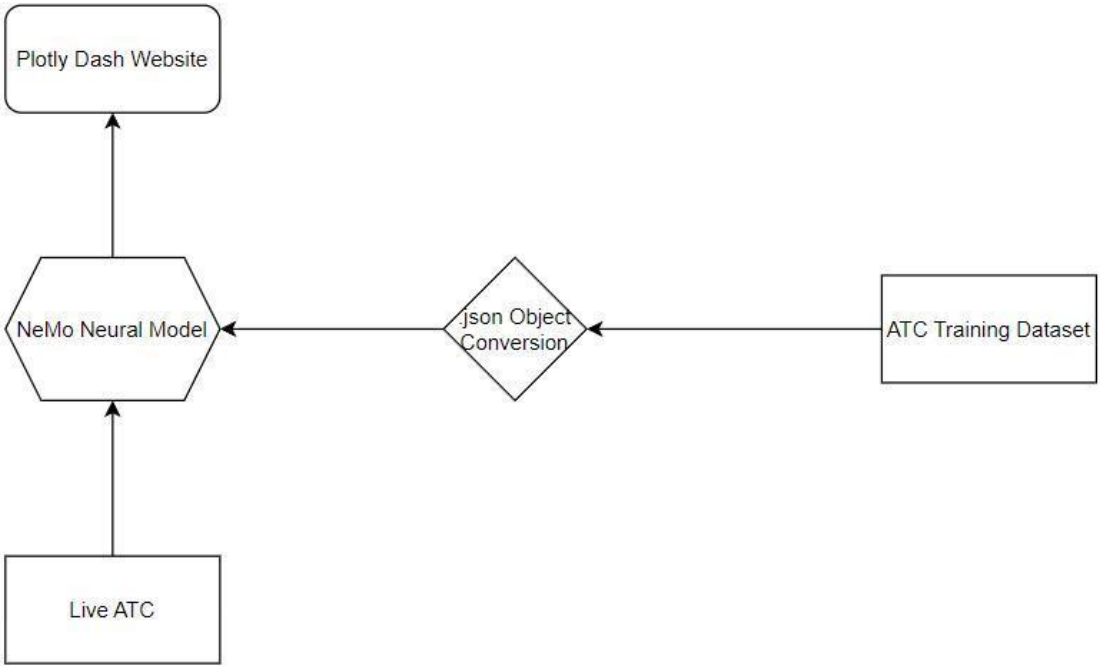

Fig 5. The above figure shows the connection between the training ATC data in the .json object and Live ATC feed used in the Neural Model and changed from speech to text for the website.

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

6 SYSTEM INTEGRITY CONTROLS

To train the Nvidia nemo models we have licensed several datasets. These datasets cannot be shared to public repositories as it would break the terms of the license agreement. To prevent this data from being shared we need to properly manage the Github repository. This means not adding the data files to be tracked by git and using git ignores to avoid it being published to our public repository and keeping these files on a private OneDrive and on the private computers we are utilizing for training to ensure these datasets do not end up in the public domain . These datasets are the only sensitive information we are concerned about in our project.