System Design Document

For

RTube NeMo Team

Team members: Taylor S., Adam F., Maegan Lucas, Darian Hopkins

Version/Author	Date
1/ Adam, Darian, Maegan	9/28/23

Previous Version: Speech Recognition for Air Traffic Control (Capstone 2022-2023)
Previous Version Team members : Braeden Burnett, Jakob Haehre, Kira McFadden, Tyler Carr

TABLE OF CONTENTS

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this document is to describe and communicate the system requirements of the NeMo Team. This team will be working on the development and integration of a database to store ATC communications data and be able to effectively display the data through a web application.

This project capsulizes two teams to work on this ongoing project. The Kaldi team, which is in charge of developing and innovating upon the existing AI technologies to be able to convert ATC (Air Traffic Control) chatter to text. The purpose of this is to be able to utilize text data to be able to better understand communications for pilots, ATC Controllers & other relevant parties. The NeMo team focuses on integrating this AI model into a web application as well as developing a database to be able to store ATC and flight data.

1.2 Project Executive Summary

1.2.1 System Overview

There are two features that are in this system:

1. *ASR Web:*

This gives the user the ability to switch between a VFR Aeronautical map and an interactive map. These two maps include a zoom in and zoom out function with the ability to click on the plan icons. With the updating tracking, the user has the ability to click and track all the aircrafts or airports. When an aircraft is clicked, it will display The flight path of the plane along with multiple circles showing where the aircraft was communicated. On the side of the screen will display the aircraft's properties such as a callsign, position and heading. When an airport is clicked on, a menu will display the information about the airport and a drop down menu of all the live radio frequencies.

2. *Transcription:*

This is a database of LiveATC that will be used to transcribe what was communicated between ATC and the aircraft into text. The datasets will be training the model what it should properly transcribe when listening to the audio. The transcription should be able to pick up what ATC is telling the aircraft and translate it into text.

1.2.2 Design Constraints

To be determined

1.2.3 Future Contingencies

The main contingency that could arise would be implementing Team Kaldi's (the other team working on the project) data into Team NeMo's code. Another contingency would be the transcription not properly reading what ATC is telling the aircraft transcribing back false information. One other contingency would be having the wrong icon for the different types of aircraft.

1.3 Document Organization

The System Design Document starts with the introduction section describing in full detail what the project is about and what the purpose of this project is about. After the general overview of the project is discussed, the document goes into a more in depth view of the System Architecture. Then the document goes over the human-machine interface, detailed design, external interfaces, and system integrity controls.

1.4 Project References

1. System Requirements Specification Document

1.5 Glossary

ATC (Air Traffic Control) - Facility used in the United States to detect and manage air traffic.

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

We do not have system hardware for this project. The only hardware we are using are computers with a GPU.

2.2 System Software Architecture

The main system software being used in this is the Nvidia NeMo ASR models which is using Python. Nvidia NeMo is focused on Automatic Speech Recognition which is used by Python's machine learning modules. This cloud-native framework helps deploy generative AI models anywhere including Speech Recognition. The NeMo architecture can be seen in the figure shown below.

(Insert architecture for NeMo)

2.3 Internal Communications Architecture

To be determined

3 HUMAN-MACHINE INTERFACE

3.1 Inputs

The main form of input into the system is the live ATC feeds in the form of datasets. The datasets are given in .json files. These .json files lead to .wav audio files. Live ATC data will be pulled from online resources. The user will give input to the system with screen interaction on the map. Using buttons, the user can choose between a geographical map sourced from online resources and eventually an aeronautical type chart. The source of this chart is still to be determined.

3.2 Outputs

The system will output a Flask-based website containing a geographical map or aeronautical chart, determined through user input, with icons displaying the locations of airplanes and airports. When one of these icons is clicked, an information popup will be displayed with the selected airplane or airport's information and live audio frequencies.

4 DETAILED DESIGN

4.1 Hardware Detailed Design

No hardware will be designed by the NeMo group. Most required hardware is already embedded in an average laptop or personal computer. A graphics card will also be needed.

4.2 Software Detailed Design

Flask, OpenSky API, and OpenStreetMap are required for the NeMo website. These dependencies also include Python version 9 or greater. Along with Flask, Pandas version 2.0.1 and OpenPyxl version 3.1.2 are also required.

4.3 Internal Communications Detailed Design

The NeMo website interfaces with the OpenStreetMap and OpenSky APIs for maps and airspace information. FlightRadar24 provides all live ATC communication. A transcription of this audio will soon be provided with Kaldi modeling. A database will be created to hold all communication data that will be queried based on user input to pull specific airplane or airport data.

5 EXTERNAL INTERFACES

5.1 Interface Architecture

A database will be developed to hold all communication data. This database will be queried from user selection of airplane and/or airport icons to produce the relevant data for the selection.

5.2 Interface Detailed Design

Data will be stored and queried using a database. This database will store an audio file of communication, a text transcription of the audio, the date the communication took place, the time the communication took place, the location that the communication took place, and the callsign of the party making the communication. This will be queried when a specific airplane or airport location is selected by the user to find the matching data.

6 SYSTEM INTEGRITY CONTROLS

To be determined