University of Alabama in Huntsville
LOUIS

Honors Capstone Projects and Theses | Honors College

4-19-2023

Engineering Friendlier Products with Iterative User Testing

Jack Richard Stookbury

Follow this and additional works at: https://louis.uah.edu/honors-capstones

Recommended Citation
Stookbury, Jack Richard, "Engineering Friendlier Products with Iterative User Testing" (2023). *Honors Capstone Projects and Theses*. 837.
https://louis.uah.edu/honors-capstones/837

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.

Engineering Friendlier Products with Iterative User Testing

by

James Richard Stooksbury

An Honors Capstone

submitted in partial fulfillment of the requirements.

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 19, 2023

Honors Capstone Director: Dr. Earl Wells, P.E.

Professor of Electrical and Computer Engineering Department

James R. Stooksbury April 19, 2023

Student Date

05/03/2023

Director Date

05/03/2023

Department Chair Date

William Wilkerson Digitally signed by William Wilkerson
Date: 2023.05.04 08:31:11 -05'00'

Honors College Dean Date

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

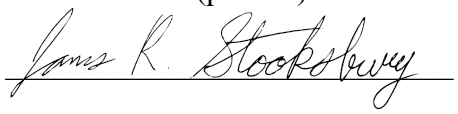
Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

_James Richard Stooksbury_____

Student Name (printed)



Student Signature

_April 19, 2023___

Date

Table of Contents

Abstract

In our modern world, software is everywhere and used by almost everyone every single day. But what does it take to engineer software that is friendly and easy to use? This project explores the product design styles and rationales behind designing good software. These design rationales are then evaluated and tested in an iterative manner to help incorporate user feedback into creating better products. The goal in mind being to apply these concepts and testing principles to help improve upon the final product of the software and product being developed.

Introduction

In our modern era, most people will find themselves interacting with multiple computers every single day throughout our lives. Desktops and laptops are what people traditionally think of when discussing computers, however, we tend to interact with far more than just those. Everything from smart light switches, car radios, smart phones, key fobs, and much more are all devices that we interact with every single day that can fall into this category. A large focus of these computer-based products that we help create as Computer Engineers is their design. Design is important because that is how our end users will interact with the products that we create.

Throughout my schooling, I have worked as support in the Information Technology field. This led me to notice that one aspect often overlooked is how easy it is for a user to be able to learn new programs. As designers of these programs and products, it's easy for us to view things as just being "obvious" when operating them. However, this is far from the truth when looked upon from an outside perspective. While designing these products, we need to put ourselves in the minds of end users. In my career work, I was the end user helping deploy these products to users instead of the one who created these products. Often times these products were not sold as being for anyone not of a technical mindset, however, they still suffered from usability problems. Some of these issues ranged from vague documentation to difficulty navigating them. Some products I've worked with had issues like leaving out entire steps needed to be done to setup up the product properly in their documentation. While, these steps may have been intuitive for the person creating the documentation, they were not for me. When researching solutions, I found that I often wasn't alone in this regard. It is very important that we try to make our products as easy to

use, when possible, to avoid having to bury our end users in documentation. This is in addition to making our documentation very thorough so that our end users can properly refer to it when necessary to learn and find solutions.

An important aspect of the field of Industrial Systems Engineering is the "people" aspect of operating things. This means that there is a great value placed upon efficiency and usability of products in this discipline. This is a topic that isn't talked about much from my experience with Computer Engineering classes aside from the basic properties of GUI interfaces. In my experience working in the Information Technology field, I found myself having to work with people with very different skillsets regarding technology. It is very easy to greatly overestimate the capabilities of the average user when we find ourselves surrounded by primarily other engineers and people from over technical backgrounds. Even in younger generations that are much more familiar with technology, most people still don't have many technical skills developed for interacting with it in non-basic ways. This means that ease of use should still be a very high priority when we design our products. These experiences highlighted to me the importance of good design and led me to strive to learn about and incorporate these principles in my engineering career.

Background of Project

For my group's senior design project, a principal aim was to create a product that is easy for the average person to use. The product we have worked on creating is a weather information and forecast device that can be used in situations where the user may be off the grid or in a situation where the traditional means of communication may not work due to something like natural disaster situations. The product we've created uses a Raspberry Pi, an SDR, and an antenna to download weather information and radar data from National Atmospheric and Oceanographic Administration's GEOS 15 and 16 weather satellites. The question was how do make this data accessible in a manner that the average person could pull off.

While we are not the first people to achieve this task of downloading GEOS weather satellite data from homemade setups, many of the existing methods and documentation was this was either difficult to complete and/or vague. This meant that we had to come up with creative solutions to making this data easily available and gatherable. The solution we settled upon was that the Raspberry Pi would broadcast a Wi-Fi network that the end user could join when plugged in that would display a website that would be able make these functions easily usable and available.

In our project, the webserver is our GUI interface. This interface will help translate all of these complicated commands into something that an average user would be able to do. In addition to this, the interface provides an easy way to view the results of the data downloaded from the result of these commands. The design of this interface is critical because it makes the difference between whether or not the user is able to effectively operate this software at

all. It is these reasons that I decided that making the user interface and product itself easy to use was of great importance to research.

The purpose of this project is to explore ways to implement user testing into designing products. This user testing is used to help improve and iterate upon the design of the product. In addition to this, this project is exploring the background between different methodologies used to achieve this task. Finally, software design approaches are researched and explored to help improve upon the designing principles of the product so that it can be as user friendly and intuitive as possible to begin with. This is especially important to the software design process because it more expensive to change things the later it is in the development software development process. All of this means that it is important to discuss and explore design principles and processes throughout the entire development process. This project functions as a "value-added" component to my senior design group's product overall and this additional work was conducted solely by myself.

Considerations to Software Design

One of the biggest issues that developers will face when designing software is striking the balance between simplicity and complexity. You want to be able to make it where the average user is able to operate your software without much difficulty while also having it not obstructing functionality from users that they may need. For the product I'm trying to make and improve upon, the process may need to be troubleshooted due to the nature of focusing the antenna to get the satellite signal. This may involve advanced functionality that the interface must be able to account for. It is important to try to strike the balance between making the interface straightforward and feature rich while also trying to make be able to accommodate advanced functionality. For this particular project, the issue was much more on making it straightforward to use due to the difficulty in making some of these complex programs and commands run seamlessly in the background.

Another important consideration that must be made when working on software design is accommodating the skill levels that the end users will have in regard to using computers. For the nature of this project, the goal is to make it easy for someone with average to slightly above average computer skills to be able to use it. When designing for average users, they will most likely need guidance and for things to be straightforward as possible. Conversely, advanced users are more likely to frustrated if they cannot configure things in more advanced ways when needed.

Staying up to date with changing technology is also another considerable challenge when designing software. When designing the software for this project, applying modern design principles was important. This helps the software feel more familiar and usable to the average end user. It also meant that there was great focus on making sure this software was

compatible with modern, secure, and up to date systems. Luckily, this task was not that daunting due to the fact we're running the software that the user sees on a web server. This meant that designing the software to work across almost all modern platforms was relatively simple. This is due to all common browsers adhering to a mostly uniform set of standards. This speed up development considerably in comparison to developing a separate application for each major OS platform. Using a web browser-based approach also helps keep the software functional with less long-term maintenance due to the nature of web browsers in comparison to directly compiled system software.

Another important consideration that is a principal component of this project is user feedback. While this is often overlooked by many while in active development, if you want truly successful software, this is a goal must be achieved. It is also one that isn't minute in size. For the goals in mind, this is an area of immense importance. This goal was not also completely straightforward to implement.

Software Design Languages

The web landscape is more of the most rapidly evolving and changing areas of software design. It could be argued that the web has caused some of the most rapid advances and changes in user design. Throughout the years, various design "languages" have emerged in software design. These design languages have evolved with the ever-changing technology landscapes. All of these design languages have their own advantages and drawbacks that must be considered. However, these various "languages" have all had their own periods of relevance in software design. These different designs are more easily seen historically in different versions of operating systems due to the constantly changing and nonpermanent state of websites. It is important that these designs languages are considered when developing software.

Having a unified design language is an important from a usability standpoint because it makes easier for the user to operate the software. For example, having buttons consistently look the same means that the user will always be able to identify that interface element as a button. Having the user be able to easily identify something means that its must less likely that they'll suffer from frustration and confusion when navigating your software. This is an issue that certain design languages like "flat" design are known to suffer from. This also means that once a design language is chosen, your user is more likely to expect to see this language everywhere. Another example of this problem occurring in real life is with Microsoft Windows. Certain components of Microsoft Windows have been updated to different languages and styles over the years while other components haven't. This makes it harder for users to identify and navigate the operating system. This is also an issue that is

more likely to occur as software and matures overtime throughout its upgrade and maintenance phases.

Some of the most well-known software design languages are skeuomorphism, flat-design, and the more recently developed one of neomorphism. Skeuomorphism is one of the earliest design languages that emerged in the early days of software becoming mainstream. It is focused heavily on making design elements mimic their physical real-world counterparts. This was important in the early days of software design due to most users having limited to no experience using software previously. This design is largely considered "ugly" and "outdated" by most nowadays. One of the best-known examples of what could be considered "peak" skeuomorphic design was Mac OS X Snow Leopard and iOS 6. Another reason it fell out of favor is due to its higher resource intensity and the difficulty involved in properly implementing it across various devices. Earlier implementations of this project tended to mimic this design language and was shown to users in the first stage of testing.

The most commonly used design language nowadays is flat design. This design most famously implemented in the large design overhauls of Windows 8 and iOS 7. It is based on making things more abstract with flat colors and text. Even when images are used, the goal is to make them simple and pictographic in nature. This is in sharp contrast the earlier designs of skeuomorphism where gradients, shadows, and reflections were common. This design exploded into rapid popularity for website design due to it being easy to implement across different screen types and it being very light resource wise. The biggest drawback to this design approach is usability. Due to the extremely simplistic nature of elements, it is very easy to mix up interface elements in addition to it being harder to identify interface

elements to begin with. This design language often will unintentionally hide things from the user if it isn't obvious that it's there.

A newer approach that has emerged within the last couple years is called Neumorphism. Neumorphism is an attempt to help bridge the best things about flat design and Skeuomorphism together. It uses the simplistic nature of colors and shapes from the flat design while combining the elements of shadows, minor textures, and reflections found in Skeuomorphism. The result is a modern looking design language that offers the familiarity of modern software design while making the interface much more friendly to interact and identify with. One of the best examples of this approach being implemented today is modern versions of macOS from Big Sur to the current version Ventura. A variation of this design called Glassmorphism attempts to apply this philosophy in a manner that can be displayed upon more complex images and backgrounds rather than just solid colors. It looks best for applications with background images. For this project, the final product is using the Glassmorphism design language. It seemed like the best fit when trying to apply modern design principles that still adhere to usability. It also appealed most to personal preferences. Examples of these design styles can be seen in Figure 1.

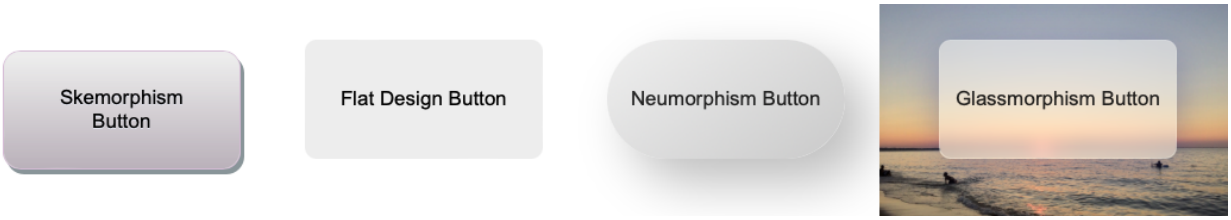


Figure 1: Simple example of the design languages discussed implementing a button. Background image on Glassmorphism button is there to illustrate it better.

User Testing Methodologies

For helping improve products with user testing there are several different methodologies that one can use to accomplish this task. These strategies all have their pros and cons as with almost anything in life. When developing software is good to try to have plan your testing methodologies along with the development cycle. After all, it's well documented and preached within development frameworks like Agile that changing things later on in the development cycle will almost always cost more than changes that occur earlier on in development. This means that choosing the right strategy can make a measurable impact on success.

There are many different testing methodologies that have arisen in prominence. Some commonly used methodologies are A/B testing, Think-Aloud Protocol, Contextual Inquiry, Heuristic Evaluation, Card Sorting, Surveys & Questionnaires, Eye-tracking, and Usability Benchmarking among many others. A/B testing involves setting up user tests involving two different versions of user design. The users are asked which version they find more effective and friendly to used or it may involve testing users about their experiences while giving each one of them access to only one of the user designs. The Think-Aloud Protocol involves having users analyze their experiences by saying their thoughts aloud while completing their tasks in the presence of an observer. This is useful as it can help research put themselves in the minds of their users. Contextual Inquiry is similar to the Think-Aloud Protocol as it also involves watching users as they attempt to use the software, but it doesn't involve asking them to speak their thoughts aloud.

Heuristic Evaluation involves having experts evaluate products against established principles and rules. These test help identify problems that professionals may see in the software. Card sorting involves asking users to sort cards with various features into groups that based on their preferences. This can be useful if you are unsure about what features should be prioritized in your product. Surveys and Questionnaires focus on asking users for feedback after they have used and experienced with your software. Eye-tracking is a more advanced method of testing that focuses on using specialized software to track where users are looking at an interface. This typically can provide insight into issues related to information visibility and hierarchy. Finally, Usability Benchmarking involves comparing the usability of the software design to industry standards or benchmarks. All of these testing methods have their benefits and drawbacks.

After researching these testing methods, the task was to select the most appropriate way to test the software of this project. For this project, multiple testing methodologies were implemented. User testing involved a combined of Contextual Inquiry, Surveys & Questionnaires, and Heuristic Evaluation. For Heuristic Evaluation, the testing was done with fellow peers as the interface was being developed. This was conducted on an informal basis with me asking peers in my vicinity about how they felt about the way certain things should look, feel, and be laid out. This technically is a variation of this method since it didn't involve outside experts, however, simply talking among your peers as its being developed in this manner can be very beneficial to help squash out issues before they develop to begin with. This was iterative in the manner that all of the suggestions and improvements were implemented and built upon one other. These layouts changes and testing can be seen in the commitment history of the GitHub repository for this project.

For the outside user testing, as mentioned before, it was done with the Contextual Inquiry and Surveys & Questionnaire methodologies. The Contextual Inquiry methodology was setup to follow and watch the test users as they attempted to use the product. This involving having them setup the physical hardware of the product and attempting to connect to & use the software of the product. The primary focus on the software though. The users were given instructions on how to assemble and use the software during the test. The users were also given guidance on using the product as need throughout. Issues were noted as they were seen in real-time. For the final portion of the evaluation the users were given a questionnaire. The users were asked to answer the following questions on a scale of 1-10 with higher numbers indicating positive praise: "How do you rank the interface of this product?", "How do you rank the usability of this product?", "How do you rank the ease of assembly?", and "How do you rank quality of this product?". These questions where followed by an open-ended essay response question asking them, "What are some issues you saw or improvements you think need to be made? You may also provide feedback in a general sense.".

For the test group, it was composed of five users. Due to resource restraints, these five test members were all students at the University of Alabama in Huntsville. Efforts were made to try to get people from a range of technical background. These tests involved students apart of the College of Business, College of Engineering, College of Science, and College of Nursing. The participants ranged in age from 19-22. Participants were provided with lunch for their participation. For this project, three different test sessions occurred. They were conducted on March 5, April 2, and April 9. Changes were made based on the results of all three of these sessions and were conducted based upon the state of senior design project as a whole at the time.

Testing Results

First Trial

For the first test, user feedback wasn't particularly good. This is due in part to the current state of the project at the time of testing. At the time of testing, much of the website functionality wasn't functional. This test mainly served as a way to receive feedback on the interface itself. At this moment in time, due to delays in the project, the project was still in its design phase infancies. This testing phase was still useful though because it allowed feedback on the initial designs of the interface. One thing that was noted in the Contextual Inquiry phase was that users struggled to know where to look around on the interface. This was a failure on my end due to my limited explanations of functionality given at this point. Users were also confused about the presence of partially implemented features.

At this point in the product development cycle, the product design style was largely done in an older Skeuomorphic like style with heavy emphasis on borders, transparency, and color. This style presented at this point was not intended as being a final product design and focused on simply getting functionality implemented at that point in time. The major take aways from this trial phase was that the style needed to be changed and pages needed to be clarified. Examples of the interface from this phase can be seen in Figures 2 and 3. Collected subject table is shown in Table 1. Note, the headings of the table are abbreviations of the questions defined earlier.

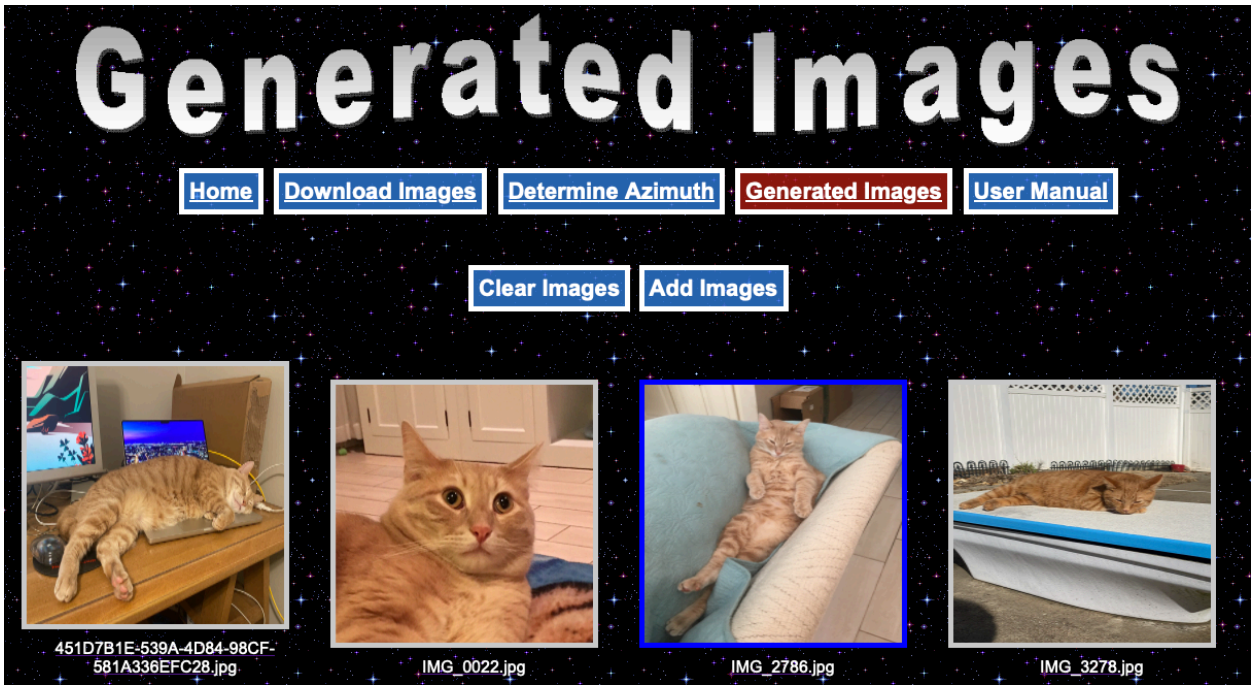


Figure 2: Example of very early application styling. Example images were placeholders for the final functionality.

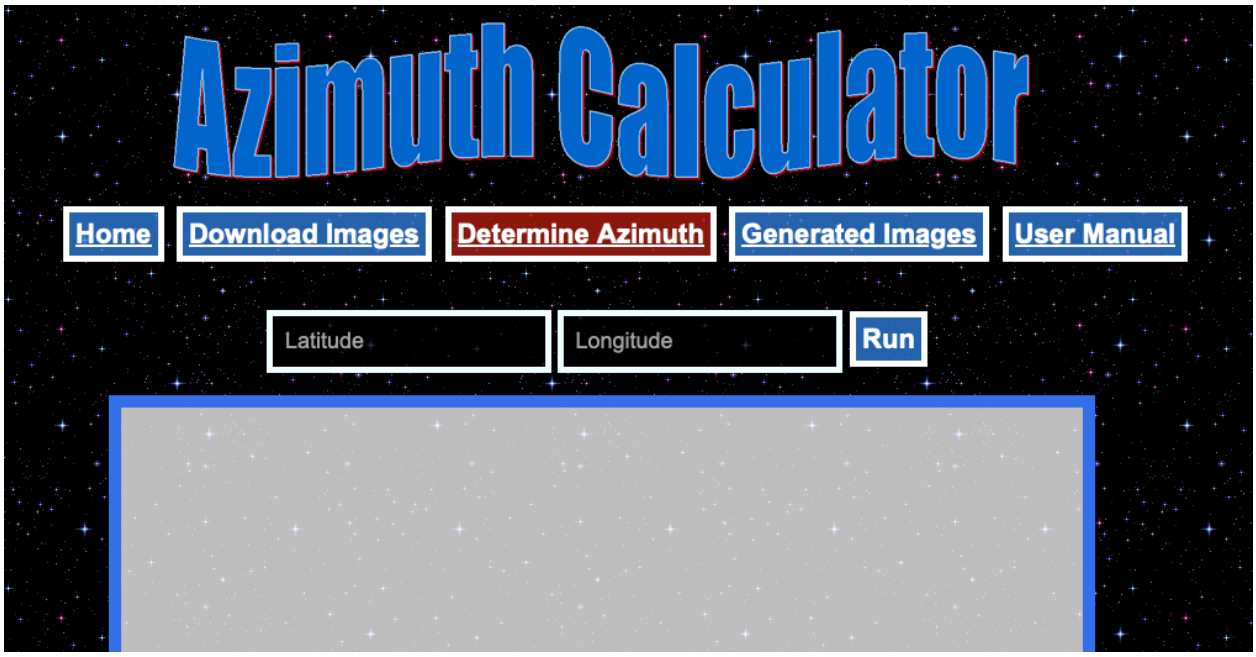


Figure 3: Additional example of very early application styling.

Table 1: Survey Results from First Trial of Testing

Test #1: March 5, 2023					
Subject	**Interface**	**Usability**	**Assembly**	**Quality**	**Feedback**
#1	4	2	3	3	The design of the software is confusing. It's unclear what the objective is.
#2	3	3	2	4	The interface does not look good. It looks like it was designed in 2000s. Somethings don't make sense.
#3	5	4	5	3	You need to put descriptions on pages better. I don't understand what a lot of this is for.
#4	4	2	2	2	A better manual is needed. I understand that much functionality hasn't been implemented yet though so this may change in the future.
#5	6	3	4	2	The space themed background is nice. Otherwise, the interface looks dated.
Overall	4.4	2.8	3.2	2.8	-

Second Trial

For the second phase of testing, the entire user interface was redesigned from scratch. After additional research, I decided to redesign the interface using the stylistic principles of Glassmorphism. Users noted that the interface felt much more "modern" to use and looked more professional in their opinions. Product functionality overall at this point in time was still rather limited. Additional instruction was given to the users this time around to help avoid confusion on setting the product up. The user manual page was updated to include some basic information about the commands needed to run the satellite data downloader.

While the functionality of data downloading was available at this time, it still rather primitive in nature and considered "burdensome". This in part due to this functionality not being available on the webserver itself. Instead, users were instructed in connected to the Raspberry Pi to run the software directly via SSH. The users struggle to operate it was very visible when observing it. The goal for the third trial was to improve that section of functionality as much as I could within the current working state of the larger project as a whole. Examples of the updated UI for this phase can be seen in Figures 4 and 5 with the user feedback from this phase available in Table 2.

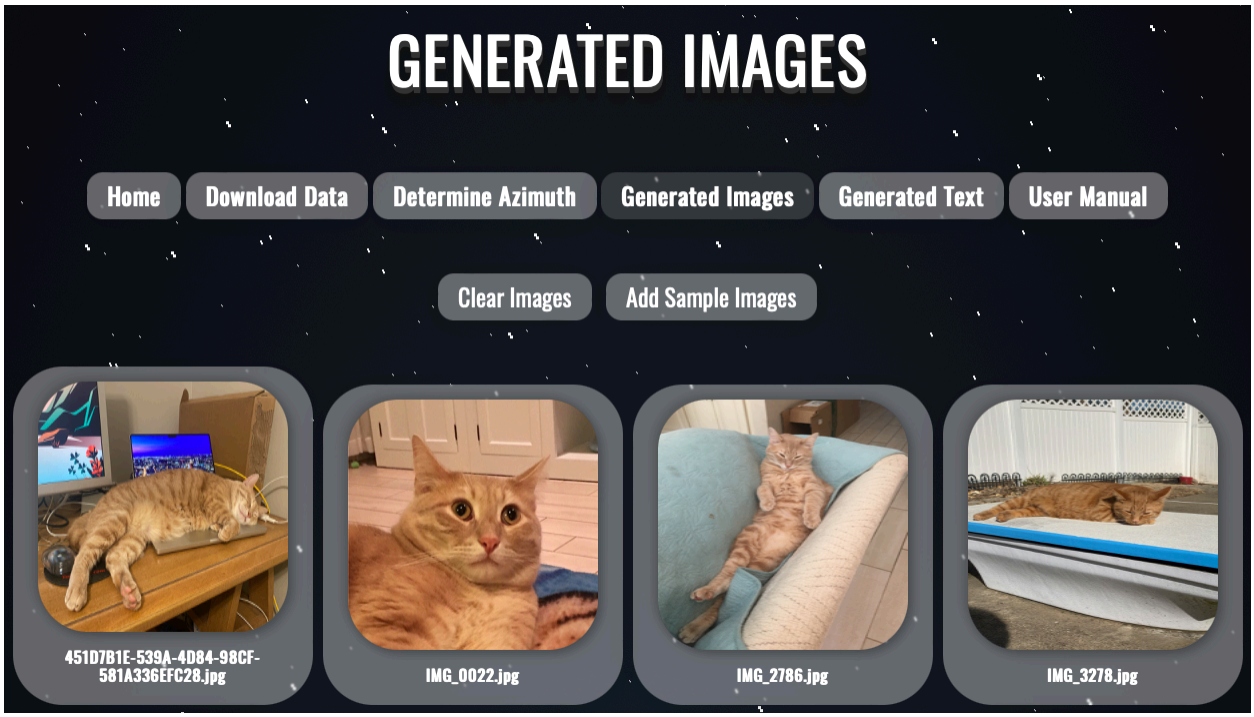


Figure 4: Example of updated design language that ended up in the final product. Cat images are still used as placeholders for radar images.

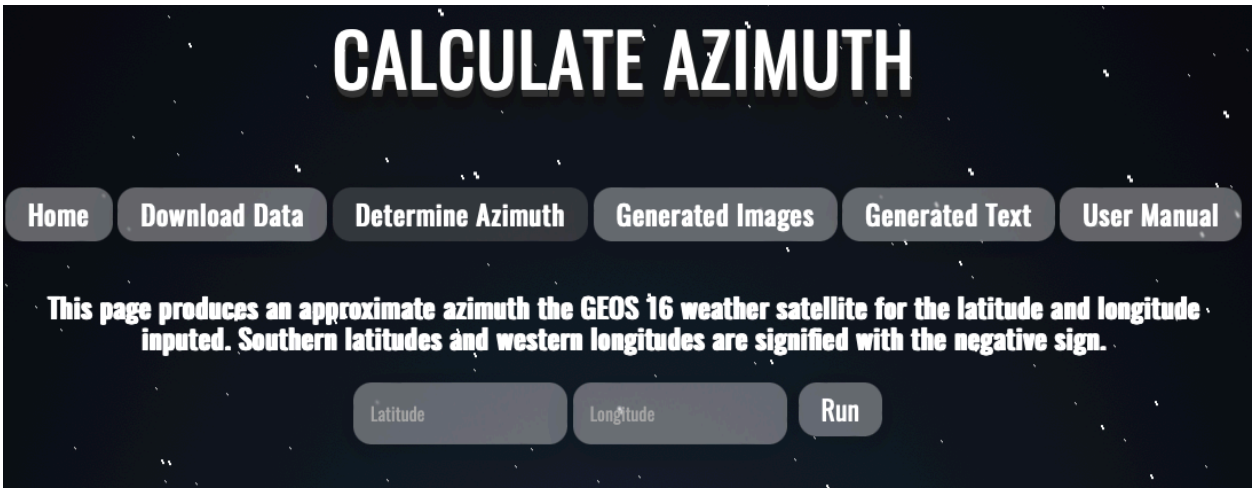


Figure 5: Example of new interface implemented on Azimuth Calculation page with explanation included to better help the user understand it.

Table 2: Survey Results from Second Trial of Testing

Test #2: April 2, 2023					
Subject	**Interface**	**Usability**	**Assembly**	**Quality**	**Feedback**
#1	6	6	6	5	The user manual page needs work. It needs to explain the functions of what everything does better.
#2	7	5	5	6	I didn't really understand the cat images on the Generated Images page. It would be more useful to have that show actual radar images as you later explained as the current placeholder since you said you don't have that working right now.
#3	6	4	7	5	I found it hard to use the product to get the data. These commands while provided, where confusing to use.
#4	6	4	6	7	Getting the data was confusing even with your help. I don't use computers like you do and don't understand the text command part at all.
#5	8	5	8	7	The interface looks much more modern and nicer looking. Downloading data could be done better. Most people don't know how to use terminals to SSH into computers.
Overall	6.6	4.8	6.4	6	-

Third Trial

For the third trial, results were improved upon from the second phase. This phase took place after a much shorter time period due to the end of the semester approaching. The main improvement in this phase of the trial had to do with the method of downloading data for the user. While one-click command buttons were still not implemented due to project stalling on the senior design group end as a whole, an interface was implemented to allow these commands to run directly within the browser window. Not having those buttons implemented was the biggest testing flaw of this phase. This interface called "Shell-in-a-box" allowed the users to run these commands within the page itself. This was considered to be a very nice user-friendly improvement. Assembly and Quality was considered to be of the same quality from the second trial by the users which was expected due to minimal changes being implemented in those categories between these two trials.

For the setup of the satellite dish, the calculations were done with the software called GPredict. This portion of the process was directly done with help by me. The original goal was for me to have to do that in the future by having the finished software help the user do that. The program GPredict, helped determine the appropriate azimuth and elevation angle needed to point the satellite. However, as with the second phase of the project, users still noted and struggled with running these commands. This confirmed a great fear of mine due to the vast majority of users having little to no command-line experience. This was the limiting factor of the progress of the larger project as a whole which unfortunately limited the improvements able to be made from this value-added product testing. One of the improvements that was implemented from this phase was that an additional button to allow the multiple terminal windows to be opened by the user at one time to make running

commands easier. The interface of the Download Data page can be seen in Figure 6 along with the results from the third trial of testing in Table 3.

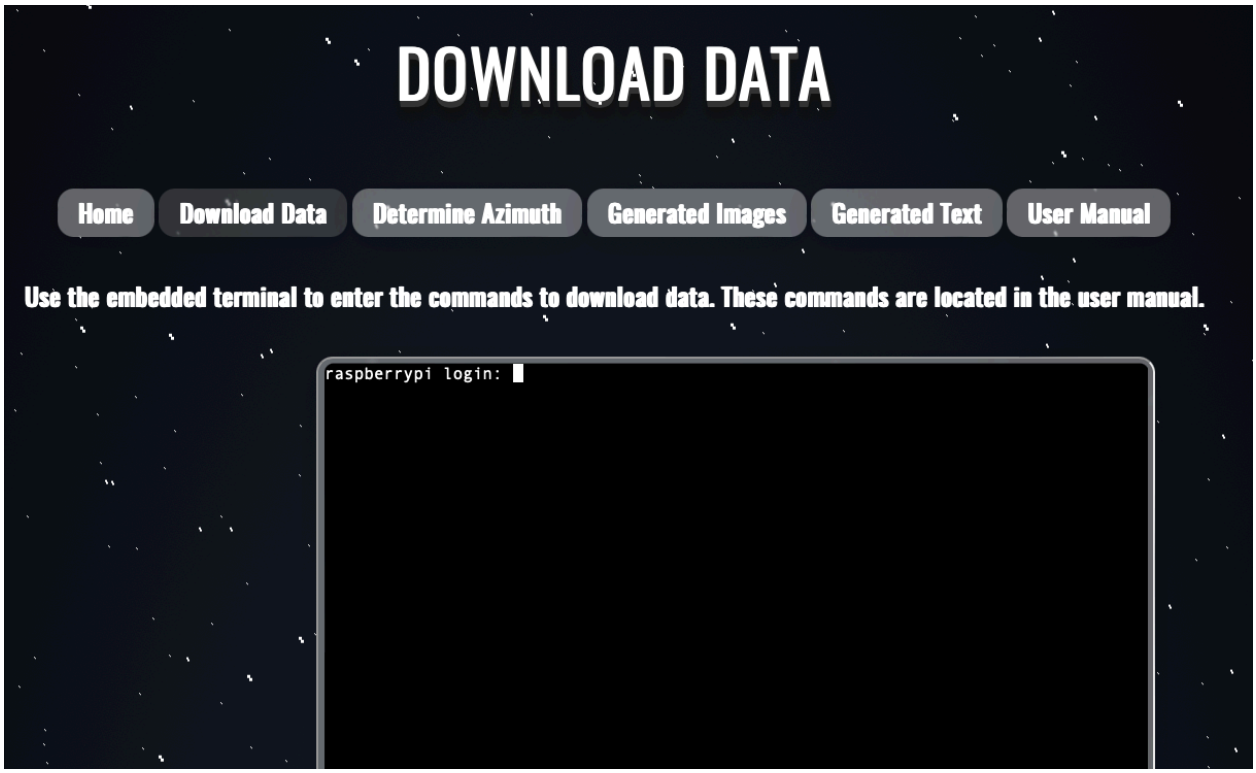


Figure 6: New interface implemented for downloading data on the Download Data page of the web server.

Table 3: Survey Results from Third Trial of Testing

Test #3: April 9, 2023					
Subject	**Interface**	**Usability**	**Assembly**	**Quality**	**Feedback**
#1	8	7	6	5	The user manual page is better than it was before. Using the software to actually get the data still isn't straightforward but it is improved from before.
#2	7	6	5	6	I liked how you changed the example images of the generated image tab to be accurate radar images. This made it easier to understand what that page was for since you don't have it working completely. I liked the download page for getting the data more than opening the SSH as you had us do last time.
#3	8	6	7	5	You still didn't make the commands easier to use. I liked the new way of inputting them, but I still want buttons that do it. Everything else but that seemed to be nice though.
#4	6	5	6	7	It was a little less confusing to use it this. I still think this product kind of goes above my head and isn't something I would use. I'd like it better if it was an app instead of a website.
#5	9	6	8	7	I liked having the terminal being embedded to run the commands in the webpage a lot better. It makes it much more convenient to actually use the product. Having buttons to run the commands directly would still have been preferred though.
Overall	7.6	6	6.4	6	-

Conclusion

This project has given myself great insight into the process of designing products with end users in mind. It also highlighted to me how complex it can be to make these kinds of improvements. Planning for this kind of testing is paramount and can be a great struggle. However, with this kind of planning, you still must account for setbacks. The setbacks from my senior design group's project as a whole held back this user testing and led to some major shortfalls with this capstone project.

A large shortfall of this testing was the very slow rate of progress that was happening on my group's senior design project as a whole. Ideally, I would've liked to include more test trials and have had more features usable for each one of these trials. I originally had a goal of conducting five trials in mind. Another shortfall of this testing was the limited diversity of test users. In an ideal environment, I would've included not only more users in my testing, but also would've included a more diverse age group. For testing a product for general public usage, test user diversity is one of the most paramount aspects to have accurate results. My product testers were also not giving me feedback that was very detailed, this issue could've been fixed if my incentives for participation were greater, but I could not do that due to financial reasons.

For all of my project shortfalls, I am pleased with what I was able to learn and achieve despite them. I enjoyed greatly reading about how designing good software and incorporating user feedback. Simply becoming knowledgeable in this area of software development should help me greatly with my future career. These concepts and principles will be on my mind for all of my future work to come. Applying these principles should make me more successful and valuable in my career too. Having easier to use software will

lead to increased user satisfaction and will make it more likely to succeed. I enjoyed greatly reading about how designing good software and incorporating user feedback regardless. The project can be seen on the GitHub repo page created for it which is included in the appendix. Overall, this project has given me more experience with valuable skills and has provided valuable insights for my future career.

Final Design Results

Final design results can be seen in the following figures 7-16.

Figure 7: Home page of Web Server Software

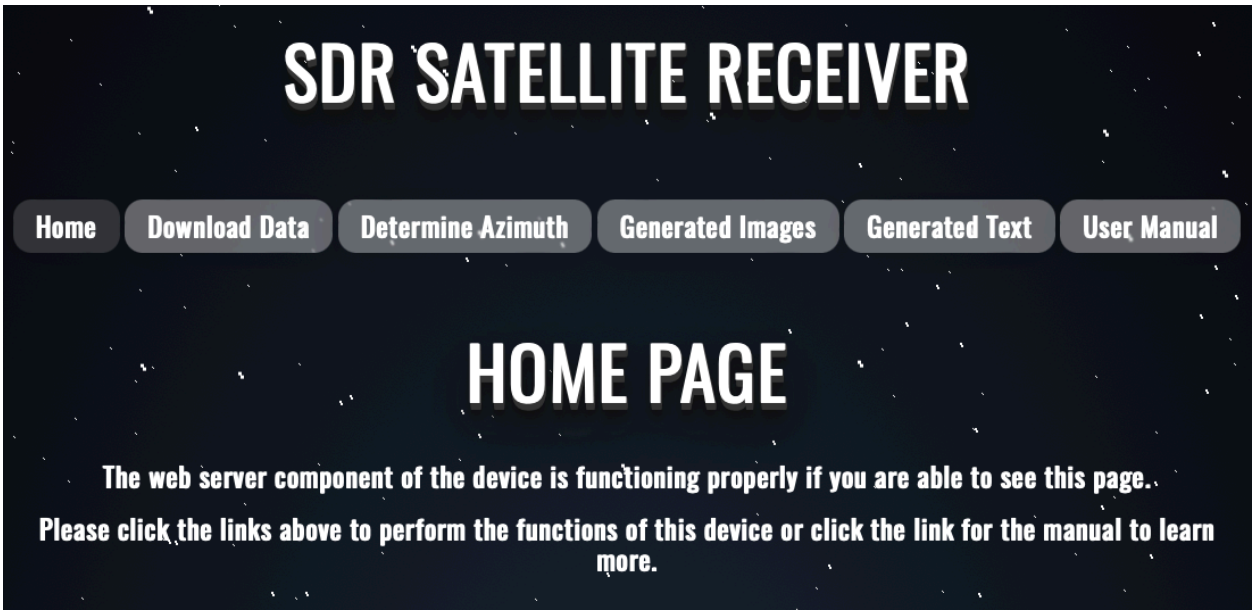


Figure 8: Azimuth Tool page of Web Server Software

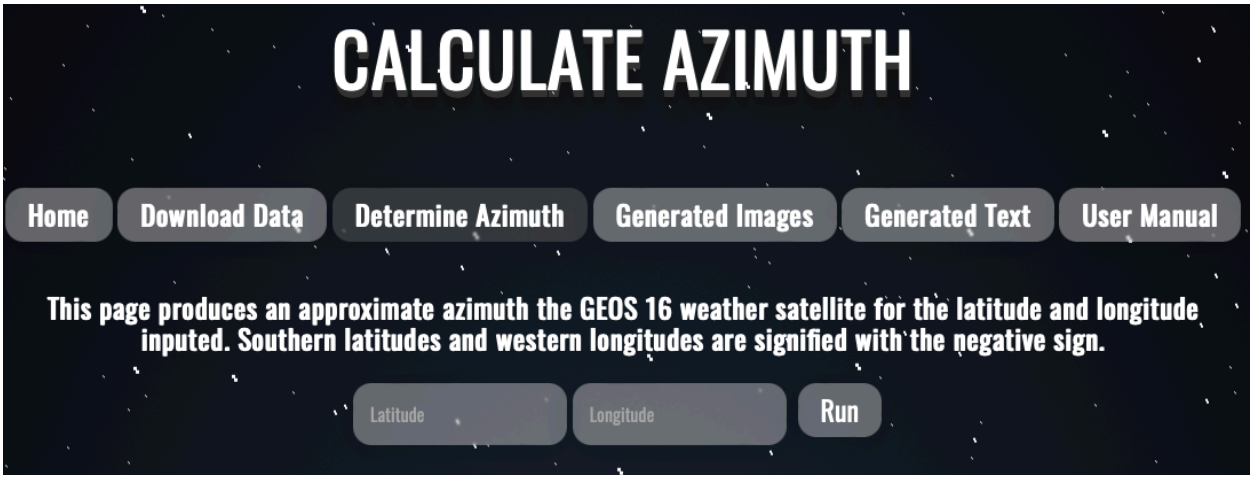


Figure 9: Generated Text page of Web Server Software

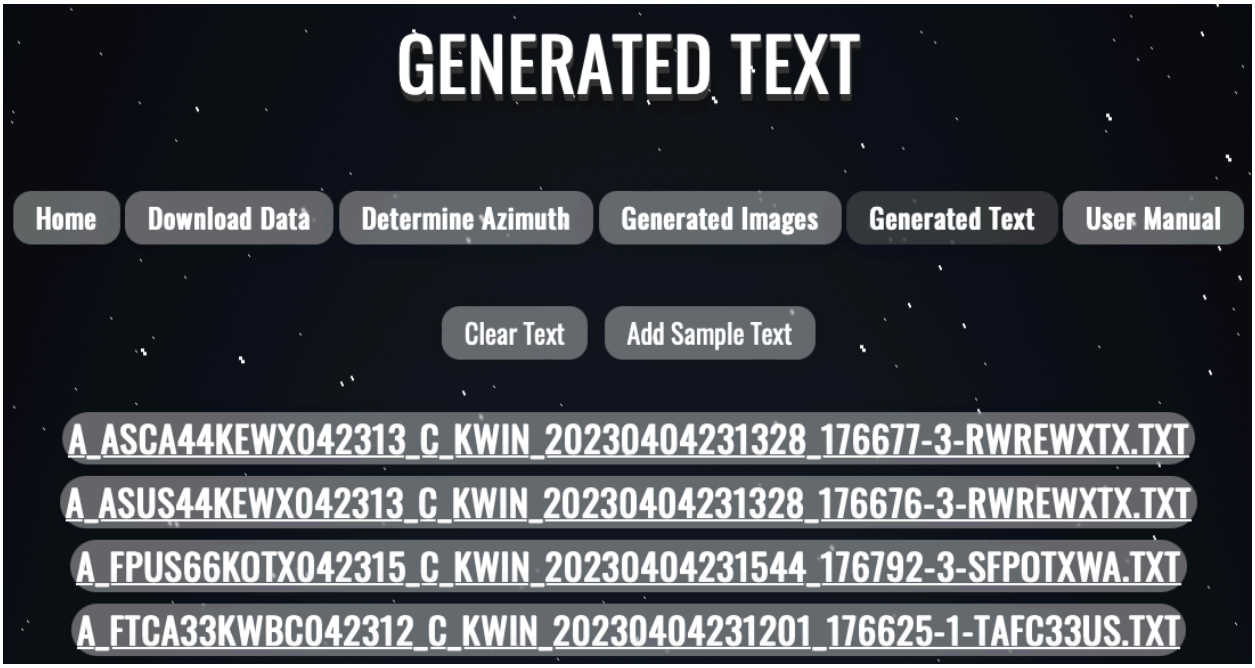


Figure 10: Generated Image page of Web Server Software

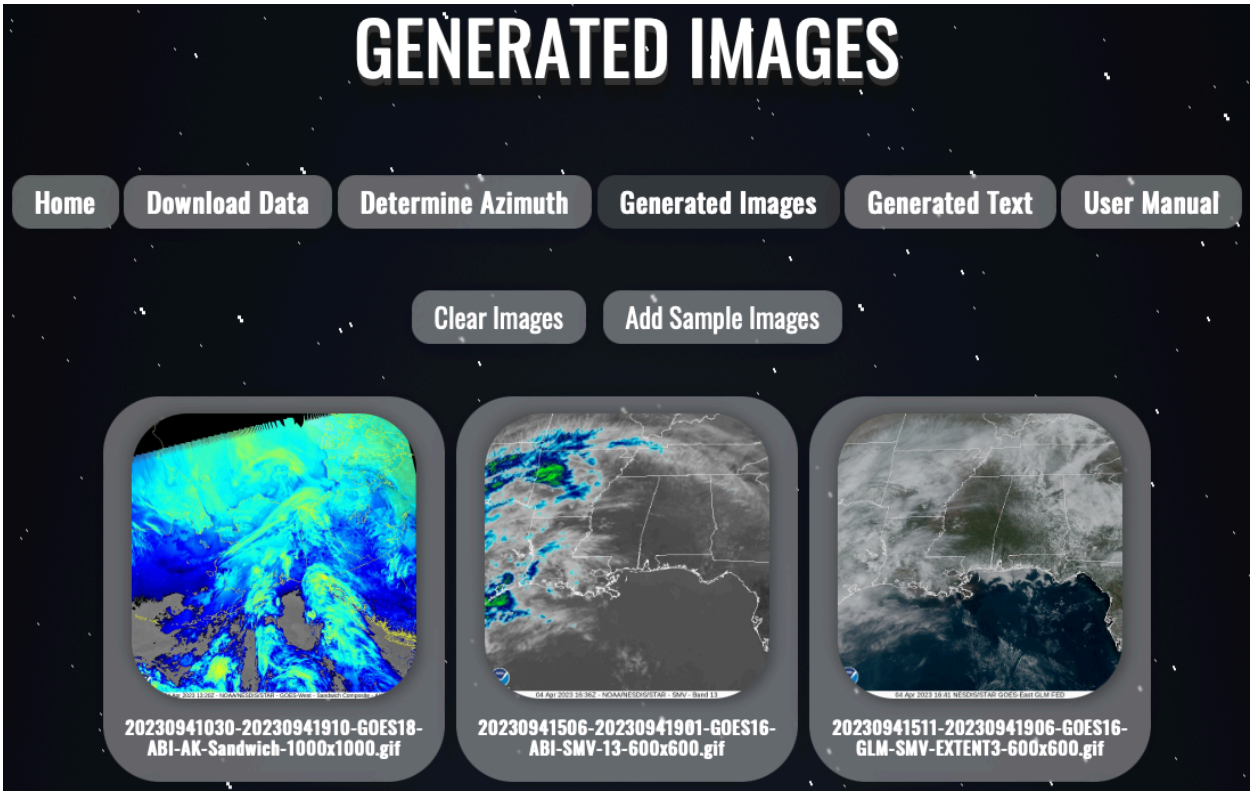


Figure 11: Text File Viewer page of Web Server Software

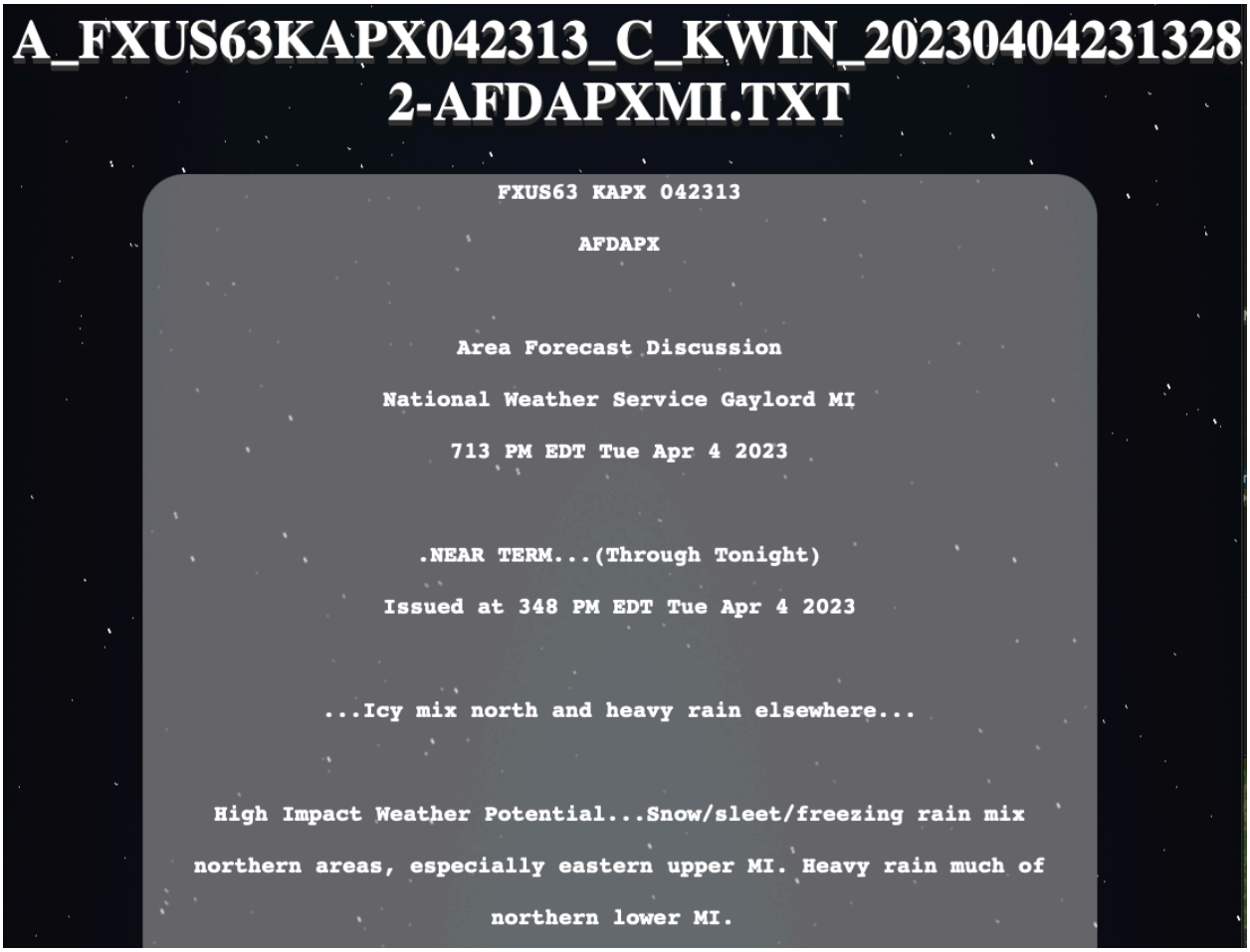


Figure 12: Image File Viewer page of Web Server Software

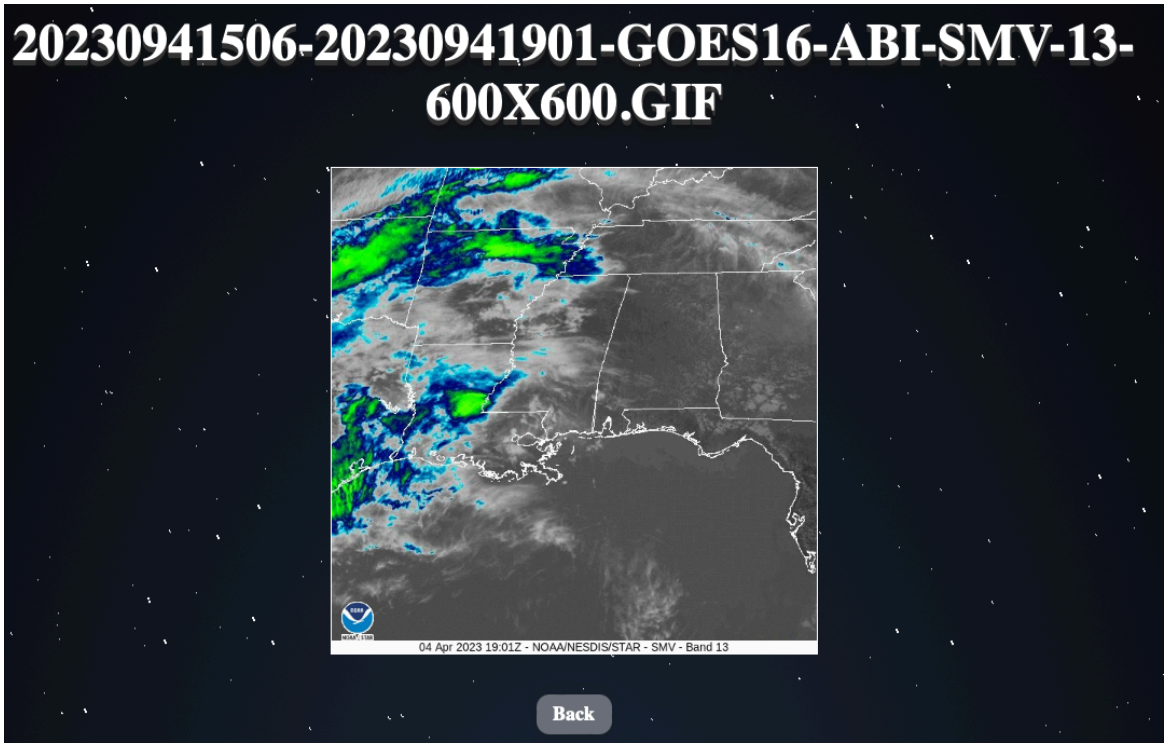


Figure 13: User Manual page of Web Server Software

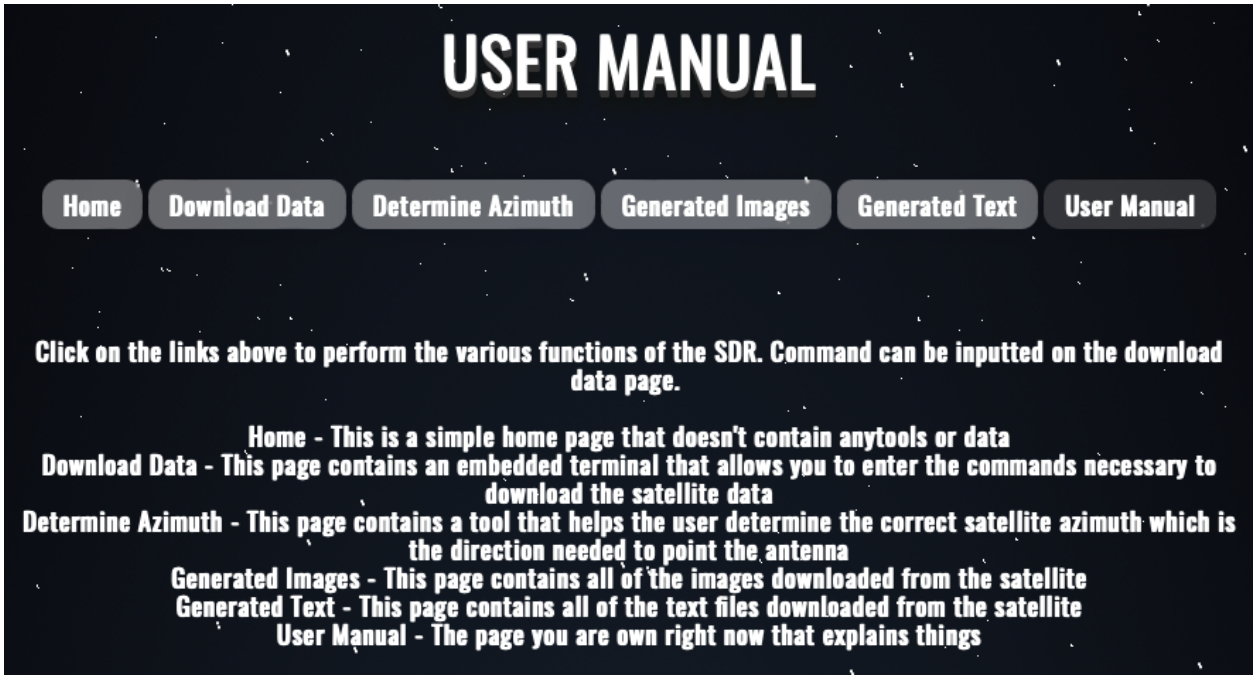


Figure 14: Download Data page of Web Server Software

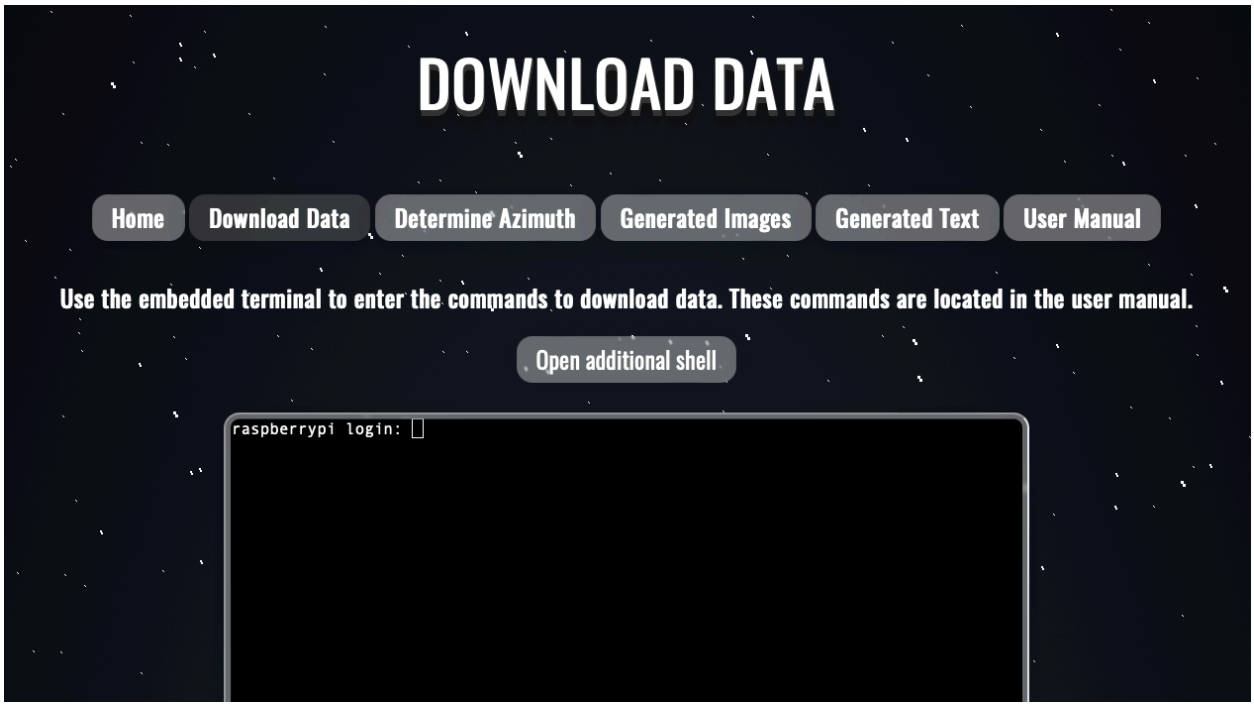
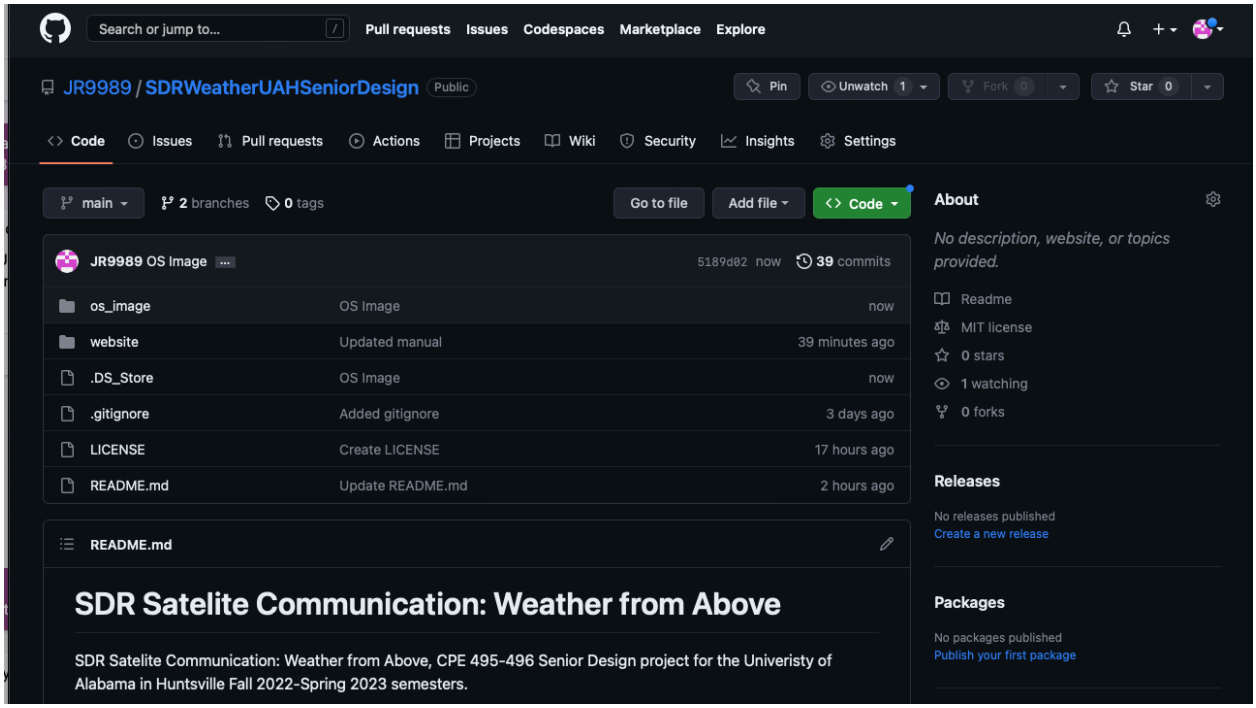


Figure 14: Additional Shell pop-up window from Download Data page of Web Server Software

Figure 15: Product setup of antenna and SDR connected to the Raspberry Pi

Figure 16: GitHub Repository Page Containing the Software of This Project



References

1) Wong, Paco. “Who, What, and Why - a Guide to User Testing Methods.” Toptal. https://www.toptal.com/designers/ux/user-testing-methods.
2) Nielsen, Jakob. “Iterative Design of User Interfaces.” Nielsen Norman Group. November 1, 1993. https://www.nngroup.com/articles/iterative-design/.
3) Rohrer, Christian. “When to Use Which User-Experience Research Methods.” Nielsen Norman Group. July 17, 2022. https://www.nngroup.com/articles/which-ux-research-methods/.
4) Wong, Paco. “Skeuomorphic vs. Flat Design.” UX Office Writings. August 19, 2022. https://ux.iu.edu/writings/skeuomorphic-flat-design/.
5) Kuznetsov, Nick Babich and Gleb. “Why Neumorphism Is Different from the Other Design Trends.” UX Planet. February 11, 2020. https://uxplanet.org/why-neumorphism-is-different-from-the-other-design-trends-efe40571b6b.
6) Hawash, Mutaz AL. “Neumorphism vs Flat & Material vs Skeuomorphism.” Bootcamp UX Design. June 14, 2019. https://bootcamp.uxdesign.cc/neumorphism-vs-flat-material-vs-skeuomorphism-cbcbb4fd913b.
7) Noor, Nor Laila Md. “A Review on User Interface Design Principles to Increase Software Usability for Users with Less Computer Literacy.” Journal of Computer Science 10, no. 12 (2014): 2514–2521. https://doi.org/10.3844/jcssp.2014.2514.2521.
8) Kowalski, Michal. “Glassmorphism in User Interfaces.” UX Collective. January 3, 2020. https://uxdesign.cc/glassmorphism-in-user-interfaces-1f39bb1308c9.

Appendix

- Stooksbury, J.R. “SDRWeatherUAHSeniorDesign.” GitHub. Last modified April, 19, 2023. https://github.com/JR9989/SDRWeatherUAHSeniorDesign.