System Design Document

For

Speech Recognition for Air Traffic Control

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

Version/Author	Date
1/Tyler	10/5/22
1.2/Braeden Jakob	10/7/22

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.

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 overlayed on the screen in a moveable box. In addition to this, basic flight tracking information is also included, if available.

Transcription

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.

Todo:

High-level system architecture diagram

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

This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans.

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

- System Requirements Specification Document (Insert link here)
- Nvidia NeMo: https://developer.nvidia.com/nvidia-nemo
- Plotly Dash: https://dash.plotly.com/layout
- Folium (leaflet.js interactive maps): http://python-visualization.github.io/folium/
    - https://www.earthdatascience.org/tutorials/introduction-to-leaflet-animated-maps/
    - https://medium.com/@shachiakyaagba_41915/integrating-folium-with-dash-5338604e7c56
- Aeronautical VFR charts: https://www.faa.gov/air_traffic/flight_info/aeronav/digital_products/vfr

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 program.
NeMo (Nueral 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

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 6, External Interfaces.

2.1 System Hardware Architecture

In this section, describe the overall system hardware and organization. Include a list of hardware components (with a brief description of each item) and diagrams showing the connectivity between the components. If appropriate, use subsections to address each subsystem.

This system utilizes two desktop machines both equipped with Nvidia Rtx 3090 GPUs. These GPUs are essential as they enable us to train the Nvidia Nemo ASR models. One of these machines will also 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.

2.2 System Software Architecture

In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

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.

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.

2.3 Internal Communications Architecture

In this section, describe the overall communications within the system; for example, LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25, Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed.

Note: The diagrams should map to the FRD context diagrams.

Communication within the system is rather limited. Live ATC data will be drawn from an online source as .wav files and will then be fed into the neural model(s). The neural models will transcribe this data then feed it to our web API that will display this information to the user in the form of a map.

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

This section is a description of the input media used by the operator for providing information to the system; show a mapping to the high-level data flows described in Section 1 .2.1, System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in Section 3, File and Database Design, may be referenced. Include data element definitions, or refer to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

- Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
- Description of any access restrictions or security considerations
- Each transaction name, code, and definition, if the system is a transaction-based processing system

The main form of operator input into the system is the live ATC feed and datasets. The datasets are .json files that point audio files in .wav format and hold audio transcriptions. They are 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 a map the user can scroll through 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. Users will also be able to click on an airplane and additional information will appear regarding the craft.

3.2 Outputs

This section describes of the system output design relative to the user/operator; show a mapping to the high-level data flows described in Section 1.2.1. System outputs include reports, data display screens and GUIs, query results, etc. The output files are described in Section 3 and may be referenced in this section. The following should be provided, if appropriate:

- Identification of codes and names for reports and data display screens
- Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
- Description of the purpose of the output, including identification of the primary users
- Report distribution requirements, if any (include frequency for periodic reports)
- Description of any access restrictions or security considerations

The output of the system is a map displaying aircraft locations and headings, and a text box outputting 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

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

4.1 Hardware Detailed Design

The hardware requirements for the NeMo ASR system are a computer with an NVIDIA Geoforce series graphics card.

4.2 Software Detailed Design

The software required for the Website includes Plotly Dash, Google Maps API, and FAA Aeronautical maps, along with google map API links, flightradar24 links and NeMo links.

The software required for the NeMo ASR includes Python3.8 and Pytorch 1.10.

4.3 Internal Communications Detailed Design

The internal communications detailed design includes the Plotly Dash created website interfacing with the Google Maps 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.

5 EXTERNAL INTERFACES

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

5.1 Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

The main interfaces of the system will be with file systems on the machines and online resources. Both Nvidia Nemo and Plotly dash are python modules and therefore run using regular python interfaces. Online resources will also be accessed by using python modules 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.

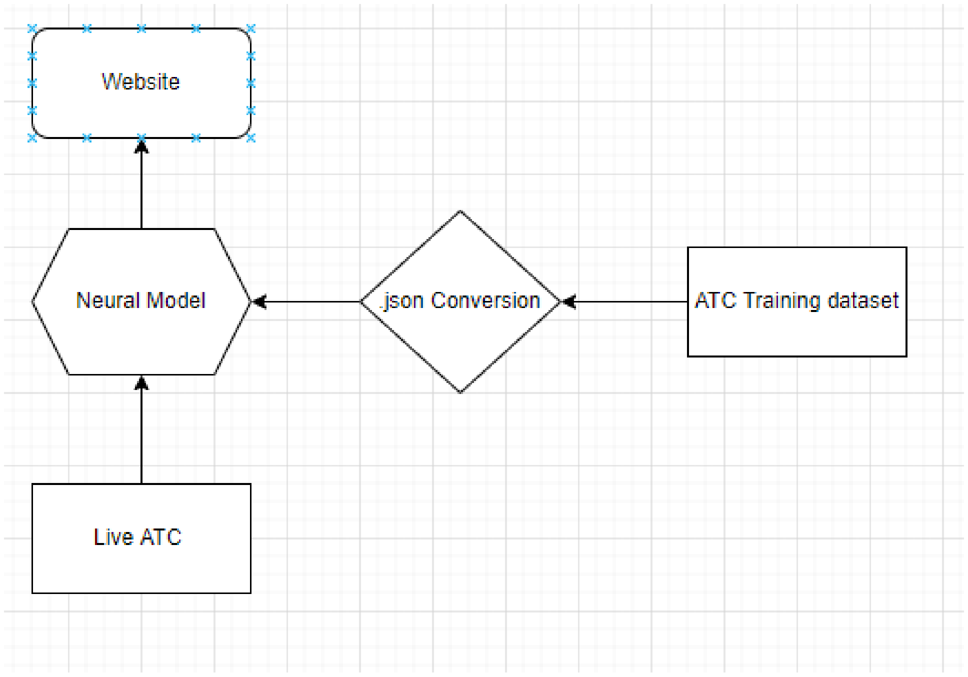
5.2 Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

- The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
- Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
- Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
- Graphical representation of the connectivity between systems, showing the direction of data flow
- Query and response descriptions

File system interfaces will be handled using standard python file system functions as well as standard .json and .wav functions. The ATC training data that is being used to train the networks 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 file that points to a .wav file. The live ATC will be drawn from the online resource and input into the neural models as a .wav file so no conversion is necessary there. However, downloading it from that online resource will require interfacing with more 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.



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