Versatile Autonomous Navigation Testing and Guidance Environment
System Requirements Specification
Version 0.6
10/26/2024

Document Control

Distribution List

The following list of people will receive a copy of this document every time an updated version of this document becomes available:

Teaching assistants:

Alejandro Gonzalez Nunez

Customer(s):

Dr. M. Ilhan Akbas
Alejandro Gonzalez Nunez

Project team members:

Jim Pamplona
Jona Ortiz
Mai Evans
Jack Lee

Change Summary

The following table details changes made between versions of this document:

Version	Date	Modifier	Description
0.1	10/25/2024	Jack Lee	Added title, version, distribution list
0.2	10/25/2024	Jim Pamplona	Fixed formatting
0.3	10/26/2024	Jim Pamplona, Jona Ortiz, Jack Lee	Edited 1.1, 1.2, 1.3, 1.4, 1.5, 1.5.1, 1.5.2, 1.5.3, 2.1, 2.2, 2.3, 2.3.1, 2.3.2, 2.3.3, 2.5, 2.6, 2.7, 3.2, 3.3, 3.4
0.4	10/28/2024	Jim Pamplona, Jona Ortiz	Edited 4.1,4.2,4.3,4.4,4.5,
0.5	10/29/2024	Mai Evans, Jack Lee	Edited/proofread: TOC, 1.2, 1.3, 1.4, 1.5.2, 2.3.3, 2.4 3.3, 4.1, 4.2, 4.3, 4.4, 4.5, 5.1, 5.2, 5.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.4
0.6	10/31/2024	Jim Pamplona	Updated Use cases and DFDs

Table of Contents

1. Introduction

1.1. Purpose and Scope

As uncrewed aerial systems (UAS) become more common in both commercial and military applications, it is critical to ensure that they can avoid collisions with other aircraft, structures, and obstacles in complex environments. Also, traditional testing methods can be time-consuming, costly, and lack the scalability required to cover a wide range of potential collision scenarios. The Versatile Autonomous Navigation Testing and Guidance Environment (VANTAGE) system aims to solve the need for more reliable and efficient testing and validation of collision avoidance for UAS. It enables the researchers and analysts of the domain to gain a clear understanding of the

The purpose of the System Requirements Specification (SRS) document is to provide a detailed description of the specific functionalities, purposes, interactions of the VANTAGE system. The scope of this system encompasses a two-tiered simulation approach to ensure accurate and rigorous testing of the system's collision avoidance.

1.2. Intended Audience and Reading Suggestions

This document is intended for a variety of stakeholders involved in various aspects of the system's development, deployment, and use. Each type of stakeholder will find specific sections of this document that are more relevant to their roles regarding the system.

- **Developers and Testers:** Sections 2.2, 2.3, 2.3.3, 2.5, 2.6, 3.3, 4, and 6
- **Project Managers**: Sections 1.1, 1.5, 2.1, 2.2, 2.3, 6.1.4, and 6.1.5
- **Documentation Writers**: Sections 1.1, 1.4, 1.5, and 2.3

1.3. Document Conventions

There are no typographical conventions that denote special significance in this document.

1.4. Project References

- Product proposal document
    - *https://github.com/MLM-Simulation-and-Testing-for-UAS/mlmst-uas/blob/main/Docs/Project_Proposal_CS490.pdf*
- Gazebo documentation
    - *https://gazebosim.org/docs/harmonic/install/*
- ROS documentation
    - *https://docs.ros.org/en/humble/index.html*
- ArduPilot documentation

  - o https://ardupilot.org/dev/docs/sitl-with-gazebo.html
- Julia documentation
  - o https://docs.julialang.org/en/v1/

1.5. Definitions, Acronyms, and Abbreviations

Various acronyms, abbreviations, and terms are utilized throughout this document that may be unknown to some (or most) stakeholders of this system. Please refer below to a glossary that specifies the definitions of said terms.

1.5.1. Definitions

This section lists terms used in this document and their associated definitions.

Table 1: Definitions

Term	Definition
Total impulse	A measure of the overall total energy contained in a rocket motor

1.5.2. Acronyms

This section lists the acronyms used in this document and their associated definitions.

Table 2: Acronyms

Term	Definition
VANTAGE	Versatile Autonomous Navigation Testing and Guidance Environment
SDD	System Design Document
UAS	Uncrewed Aerial Systems
ML	Machine Learning
ROS	Robot Operating System
CSV	Comma Separated Values
SITL	Software in the Loop

1.5.3. Abbreviations

This section lists the abbreviations used in this document and their associated definitions.

Table 3: Abbreviations

Term	Definition
e.g.,	For example

2. General Description

2.1. Product Perspective

Uncrewed Aerial Systems (UAS) have slowly become commonplace as more military and commercial applications come to fruition. This project aims to develop an open-source framework for training UAS systems. This framework employs a two-tiered simulation model which shall start with a low-fidelity simulator for complex testing and then process the data created into a high-fidelity 3d simulator for more precise collision avoidance.

This simulation-driven development would provide a safer and more cost-effective solution to testing complex behaviors. With the application development it would be ideal for experimenting with multi-agent scenarios, adjusting parameters, and observing the potential outcome in controlled scenarios. It would allow for users to focus more on development with the benefit of testing without physical limitations. This in turn would eliminate the need for regulatory compliance in the simulation phase and would help researchers and developers explore new UAS dynamics taking into account the constraints of air traffic compliance in later development stages.

The model employs a few existing technologies for its development and implementation, namely ArduPilot, Gazebo, ROS, and the programming languages Julia and Python. ArduPilot is an open source UAS development framework with built-in support for a wide range of drone types, sensors, and environments. Gazebo is a visual simulation software that can interface with ArduPilot to render drone simulations. ROS is a set of software libraries that empower developers to build robotics software. These technologies enable the developers of this project to create simulations, visualize them, and ultimately load them onto physical UAS to test them in the real world.

2.2. Product Features

The main purpose of the system is to provide a rigorous testing environment in a two-tiered simulation approach:

- **Low-Fidelity Simulation (JuliaSim):** Used for rapid scenario testing to evaluate initial strategies and identify potential issues.
- **High-Fidelity Simulation (Gazebo & ArduPilot):** Provides a more detailed and realistic simulation for in-depth analysis and fine-tuning.

The system's design aims to optimize testing efficiency by combining the strengths of both low-fidelity and high-fidelity simulations, allowing users to seamlessly transition between them while refining and validating their strategies. The two approaches are integrated to enable simultaneous and complementary testing, enhancing the evaluation and development process.

Users can create and manage different testing scenarios, including predefined and custom scenarios. The system supports storing, loading, and modifying scenarios for repeated testing

or refinement. It facilitates iterative testing and development of UAS collision avoidance strategies by enabling scenario reuse and modification.

The system offers a user-friendly graphical interface for setting up and running simulation scenarios. Users can input telemetry parameters, such as speed, altitude, and heading, and easily configure testing conditions. It simplifies the creation of test scenarios and allows users to visualize the inputs and outputs of the simulations in real-time.

2.3. User Classes and Characteristics

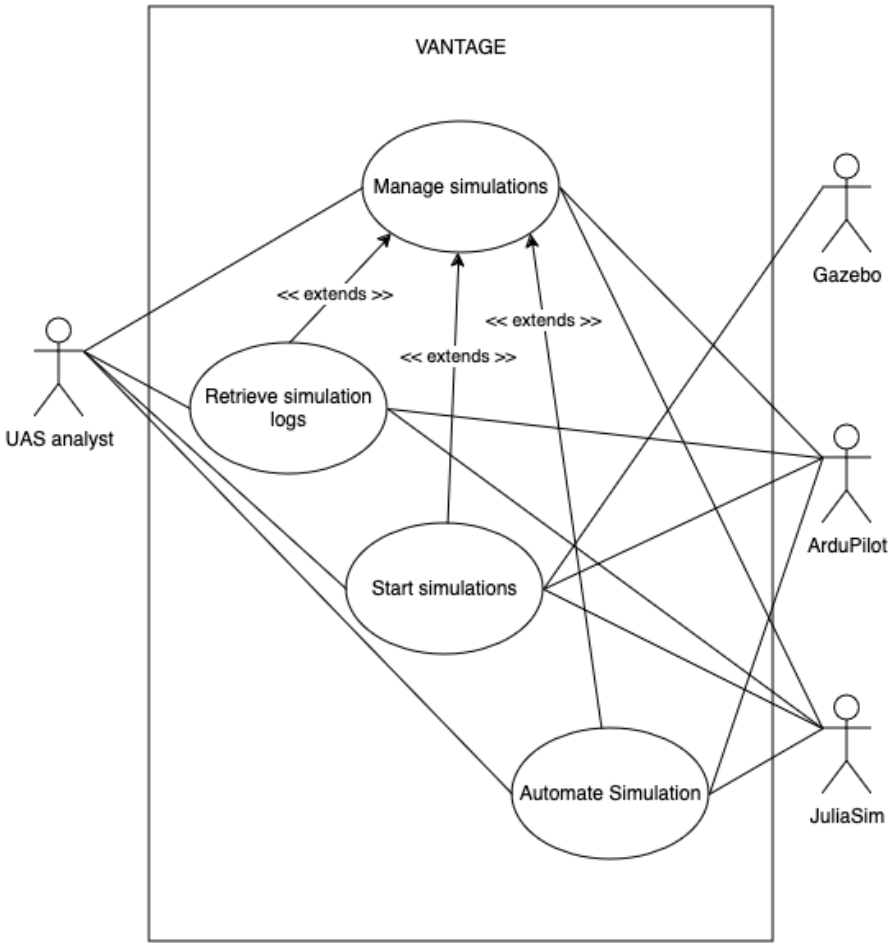


Figure 1. Use Case diagram of the System.

2.3.1. Actors

This section presents the actors in the system.

- UAS Analyst
    - End user. Responsible for running, managing, and analyzing simulations completed on the system. Reviews logged data for trends, issues, and patterns.

Provides insights for further development and optimization of UAS collision avoidance strategies.

- Developer
    - Responsible for ensuring the integrity and development of the system and updating dependencies to best support external entities such as Gazebo, ArduPilot, and JuliaSim.
- Gazebo
    - Displays the 3D environment in which the simulation is being run.
- ArduPilot
    - Acts as the brains of the high-fidelity simulations, as it provides real-time telemetry data for the said simulations.
- JuliaSim
    - Runs and displays low-fidelity simulations. It also provides telemetry data for the said simulations upon completion.

2.3.2. Use Cases

This section presents the Use Cases, developed for the system.

- Manage simulations
- Start simulation
- Automate simulations
- Retrieve simulation logs
- Manage system

2.3.3. Scenarios

Scenario 1: Manage Simulations

Description: The user creates, views, edits, or deletes a simulation

Actors: Current User (can be an UAS Analyst or Developer).

Precondition: The system has been launched successfully.

Trigger Condition: The user chooses to create, view, edit, or delete a simulation.

Steps:

1. The user selects the Create Simulation button (ALT 1, 2, and 3)
2. The system presents the user with a list of parameters and inputs
3. The user edits the parameters to create the simulation
4. The system saves the parameters to a simulation file
5. End of use Case.

(ALT 1: View Simulation)

1.1 The user selects a simulation
1.2 The system displays View, Edit, and Delete buttons
1.3 The user selects the View Simulation button

1.4 The system displays the simulations parameters
1.5 Use case continues at Step 5

(ALT 2: Edit Simulation)

1.1 The user selects a simulation
1.2 The system displays View, Edit, and Delete buttons
1.3 The user selects the Edit Simulation button
1.4 The selected simulation's parameters are displayed to the user
1.5 The user edits the parameters
1.6 Use case continues at Step 4

(ALT 3: Delete Simulation)

1.1 The user selects a simulation
1.2 The system displays Run, View, Edit, and Delete buttons
1.3 The user selects the Delete Simulation button
1.4 The selected simulation's file is deleted from the computer
1.5 Use case continues at Step 5

Scenario 2: Start Simulations

Description: The user starts a simulation

Actors: Current User (can be an UAS Analyst or Developer).

Precondition: The system has been launched successfully.

Trigger Condition: The user chooses to run a simulation.

Steps:

1. The user selects a simulation
2. The user selects the Run Simulation button
3. The system loads the simulation into ArduPilot, Gazebo, and JuliaSim
4. Gazebo displays the 3D high-fidelity simulation to the user
5. JuliaSim displays the low-fidelity simulation to the user
6. End of use case

Scenario 3: Automate Simulations

Description: The user runs an ML-based simulation

Actors: UAS Analyst, Gazebo, ArduPilot, JuliaSim

Precondition: The system has been launched successfully.

Trigger Condition: The user chooses to run an ML-based simulation

Steps:

1. The user selects a simulation
2. The system loads an ML-based simulation into ArduPilot, Gazebo, and JuliaSim
3. Gazebo displays the 3D high-fidelity simulation to the user

4. JuliaSim displays the low-fidelity simulation to the user
5. End of use case

Scenario 4: Retrieve Simulation Logs

Description: The user views log history of a simulation

Actors: UAS Analyst, Gazebo, ArduPilot, JuliaSim

Precondition: The system has been launched successfully.

Trigger Condition: The user chooses to view a simulation's logs

Steps:

1. The user selects a simulation
2. The user selects the View Logs button
3. The logs of past simulation runs are displayed to the user

(ALT 1: No logs are available)

1. The user selects a simulation
2. The user selects the View Logs button
3. A message indicating that no logs are available is displayed to the user

Scenario 5: Manage System

Description: The user performs maintenance on the system

Actors: Developer, Gazebo, ArduPilot, JuliaSim

Precondition: The system has been launched successfully and has changed due to an update, bug, or error

Trigger Condition: The user chooses to enable Developer Mode

Steps:

1. The system displays documentation to download and modify the system's source code
2. The user follows the documentation to modify the system as needed

2.4. General Constraints

Several key constraints will affect the project's progress.

1. System Accessibility: The system will be available in field applications and will be created as open-source software.

3. Framework and Middleware Compatibility:

- Since the Gazebo version "Harmonic" preserves compatibility with both Ubuntu 22.04 and a Long-Term Support (LTS) version of ROS2 (Humble), it must be used as the simulation environment.
- ROS2 version "Humble" is the only ROS version that works with Ubuntu 22.04 and Gazebo Harmonic, so it must be the designated middleware.

4. Programming Languages:

- The low-fidelity simulation is made with the Julia programming language.

5. Additional Software Requirements: The use of ArduPilot and Software in the Loop (SITL) simulation is mandatory, aligning with the project's open-source and field-accessibility goals.

6. Budget: The project will also be constrained to a currently unspecified budget.

2.5. Operating Environment

The system operates on a computer running the Ubuntu 22.04 operating system. The Ubuntu 22.04 operating system may be loaded with or without a hypervisor. The computer should have at least 16 Gigabytes of RAM and 50 Gigabytes of storage available. To use the models, Gazebo Harmonic, ROS Humble, and ArduPilot need to be installed on the computer. The computer's processor should be supported by the forementioned software, and an x86_64 architecture is recommended. The only physical environmental constraints are that the computer should be able to operate safely in the environment (e.g., not submerged in water or subjected to extreme temperatures).

2.6. User Documentation

The documentation includes:

- A guide on manually installing and setting up the software dependencies
- A script to automatically install the software dependencies
- A description of the system's components and their functions

2.7. Assumptions and Dependencies

The development team has made the following assumptions:

- The source code will be provided to the user
- The software dependencies will be available for the user to download and install

3. External Interface Requirements

3.1. User Interfaces

1) Developer Interface:
   - Parameter Input Interface: Defines how the developer wants to set the simulations parameters
   - Result Visualization: Describes how the simulation outcomes are displayed through Gazebo or a data export.

3.2. Hardware Interfaces

At the time of writing, the system does not interact with any physical hardware, so no hardware interfaces are required for the system to function.

3.3. Software Interfaces

Software interfaces that included in the project are as follows:

A) Julia 1.11.1
   a. Source: https://docs.julialang.org
B) ROS Humble Hawksbill
   a. Source: https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html
C) Gazebo Harmonic
   a. Source: https://gazebosim.org/docs/harmonic/install_ubuntu/
D) Python version 3 or higher
   a. Source: https://www.python.org/downloads/release/python-3130/
E) Ubuntu 22.04 Desktop
   a. Source: https://www.releases.ubuntu.com/jammy/

3.4. Communications Interfaces

A) Data synchronization: ensures data is synchronized between Gazebo and Julia.
B) Data Logging and Export (potentially ROS or Julia based)

4. Behavioral Requirements

4.1. Same Class of User

- [REQ-1] The system shall support configuration of agent behavior parameters, simulation environments, and access to detailed data logs.
- [REQ-2] Users with access to the system must have access to features that allow for FAA-compliant parameters, including altitude adjustments, speed and operational limits.

4.2. Related Real-world Objects

- [REQ-3] The system shall simulate real-world drones
- [REQ-4] Obstacles in the simulation shall influence agent navigation and response behaviors.
- [REQ-5] The system shall model environmental conditions such as wind and low visibility which impact the agents' behavior.
- [REQ-6] The system shall have included a calibration module allowing users to adjust the sensitivity of the Environmental stimulus.
- [REQ-7] The system shall simulate a varied terrain.

4.3. Stimulus

- [REQ-8] If an obstacle enters a 10-meter radius of a drone, the drone shall execute a collision avoidance maneuver to prevent impact.
- [REQ-9] If the simulated wind speeds exceed a limit *N* (to be adjusted with testing), the drone shall reduce altitude and speed while adjusting stability parameters.
- [REQ-10] The system shall adjust stability parameters to counter winds.
- [REQ-11] If a drone’s battery level falls 20% the system shall activate a “Return to base” behavior while prioritizing a safe return.
- [REQ-12] Agents shall have a communication feature that enables drones to share position and speed data allowing coordinated navigation. (swarm)
- [REQ-13] In crowded airspaces, the agents shall adjust light paths collaboratively to maintain separation distance.

4.4. Related Features

- [REQ-14] The collision avoidance feature shall continuously monitor for other drones and obstacles at the user’s selected radius.
- [REQ-15] The pathing feature shall allow drones to follow predetermined waypoints.
- [REQ-16] A verification system shall flag any agent actions that deviate from a regulatory standard.

4.5. Functional

- [REQ-17] The system shall validate all input parameters upon initializing.
- [REQ-18] If the FAA compliance system is enabled, the system shall reset any values that exceed safe operational limits.
- [REQ-19] The simulation shall log vital information including (coordinates, altitude, speed, battery level, environmental conditions etc.) in CSV format.
- [REQ-20] Upon detecting any abnormal conditions, the system shall trigger the response sequence and alert the console.
- [REQ-22] All agents shall conduct regular ‘system checks’ to comply with the FAA regulations in “FAA compliance mode”.
- [REQ-23] Users shall be able to input waypoints, specify mission goals, and adjust environmental parameters though a GUI.
- [REQ-24] The system shall log any emergency actions taken (low battery, emergency landing, collision avoidance).
- [REQ-25] The system shall support an increased number of agents for testing the limits of multi-agent coordination
- [REQ-26] The system shall provide a graphical user interface.
- [REQ-27] The system shall allow users to analyze data on agent performance outputting metric charts and summaries.
- [REQ-28] The system shall log each session's data.
- [REQ-29] The session shall produce an “end-of-session” reports summarizing the outcomes of the simulation
- [REQ-30] The system shall implement an FAA compliance mode that ensures the FAA compliances

5. Non-behavioral Requirements

5.1. Performance Requirements

The performance requirements may be overly ambitious considering the known limitations of Gazebo's speed.

5.2. Safety Requirements

Are safety requirements necessary for this project?

5.3. Qualitative Requirements

Should this section include individual requirements, or is it intended as a general overview?

5.3.1. Availability

- [REQ-?] The system shall detect and handle errors gracefully without compromising the entire simulation.
- [REQ-?] The system shall ensure that all logged data is saved consistently.

5.3.2. Security

Is security necessary for the project? It may be outside the project's scope, as security considerations are not currently defined.

5.3.3. Maintainability

Given our use of specific versions, it seems we will not anticipate future updates post-deployment.

5.3.4. Portability

The operating environment is clearly specified, and the system is not designed for portability beyond this context.

5.4. Design and Implementation Constraints

What additional non-behavioral requirements should be considered at this stage? Hardware constraints are anticipated to be developed next semester, as they are currently deemed out of scope and we have general constraints which already affect the design and implementation.

6. Analysis Models

6.1. Data Flow Model

6.1.1. Data Sources

The data sources and their inputs to the system identified in the data flow model are as follows:

- UAS Analyst
  - Command inputs
  - Parameters
  - System issue alerts
- Developer
  - Bug fixes
- Gazebo
  - 3D rendering
- ArduPilot
  - Simulation data
- JuliaSim
  - Simulation data

6.1.2. Data Sinks

The data sinks and their system outputs identified in the data flow model are as follows:

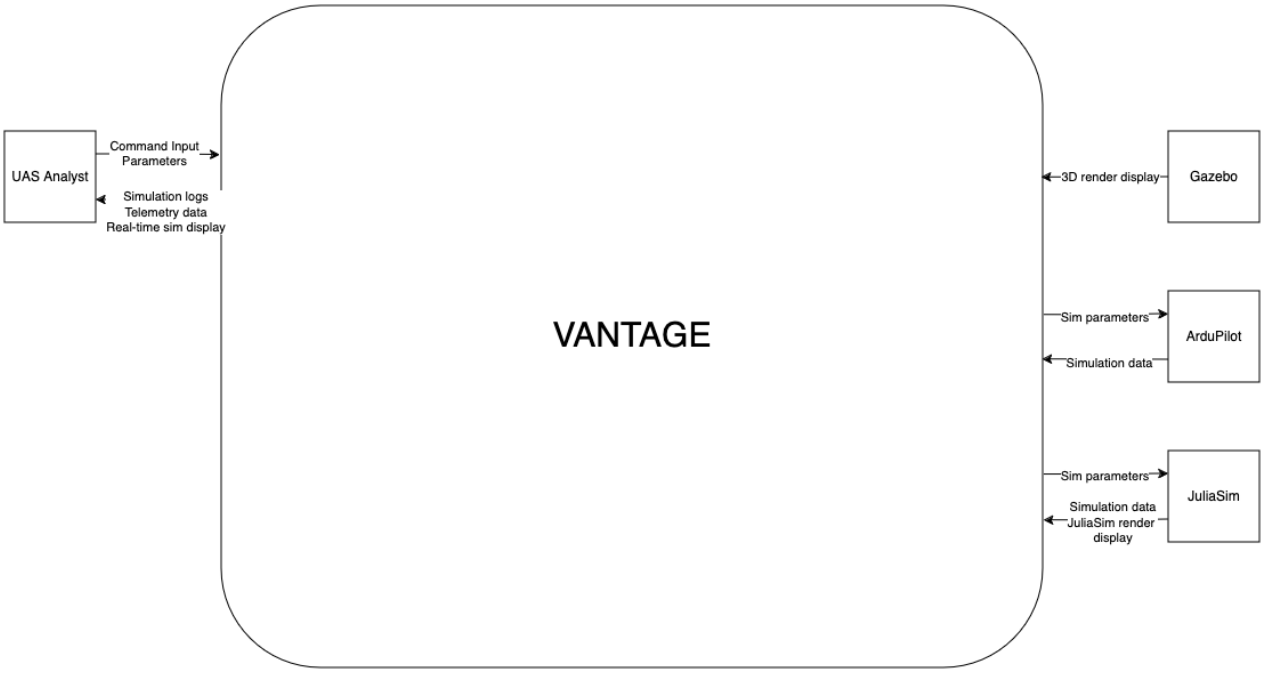
- UAS Analyst
  - Simulation logs
  - Telemetry data
- Developer
  - System issues
- ArduPilot
  - Sim parameters
- JuliaSim
  - Sim parameters

6.1.3. Data Dictionary

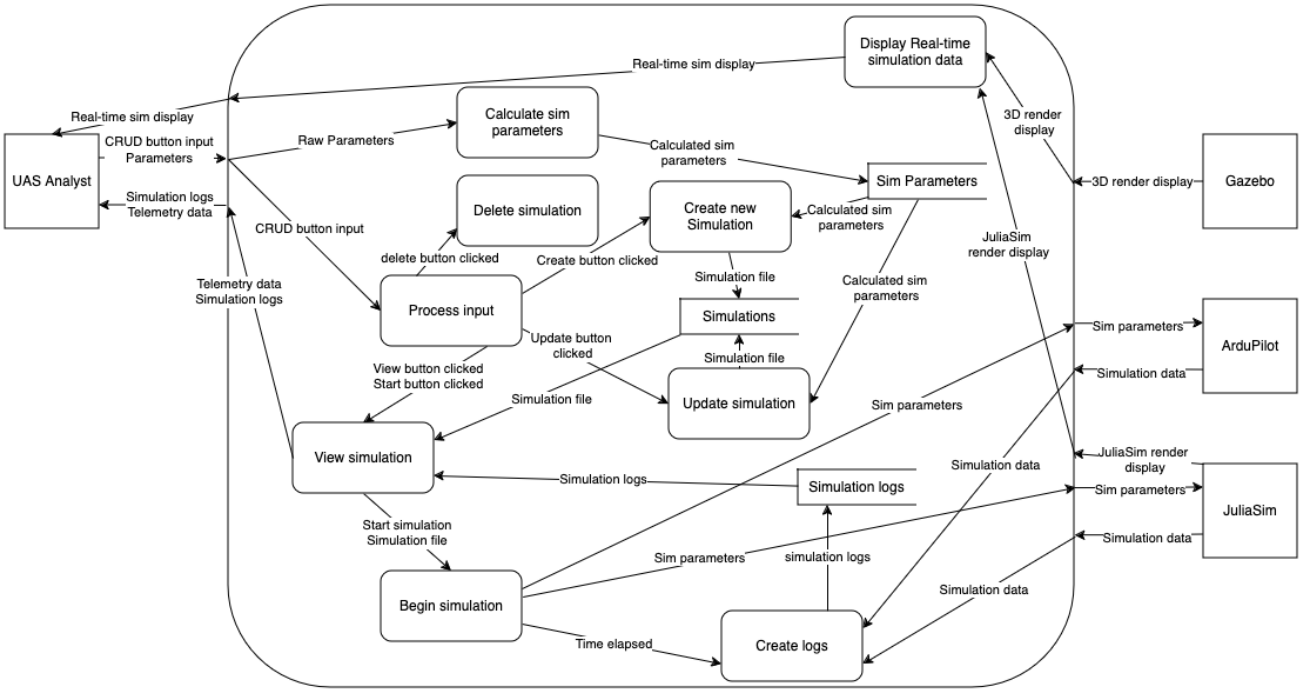
The data types are described in the data dictionary below. This section includes the name of the data type; a description of the contained data; how the data is structured; and the range of values.

Name	***Description***	***Structure***	***Range***
Command input	*Identification numbers are used to differentiate Entities from one another.*	*Positive Integer*	*0<*
Parameters	*Identification numbers are used to differentiate POIs from one another.*	*Positive Integer*	*0<*

6.1.4. Context Diagram (Level 0 Data Flow Diagram)



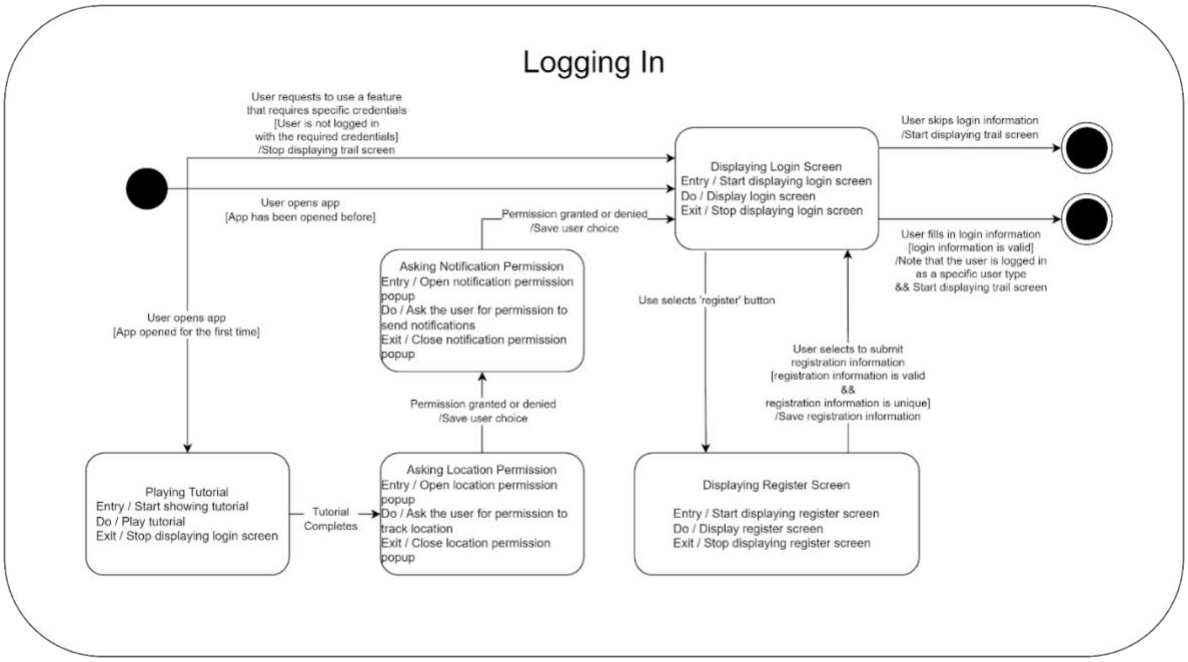
6.1.5. Level 1 Data Flow Diagram


Display Real-time simulation data
Real-time sim display
Real-time sim display
UAS Analyst
CRUD button input
Parameters
Simulation logs
Telemetry data
Raw Parameters
Calculate sim parameters
Calculated sim parameters
Sim Parameters
3D render display
Gazebo
Delete simulation
Create new Simulation
Calculated sim parameters
JuliaSim render display
CRUD button input
delete button clicked
Create button clicked
Simulation file
Calculated sim parameters
Telemetry data
Simulation logs
Process input
Simulations
Sim parameters
ArduPilot
Update button clicked
Simulation file
Simulation data
View button clicked
Start button clicked
Simulation file
Update simulation
Sim parameters
JuliaSim render display
View simulation
Simulation logs
Simulation logs
Simulation data
Sim parameters
JuliaSim
Start simulation
Simulation file
Sim parameters
simulation logs
Simulation data
Begin simulation
Time elapsed
Create logs

6.2. State Model

(Top-level state chart here)

The substate chart for the Logging In state is shown below.



7. To Be Determined List

<< Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure. >>