

# Groundstation: Design

**Team #25**

**High-Altitude Rocketry Challenge**

Natasha Anisimova

Terrance Lee

Albert Morgan

## **Abstract**

The *Groundstation* software will collect telemetry from a rocket while it is in flight and graphically display the telemetry in real-time. Groundstation is made up several different components: collection of data, storage of data, interpolation of data, and display of data. This document will describe in detail the design of Groundstation.

## CONTENTS

<b>I</b>	<b>Introduction</b>	<b>3</b>
I-A	Identification . . . . .	3
I-B	Date of Issue . . . . .	3
I-C	Scope . . . . .	3
I-D	Stakeholders . . . . .	3
<b>II</b>	<b>Structure View</b>	<b>3</b>
<b>III</b>	<b>Interaction View</b>	<b>9</b>
<b>IV</b>	<b>Interface View</b>	<b>10</b>
<b>V</b>	<b>Glossary</b>	<b>14</b>
	<b>References</b>	<b>14</b>

## I. INTRODUCTION

### A. Identification

This document will provide an in-depth software design description of the Groundstation software package.

### B. Date of Issue

This document is issued on December 2, 2016. At this time, the design of the software is complete, but no work has begun on the implementation.

### C. Scope

This document will provide details about the Groundstation software package. The concerns of the software will begin when the telemetry is received via the serial port on the Raspberry Pi. Collecting the data from the rocket and converting it into a protocol readable through the serial port is the concern of the Oregon State University (OSU) American Institute of Aeronautics and Astronautics (AIAA) High-Altitude Rocketry Team avionics section.

### D. Stakeholders

The stakeholders of the Groundstation software package are members of the OSU AIAA High-Altitude Rocketry Team. The OSU AIAA High-altitude rocketry team is a group of multidisciplinary engineering students who are working together to build the rocket. The stakeholders require the ability to:

- View the telemetry in real time in order to track the altitude and aid in recovery
- View the data graphically so that it is easy to understand
- Log the data so that it may be analyzed at a later date

## II. STRUCTURE VIEW

### PC

Entity

*Author:* Your name here

*Type:* Component

*Purpose:* Why does it exist?

*Contents:* I'M MAKING CHANGES!

### Package Manager

Entity

*Author:* Albert Morgan

*Type:* Subprogram

*Purpose:* The package manager will install, track, and update software dependencies on the server.

*Contents:* Because Groundstation will be using Node and JavaScript for both the frontend and backend, Node Package Manager (NPM) will be used [1]. NPM has a large repository of both server-side and client-side JavaScript packages.

### Frontend

Entity

*Author:* Natasha Anisimova

*Type:* Component

*Purpose:* User Interface

*Contents:* The frontend of the web server will create and display the information that it was given by the backend. Given that the information has to do with location: longitude, latitude, and altitude, a 3D display is preferred. The frontend will consist of programs using 3.js and WebGL to create the visualization necessary to quickly understand the location of the rocket.

### Backend

Entity

*Author:* Albert Morgan

*Type:* Component

*Purpose:* The purpose of the backend is to facilitate communication between the rocket and the user.

*Contents:* The backend takes care of all data collection, transformation, logging, and serving that data to the user. A Node server will handle reading the data from the serial port, transforming the data into JSON, storing the data, and making the data available to the user.

## **Node**

Entity

*Author:* Albert Morgan

*Type:* Subprogram

*Purpose:* Node is the software that runs the backend.

*Contents:* The backend will run on Node. Several choices were considered to run the backend, particularly Node, PHP, and Ruby. The backend was chosen two criteria:

- Speed. The backend will run on a Raspberry Pi, so speed is important to minimize the system resources used.
- Interoperability. The backend needs to work with multiple components, including the logging software, the data coming in from the serial port, serving the frontend to the user.

Node uses an event driven architecture, which is ideal for reading data from the serial port. PHP and Ruby on Rails are both HTML preprocessors, so they don't get activated until the web site is requested. Node, on the other hand, runs continuously in the background and uses an event driven architecture. The event driven architecture is ideal for reading the data from the serial port. Additionally, Node much faster than either ruby or PHP.

## **Serialport**

Entity

*Author:* Your name here

*Type:* Library

*Purpose:* Node serialport library

*Contents:* Stuff here.

## **Log**

Entity

*Author:* Albert Morgan

*Type:* Data store

*Purpose:* Store the data in non-volatile storage for later retrieval and update new connecting clients

*Contents:* Groundstation will store telemetry in JSON format. The log should limit the possibility of the corruption of the data due to a programming error and make the data easily accessible for newly connecting clients. Several choices were considered for the logging, including relational databases, JSON documents, and logging the telemetry. A relational database would be unnecessary because the data does not have any relations that need to be tracked; each telemetry packet is an independent piece of information. Logging the telemetry as it comes in from the rocket would limit the possibility of corruption. A programming error in the JSON parsing routines could cause all of the data to be unusable. However, storing the data in JSON format would make it very easy for new clients that connect to the server in the middle of the rocket flight to be updated with all of the past data. For this reason, the telemetry will be stored in a JSON format.

## **jQuery**

Entity

*Author:* Your name here

*Type:* Library

*Purpose:* UI stuff

*Contents:* Queries the J

## **3.js**

Entity

*Author:* Natasha Anisimova

*Type:* Library

*Purpose:* 3.js is a JavaScript 3D library used to create and display animated 3D computer graphics using WebGL.

*Contents:* Since the information about the rocket will be displayed by using a web browser through a local Wi-Fi network, making sure the information is displayed on time and correctly is crucial. 3.js allows for easy and rapid development of WebGL applications, which means we can make the most of the specialized graphics hardware on the users' PCs that it gives access to.

**Rocket**

Entity

*Author:* Natasha Anisimova

*Type:* Component

*Purpose:* Physical manifestation of the object being tracked.

*Contents:* The rocket will have sensors attached to it that will transmit information about its altitude, GPS coordinates, tilt. While the rocket is on its journey the GPS signal will cut out due to US government laws meaning that the longitude and latitude will have to be calculated based upon the other information that is recieved.

**Web server**

Entity

*Author:* Albert Morgan

*Type:* Part

*Purpose:* The web server will serve three primary functions:

- Server web pages to the clients.
- Receive telemetry from the serial port and convert it into json.
- Make the JSON data available to the clients.

*Contents:* The web server needs to be stable so that there is no data loss during the flight. Additionally, the web server should be as lightweight as possible to minimize the resource consumption on the Raspberry Pi. Groundstation will use the Apache [2] web server. This web server was chosen over other options primary for it's stability. Apache uses a one-thread-per-connection model, so if one thread crashes, it will not affect the rest of the connections. NGINX [3] was also considered. On large-scale servers, it is much more lightweight because it uses a many-connections-per-thread model. However, the number of connections will be small and the reuse of threads could reduce stability. Lighttpd [4] was also considered. Although Lighttpd is much more lightweight than Apache, it is not nearly as mature. Because stability is a primary concern, Apache was selected.

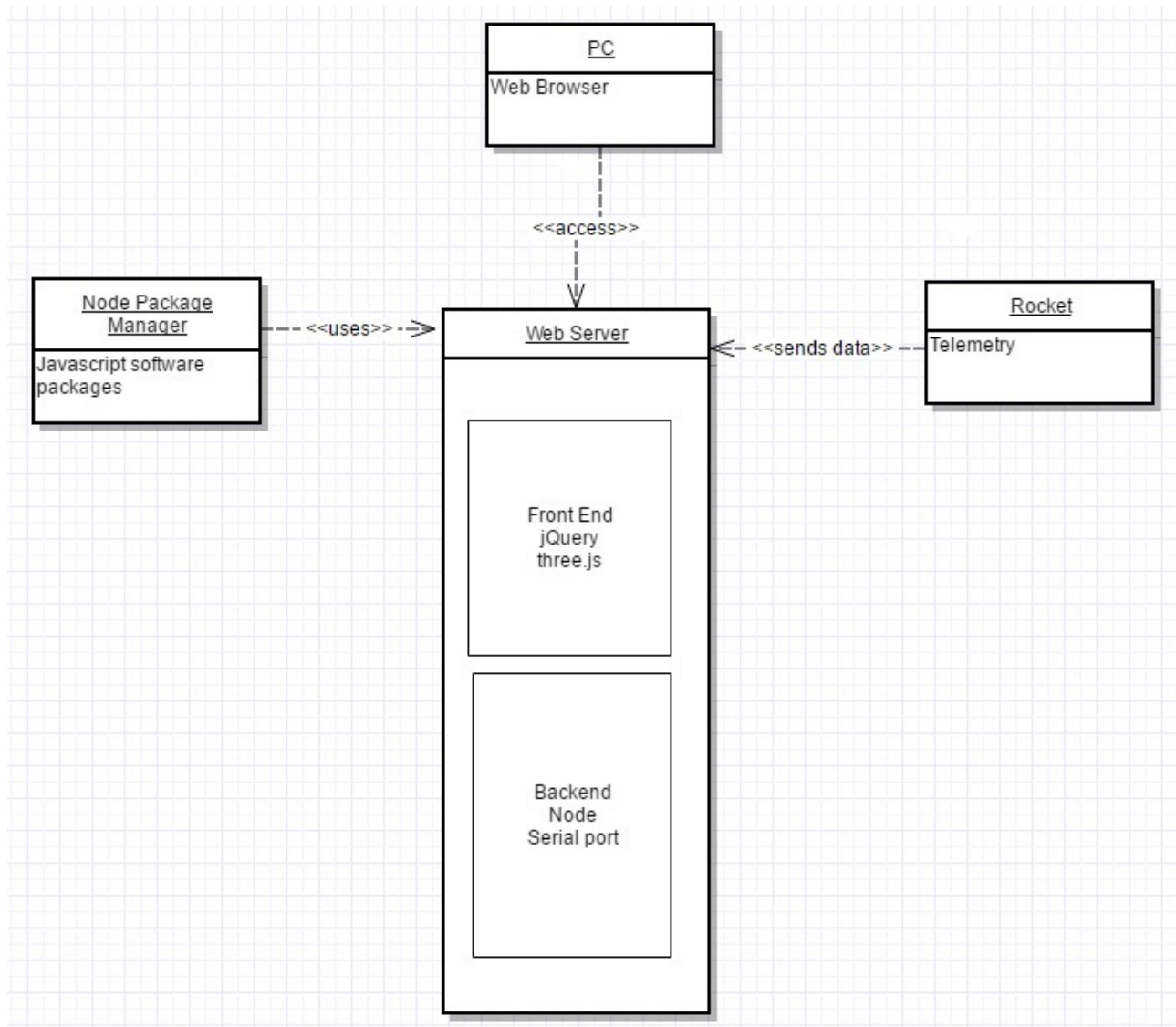


Fig. 1. A UML diagram of what the web server is and how it is connected to everything else.

**Web browser**

Entity

*Author:* Albert Morgan

*Type:* Process

*Purpose:* The web server

*Contents:* The client will use a web browser to connect to the Groundstation web server and access the content. The web browser may be any of:

- Chrome version 54 or higher
- Edge version 14 or higher
- Firefox version 49 or higher
- Safari version 10 or higher

002

**Web browser / Web server**

Relationship

*Author:* Albert Morgan

*Type:* Connected

*Contents:* The web browser connects to the web server over a WiFi network via the HTTP protocol.

**Frontend / Backend**

Relationship

*Author:* Natasha Anisimova

*Type:* Connected

*Contents:* The Frontend will take what the Backend of the software gives it and use 3.js to create and display information to the user.

**jQuery composition**

Relationship

*Author:* Your name here

*Type:* Composition

*Contents:* Stuff

**3.js**

Relationship

*Author:* Natasha Anisimova

*Type:* Part of

*Contents:* The JavaScript Library 3.js will be used with WebGL to create the frontend of Groundstation. Figure 2 goes through how 3.js is a part of the creation of the frontend

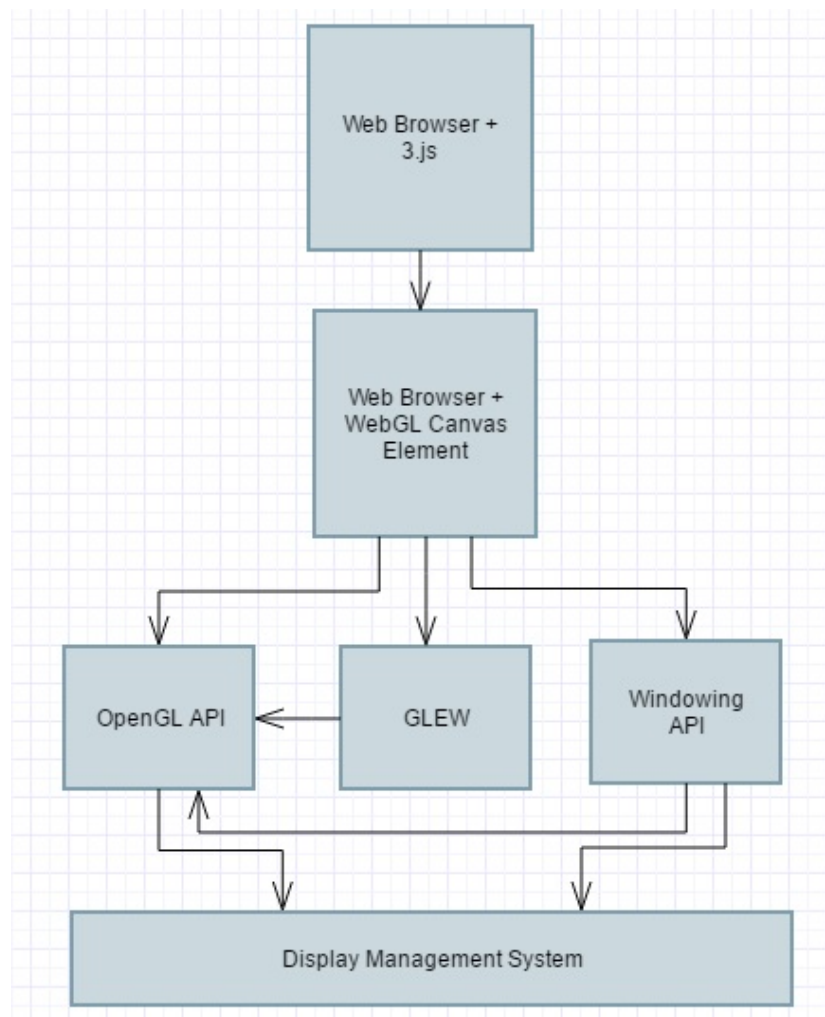


Fig. 2. A diagram of how 3.js is connected to WebGL and the frontend.



**Node / Backend**

Relationship

*Author:* Albert Morgan*Type:* Part of*Contents:* Node is used by the backend to do the majority of the work. Node is responsible for receiving, transforming, logging, and sending the telemetry to the user.**Serialport use**

Relationship

*Author:* Your name here*Type:* Use*Contents:* Stuff**Log / Backend**

Relationship

*Author:* Albert Morgan*Type:* Part of*Contents:* The log is part of the backend, and will be written to by and exist on the same device that runs the backend.**Backend / Rocket**

Relationship

*Author:* Albert Morgan*Type:* Connected*Contents:* The back end will retrieve data from the rocket through the serial port. The exact protocol used will be determined at a later date by the OSU AIAA High-Altitude Rocketry avionics team.**NPM / Frontend**

Relationship

*Author:* Albert Morgan*Type:* Connected*Contents:* NPM will be used to install all frontend software dependencies.**NPM / Backend**

Relationship

*Author:* Albert Morgan*Type:* Part of*Contents:* NPM is part of the backend, and will be used to update and install all backend software dependencies.**III. INTERACTION VIEW****Rocket**

Entity

See *Structure View - Rocket***Web Server**

Entity

See *Structure View - Web Server***Web Browser**

Entity

See *Structure View - Web Browser***Telemetry Protocol**

Relationship

*Author:* Albert Morgan

*Type:* Signal

*Contents:* Once every second, the rocket will send a telemetry packet to the Groundstation. This packet will be received and transformed into a protocol that will be transmitted to the Groundstation software package via the serial port. The specification of this protocol will be determined by the avionics team at a later date.

## JSON Protocol

Relationship

*Author:* Albert Morgan

*Type:* Signal

*Contents:* Data will be stored in the log file and transmitted to the user as a list of JSON objects in the following format:

```
{
  sensor: ...,
  value: ...,
  timestamp: ...
}
```

The JSON object has the following attributes:

- **Sensor:** the sensor that the data is recorded from. This is an arbitrary string that uniquely identifies a specific sensor on the rocket. This string may be anything, and the user interface must understand which sensors provide which data. The specification of which strings correspond to which sensors will be done at a later date, after the protocol is defined.
- **Value:** the value that has been read off the sensor. The type of this data will depend on the specific sensor.
- **Timestamp:** The time and date that the data was received. This value is an integer representing the number of milliseconds that have elapsed since the epoch (January 1, 1970).

JSON objects will be transmitted to the client as a single object with a list of smaller objects inside. For example:

```
{
  {
    sensor: altimeter,
    value: 900,
    timestamp: 00584747458945
  },
  {
    sensor: tilt,
    value: 4,
    timestamp: 00584747458945
  }
}
```

## IV. INTERFACE VIEW

### 3D View

Entity

*Author:* Natasha Anisimova

*Function:* Displays the current location of the rocket.

*Interface:* The user can interact with the view by using a mouse or track pad.

*Contents:* The view will be a 3D environment that will resize as the user resizes the window of the web browser. If the user clicks and drags in a particular direction the 3D environment will move in that direction. If the user just moves the mouse without clicking, the mouse will move freely on the screen. To make orientation easier in this environment there will be a set of buttons, shown in the figure below, that will have preset views readily available for the user. The top most button will show a bird's eye view. The left button will show a three-fourths view facing the left side of the launch site. The right will do the same except facing the right side of the launch site. Last but not least, the bottom button in this set will give a view that is directly aligned with the ground, or X plain, looking out into the horizon. With the help of these buttons the user should never feel like they are unable to get back to looking at the journey of the rocket. The scroll button will zoom in and out, with the point of origin being the launch location of the rocket. Figure 3 shows camera view controls that will be located in the top right corner of the browser.



Fig. 3. A diagram of the camera view controls.

### 3D Environment

#### Entity

*Author:* Natasha Anisimova

*Function:* Displays the current location of the rocket in the launch environment.

*Interface:* The user can interact with the view by using a mouse or track pad.

*Contents:* As for the 3D environment itself, that will consist of the topological map of the launch site, Spaceport America in New Mexico, with satellite images of the area mapped to it from NASA. If the user zooms past the point that the topological map covers it will just be a flat gray plane that goes on forever in every which way the ground would be. Since the rocket's goal is to reach 100,000 kilometers in height, labeling the different levels of the atmosphere that it enters would be helpful. Each level can be a transparent shade of blue, going from light to dark. Figure 4 shows the layers of the Earth's atmosphere.

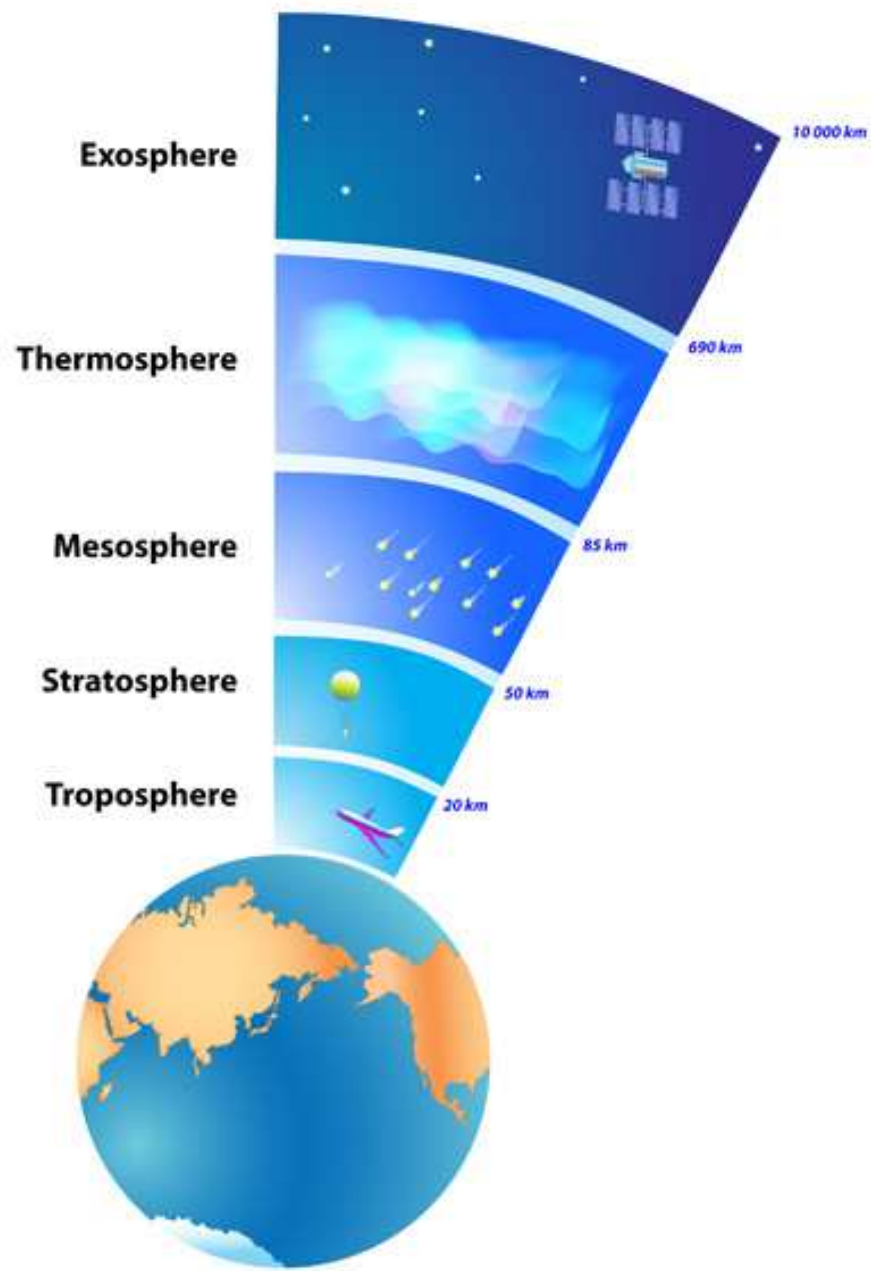


Fig. 4. A diagram of the Earth's atmosphere.

### Numerical Data

Entity

*Author:* Natasha Anisimova

*Function:* Shows numerical data for the location of the rocket.

*Interface:* The user can view or hide the table.

*Contents:* Visual data can be hard to interpret without any numerical data to put it into perspective. Due to this, a table with the altitude, longitude, and latitude will be a part of the user interface. This table will be updated every time that a telemetry packet is received and given to the frontend of the web server. Figure 5 shows an example of how the table could be visible or hidden based off of what the user wanted.

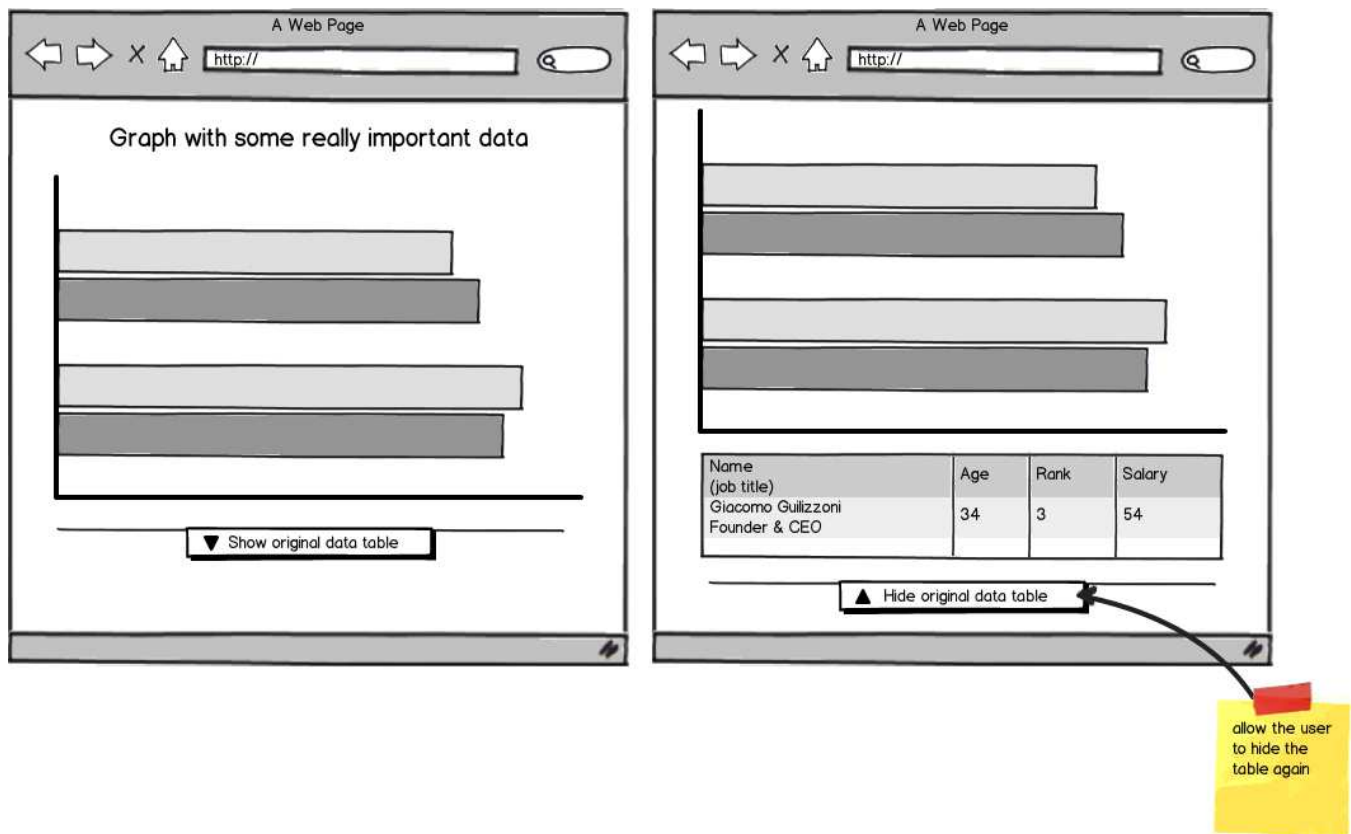


Fig. 5. A diagram of how to show or hide a table.

## V. GLOSSARY

- **Accuracy:** The absence of errors in the telemetry GS receives, logs, and displays.
- **Binary data:** Telemetry that represents one of two states, for example, “stage 2 activated” and “stage 2 not activated.”
- **COCOM limits:** The Coordinating Committee for Multilateral Export Controls (COCOM) limits apply hard limits to the speed and altitude of a GPS device. If a device exceeds these limits, it will stop providing GPS data. This is done to ensure that the device cannot be used as a missile.
- **Corruption:** The process by which data is altered or made unreadable.
- **Crash:** A software crash; the event in which a piece of software ceases operation unexpectedly.
- **Die:** A process dies when it ceases operation and is removed by the operating system. The difference between dying and crashing is that a crash may cause the program to become non-responsive, whereas dying causes the process to end.
- **GS:** Groundstation, the name of our software.
- **Graphical display:** Data that is displayed using a visualization.
- **Live:** Updated in real-time.
- **Non-volatile storage:** Storage that will not be erased when the system is powered down. For example, a hard drive or flash storage.
- **Page:** A web page that users of GS may connect to in order to view the telemetry.
- **Process:** A running program on a computer.
- **Raspberry Pi:** A small, inexpensive computing platform.
- **Real-time:** Each telemetry datum received from the rocket must be processed and displayed in under one second.
- **Redundant sensors:** Two or more sensors that provide the same type of data.
- **Reliability:** In the event of a software crash, the Groundstation software should automatically start and begin all normal functions in under five seconds.
- **Robustness:** In the event that GS receives data that is garbled or otherwise does not adhere to the protocol, it must continue to receive and display data and not break the real-time requirement.
- **Storage:** A device where data is logged.
- **Telemetry:** Data received from the rocket while the rocket is in flight.
- **Telemetry packet:** The rocket will send a telemetry update once per second. Each one of these updates is a “telemetry packet.”
- **Visualization:** Information or data, transformed into an visual context.
- **X, Y, Z coordinates:** A three coordinate system that pinpoints a point in 3D space.

## REFERENCES

- [1] Node package manager. [Online]. Available: <https://www.npmjs.com>
- [2] Apache. [Online]. Available: <https://www.apache.org>
- [3] Nginx. [Online]. Available: <https://nginx.com>
- [4] Lighttpd. [Online]. Available: <https://www.lighttpd.net/>

---

Nancy Squires

---

Date

---

Natasha Anisimova

---

Date

---

Terrance Lee

---

Date

---

Albert Morgan

---

Date