System Design Document

For

RTube NeMo Team

Team members:
Maegan Lucas, Maxwell Moolchan, Darian Hopkins, Taylor Sumlin, Adam Fitch

Version	Author	Date	Description of Changes
1.0	Maegan Lucas	26 September, 2023	Initial creation of document
1.1	Maegan Lucas	26 September, 2023	Updated document name
1.2	Adam Fitch	28 September, 2023	Added Section 1.2, Edited Section 1.2
1.3	Darian Hopkins	28 September, 2023	Added Sections 1.4, 3.1, 4.2, 4.3, 5.1, 5.2, 6, Team Member Names, and gave credit to previous team
1.4	Adam Fitch	29 September, 2023	Added Section 1.3, Edited Sections 1.2, 1.2.1
1.5	Adam Fitch	29 September, 2023	Added Section 1.2.3
1.6	Taylor Sumlin	29 September, 2023	Updated Table of Contents
1.7	Adam Fitch	29 September, 2023	Added Sections 2.1, 2.2
1.8	Maegan Lucas	29 September, 2023	Added Section 1.3.2, 1.6, 4.1, Edited Sections 4.2, 5.1, 5.2, Minor text edits through entire document
1.9	Maegan Lucas	29 September, 2023	Edited Sections 4.3, 5.1, 5.2, 6
2.0	Maxwell Moolchan	30 October, 2023	Edited Team names to be full names, Updated table of contents, added detailed log of all document changes up to this point, added page numbers & revision comments
2.1	Maxwell Moolchan	30 October, 2023	Edited Section 1.2, 1.2.3, 1.5, 2.1, 4.2, Minor edits through the entire document
2.2	Maegan Lucas	31 October, 2023	Edited Section 1.1, 1.2
2.3	Maxwell Moolchan	31 October, 2023	Added Term to Section 1.5
2.4	Maxwell Moolchan	31 October, 2023	Added Section 1.2.2, Edited Section 1.4
2.5	Darian Hopkins	31 October, 2023	Added Diagram to Section 4.2, 4.3
2.6	Maegan Lucas	31 October, 2023	Added ER Model and made final updates
2.7	Maegan Lucas	6 November, 2023	Added diagrams

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

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) speech 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 NVIDIA NeMo models, and in the future, the Kaldi model, into the web application as well as developing a database to be able to store audio communications between aircraft and ATC along with airport and flight data.

1.2 Project Executive Summary

The project was planned to be developed over two semesters using the Agile Scrum model during the development of the project. Each semester was divided into 3 sprints varying in length. Semester 1's focus was integrating the concepts learned from the previous group and building those concepts in software that was faster and better suited for the purposes of this app, with the primary example being moving the application's user interface from Python to JavaScript & HTML and the implementation of a database. Another focus of Semester 1is to update the user interface with new features such as different icons for different aircraft categories.

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 series VFR Aeronautical map and a geographic map. These maps include a zoom in and zoom out function with the ability to click on the plane icons, providing details about the associated aircraft such as callsign, position, and heading on the left side of the interface. 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. When an airport is clicked on, a menu on the left side of the interface will display the information about the airport and a drop down menu of any available live radio frequencies.

2. Transcription:
This is a database of Live Aero that will be used to transcribe what was communicated between ATC and the aircraft into text. The transcription should be able to pick up what ATC is telling the aircraft and translate it into text.

Figure 1 depicts the use-case diagram for the NeMo system. It shows in more detail what interacts within the system to provide the features of NeMo. While there are different types of users intended to interact with the NeMo system, they currently are able to access the same functionalities, so they have been simplified into one user for the sake of the diagram.

The user is able to *View Live Maps*. This automatically includes *View Live Geographic Map* as that is the default map setting. An extension of this case is *View Live Aeronautical Maps*, an optional map selection. The following APIs interact with this case internally: OpenSky, VFRMap, and OpenStreetMap.

Users are able to *Listen to Live ATC Audio*. Internally, Near Aero provides the audios. Similarly, users are able to *Transcribe Live Audio*. Near Aero provides the audio, the NVIDIA NeMo application transcribes the audio which is then saved to the Communications database, which is then displayed to the user.

As more functionalities are added to the system, and those functionalities start to vary between user types, the Use-Case Diagram will be updated accordingly.

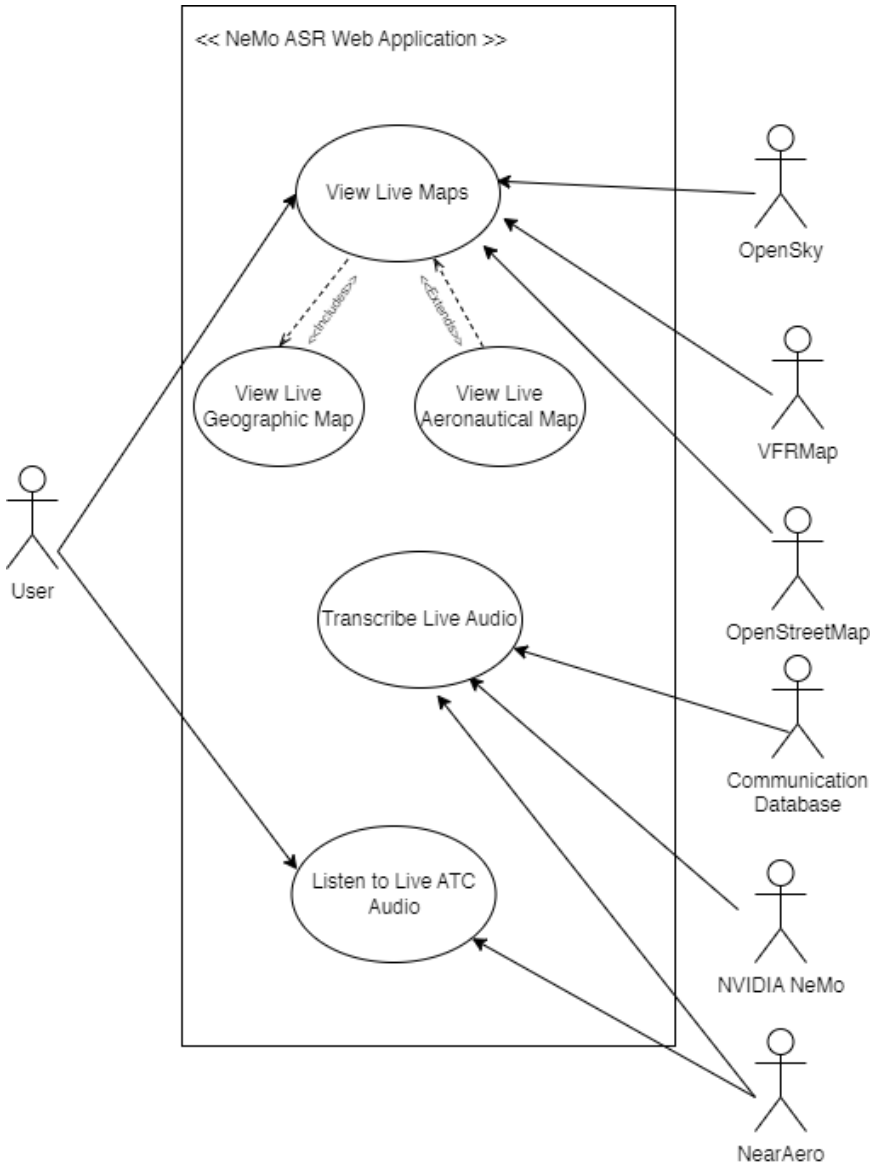

Figure 1: Use-Case Diagram for NeMo system.

1.2.2 Design Constraints

One of the key assumptions we made is that the data obtained from the APIs such as Opensky and the tilesets for the maps from OpenStreetMap and VFRMap are sufficiently correct information to be displayed to the user and that the audio from Near Aero and other sources is the correct audio communication that is to be translated to text. We also are aware of current limitations regarding ASR technology still having a sufficiently high WER that may result in incorrect and or unreadable text. We concluded that based on the ASR limitations the end user should be made aware of these limitations and display the limitations on the “about” page of the website.

Another key constraint while designing and creating this web application is the requirement for developers, testers, and the machine acting as a host to have a NVIDIA GPU in order to run NVIDIA NeMofor the ASR models even though we are not creating or training any models.

Additionally when developing this web application there was a tradeoff between getting certain systems working and getting them working efficiently for the majority of end users. This includes the optimization and balance between taking more memory and calling data from sources such as the database or an API. Many of the APIs have limitations on how often they can be called.

1.2.3 Future Contingencies

The primary contingency that could arise would be implementing Team Kaldi’s (the team working on one of the ASR models) model into Team NeMo’s code. A potential way to overcome this would be waiting until Team Kaldi has a usable and stable model that has support to take in a stream of audio and output text. Currently the application will use NVIDIA NeMo running a trained model specifically for ATC Communications. Team Kaldi’s model will either replace or work in conjunction with the NVIDIA NeMo model when ready.

Another contingency that will likely arise is speech-to-text transcriptions or other data not being fully accurate, resulting in incorrect text and/or false information being displayed to the end user. Due to current limitations with Automatic Speech Recognition (ASR) technology a low or near-zero Word Error Rate (WER) is difficult to achieve. A good ASR Model using current technology can only achieve a WER of about 0.5 - 0.10 (about 5% to 10% error rate). In the application “about” page end users will find information that transcriptions and data including aircraft type, speed, altitude, and position pulled from third-party APIs are not guaranteed to be 100% correct, and should not be used as the only source of information for a pilot. The FAA and official audio channels should always be used as the primary source of information.

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

Akbas, M. I. (2023). *System Design Document For <Project Name>*.

Team NeMo 2023 - 2024. (2023). *System Requirements Specification Document.* https://github.com/maeganlucas/CS490-ATC/blob/main/NeMo/Documents/System%20Requirements%20Specification%20-%20NeMo%20-%20V2.docx.pdf

Team NeMo 2023 - 2024. (2023). System Test Plan. https://github.com/maeganlucas/CS490-ATC/blob/main/NeMo/System%20Test%20Plan%20-%20NeMo.pdf

LeafletJS. (2023). *LeafletJS Reference.* https://leafletjs.com/reference.html

Flask. (2023). *Flask User's Guide.* https://flask.palletsprojects.com/en/3.0.x/

Opensky Network. (2023). *Opensky Network API Documentation.* https://openskynetwork.GitHub.io/opensky-api/index.html

Sphinx. (2023). *Sphinx Documentation.* https://www.sphinx-doc.org/en/master/

GitHub. (2023). *NVIDIA NeMo GitHub.* https://GitHub.com/NVIDIA/NeMo

NVIDIA. (2023). *NVIDIA NeMo tutorials.* https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/starthere/tutorials.html

Team NeMo 2023 - 2024. (2023). *Team NeMo documentation GitHub.* https://GitHub.com/maeganlucas/CS490-ATC

Team NeMo 2023- 2024. (2023). *Team NeMo active GitHub.* https://GitHub.com/TheCreepOfWar/asr-webapp

Team NeMo 2022 - 2023. (2023). Team previous NeMo team GitHub. https://GitHub.com/Burnetb8/Senior-Capstone

1.5 Glossary

#	Term	Abbreviation	Description
1	Air Traffic Control	ATC	Refers to tower communication to or from aircraft.
2	Automatic Speech Recognition	ASR	Software that recognizes speech patterns.
3	Word Error Rate	WER	Refers to the error rate in ASR technology typically on a scale of 0 to 1 but can exceed 1 depending on sample comparisons.
4	Federal Aviation Administration	FAA	United States government agency that oversees and regulates civilian aviation in the United States.
5	NVIDIA NeMo		NVIDIA NeMo is a generative AI framework that allows for conversational AI to function and transcribe ATC communications within the web application.
6	Graphical Processing Unit	GPU	A hardware component used to improve computer graphics.

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

The end user (client) does not require any special hardware to load the application.

The server or machine acting as the host for the project (our computers during development) will require a NVIDIA GPU in order to run the NVIDIA NeMo models, otherwise the speech-to-text transcription part of this project will not function.

Despite this, the user is still capable of tracking aircraft across the map and has the ability to listen to live ATC communications.

2.2 System Software Architecture

A user will interact with the NeMo system through the frontend of the application. This will make the appropriate requests to the backend portion of the system.

For visual maps, the backend will make requests to the respective APIs, OpenStreetMap for geographic maps and VFRMap for aeronautical maps, and display them to the user. To display the active planes, the backend will make request as necessary to the OpenSky API to determine plane data. The system will then complete the necessary logic to add a plane icon of the correct type to the map and display it to the user. The system will also query the

Communications database when an airplane is selected to pull the waypoint data for the specified plane for display.

The main system software being used in this is the NVIDIA NeMo ASR models which are using Python. NVIDIA NeMo is focused on Automatic Speech Recognition which is used by Python's machine learning modules. Using NVIDIA NeMo, transcriptions are able to be made of live ATC audio. This will occur when a user selects a transcription button on the frontend. From there, the backend portion of the system will make a query from the database for the audio. It will be passed into NVIDIA NeMo to be transcribed and this will be added to the database entry for the audio. This entry will then be displayed to the user.

The NeMo architecture can be seen in the *Figure 2* below.

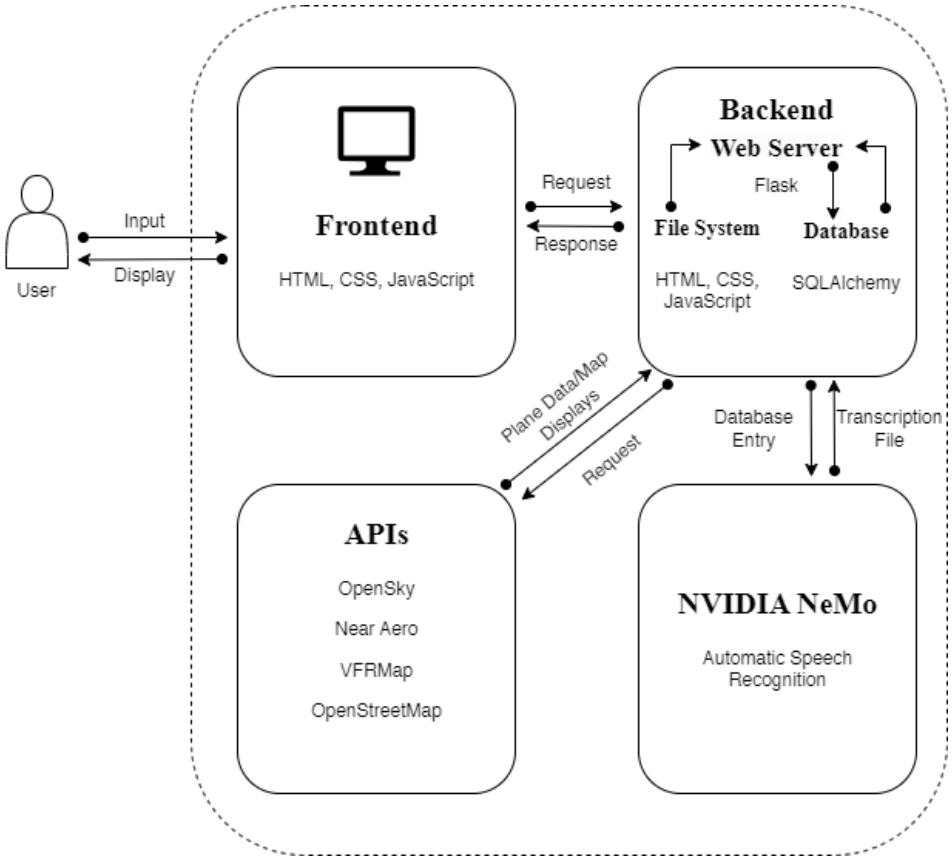

Figure 2: System Architecture Diagram for NeMo.

2.3 Internal Communications Architecture

The internal communication architecture for the NeMo web application is depicted in *Figure 3*. The user of the system represents one source of the diagram as they input airplane, airport, and button selections into the system. Other sources include the various APIs that the application utilizes. OpenSky is a source for the live plane data. OpenStreetMap and VFRMap are the sources for the geographic and aeronautical map tiles respectively. Near Aero provides the live ATC audios to the system. The Communication

Database developed by the team can also be considered a source as plane and audio data are queried from the entries within the database into the system to be transcribed by NVIDIA NeMo. The database will also provide the written transcriptions back to the system to be displayed to the user.

The sinks for this system are the Communication Database and the Display to the user. For the Communication Database, transcriptions for given audios are then saved to their respective entry within the database. The Display recieves all visual aspects such as airplane and airport icons, airplane and airport details, flight paths, transcriptions, and waypoint indicators.

More detailed diagrams of how data flows within the system can be seen in section 4.3, *Internal Communications Detailed Design*.

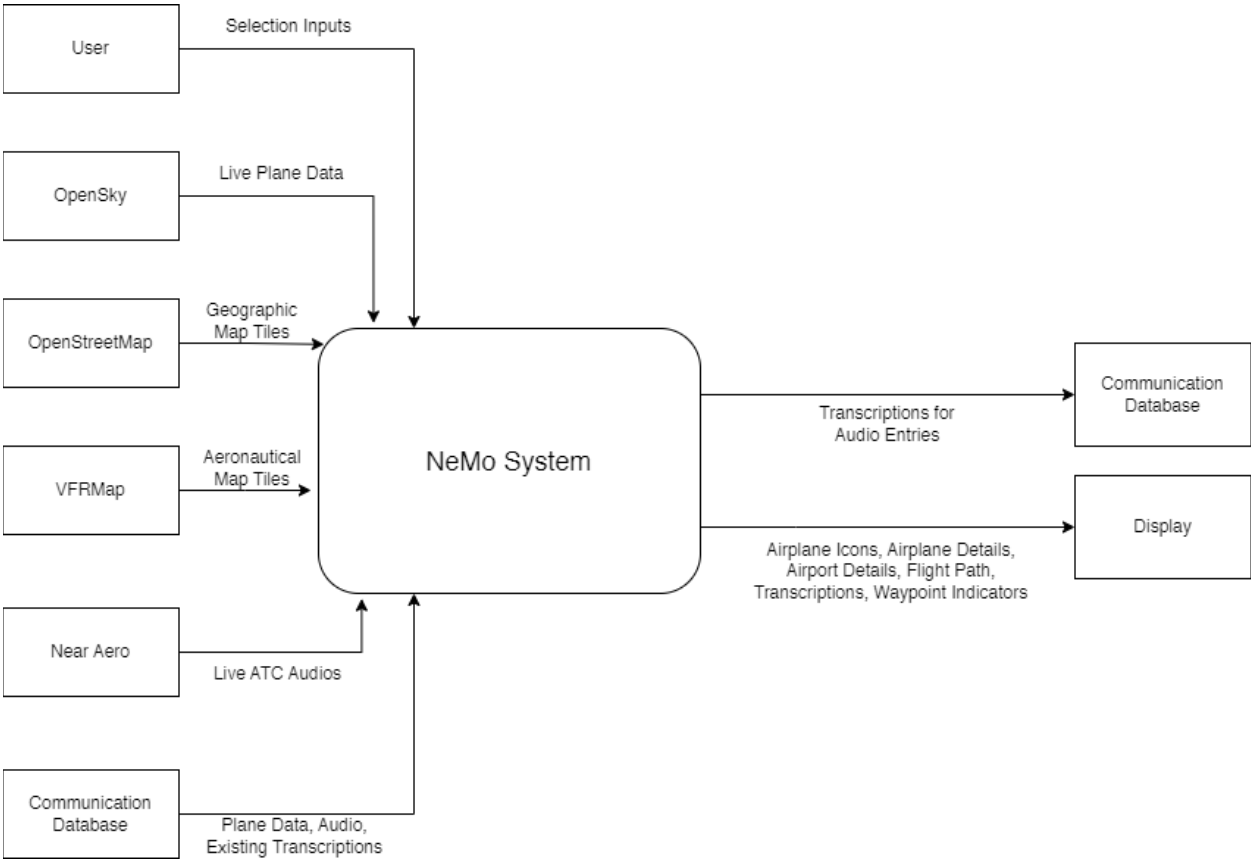

Figure 3: Context Diagram for NeMo system.

3 HUMAN-MACHINE INTERFACE

3.1 Inputs

The main form of input into the system is the live audio feeds in the form of streamed .mp3 files which will later be converted to .wav files. The live audio feeds will be pulled from various online sources such as LiveATC and Near Aero. 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 Openstreetmap.org, and a series of aeronautical maps, from VFRMap.com.

3.2 Outputs

The system will output a Flask-based website containing a geographical map and a series of aeronautical charts, determined through user input, with icons displaying the locations of airplanes and airports. When one of these icons is clicked, an information popup on the left side of the interface will be displayed with the selected airplane or airport's information and any available live audio frequencies. For airplanes, this information will include current location, arrival and departure airports, aircraft category, call sign, and more. For airports this information will include location, website, and more.

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 NVIDIA graphics card will be needed on the server or machine acting as the host for the project, otherwise the speech-to-text transcription part of this project will not function.

4.2 Software Detailed Design

Python version 9, Pandas version 2.0.1, and OpenPyxl version 3.1.2 are required for the backend to operate.

Flask web framework for creation and framework of the application, managing and creating templates, pages, communication between front end HTML interface and python and JavaScript scripts which do backend processing and communication with the other APIs.

OpenskyAPI is used to obtain current aircraft in the air and data associated with each plane including: icao24, callsign, origin country, time position, last contact, longitude, latitude, geospatial altitude, ground status, velocity, true track (heading), vertical rate, squawk, position source, and category of the aircraft.

LeafletJS is the API we use to create, display, and control the maps themselves; this includes the creation of the Icons on the map, the base which tilesets are connected to, and the zoom controls.

OpenStreetMap is used for the tileset (set of images that comprise the map) for the geographical map display & VFRMap is used for the tileset for the various aeronautical maps.

NVIDIA NeMo is the primary ASR model software used. We will not be creating or training the models as part of this project, that is for other teams to do. This project however will take predefined classes and functions NVIDIA NeMo provides to take in and process audio files and streams and provide a text output given a model created and trained by other teams.

4.3 Internal Communications Detailed Design

The NeMo website interfaces with the OpenStreetMap, VFRMap, and OpenSky APIs for maps and airspace information. Near Aero provides all live ATC communication. A transcription of this audio will be displayed to the user after being passed through the NVIDIA NeMo ASR model. A database will be created to hold all communication data that will then be queried based on user input to pull specific airplane or airport data, audios and audio data, and transcriptions. Below are a level zero (*Figure 4*) and level one Data Flow Diagram (*Figure 5*) representing the flow of data within the ASR web application.

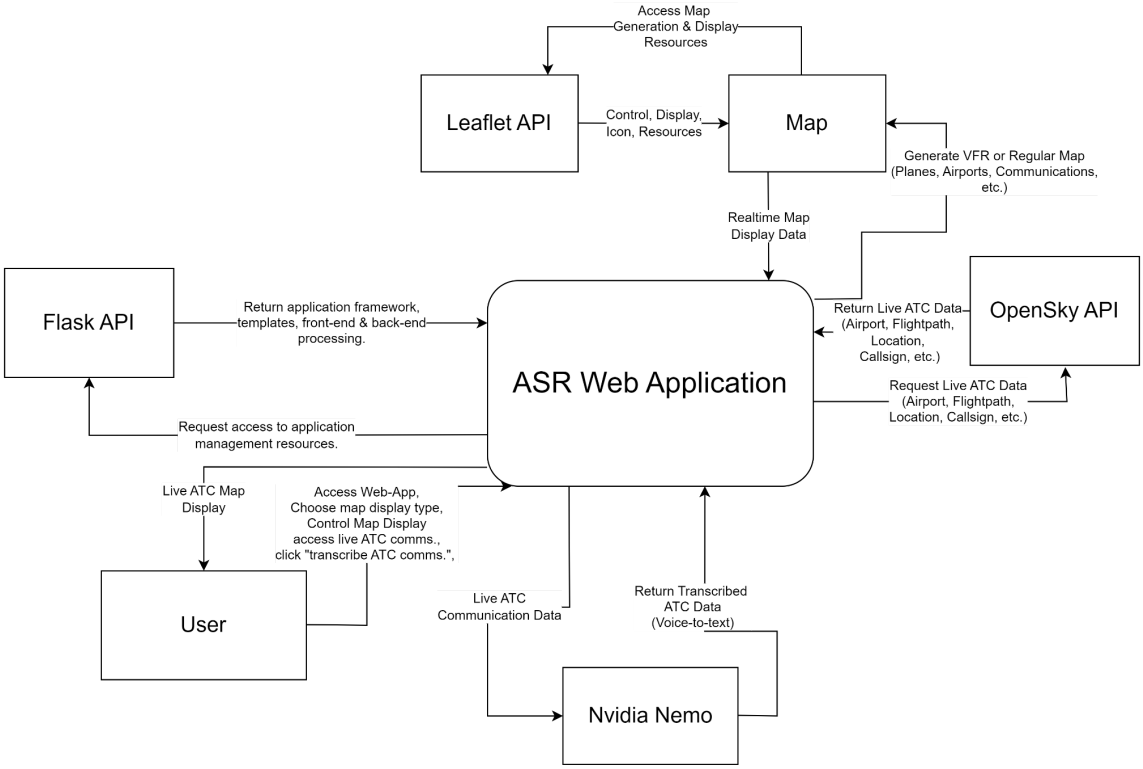

Figure 4: The above diagram is a high level view detailing the data communications between the "components" of our system.

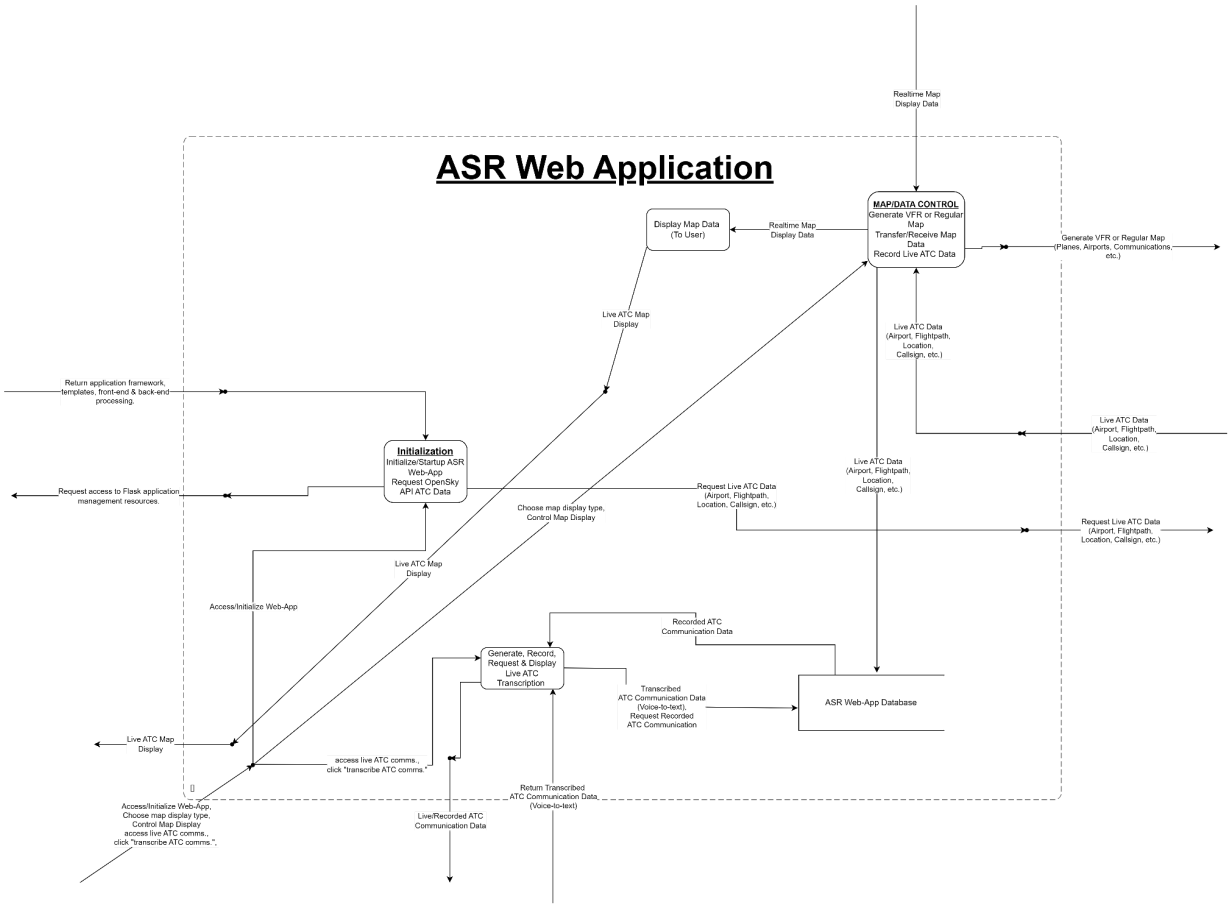

Figure 5: The above diagram is a level 1 Data Flow Diagram detailing the processes and flow of data within the ASR Web Application (please zoom for clarity or see attached documentation).

Internal ASR-Web Application Processes:

- Map/Data Control - Handles the transfer of vital map data. Directly Interfaces with Map source and indirectly interfaces with the Leaflet API.
- Display Map Data - Displays readable map data to user.
- Generate, Record, Request & Display Live ATC Transcription - Sends and receives live ATC voice data (to be transcribed) to Nvidia Nemo. Sends newly transcribed data to the internal ASR Web-App database. In addition, can request previously recorded voice communications from the internal ASR Web-App database
- Initialization - Contacts the Flask API and OpenSky API and requests required resources for NeMo website application initialization.
- ASR Web-App Database - Stores ATC communications and flight data from OpenSky API. In addition, stores transcribed ATC communications received from Nvidia NeMo.

The Entity-Relationship Model for the communication database can be seen in *Figure 3* below.

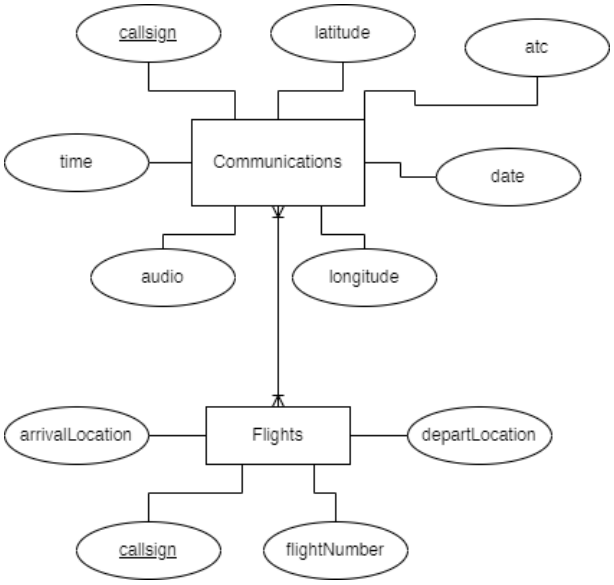

Figure 6: Entity-Relationship Model for the communication databases.

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 and location that the communication took place, the callsign of the party making the communication, and a Boolean value indicating whether the party making the communication is ATC. 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

This project does not require system integrity specifications as it does not handle any sensitive data.