Internet Backpacks

OREPACKAGERS

(**O**rediggers **R**edesigning and **E**ngineering **P**acking **A**dvancements **C**reatively and **K**nowledgeable **A**lso **G**aining **E**xperience with **R**eal **S**olutions)

Jake Vossen, Nahom Mesfin, Nick Kaiser, William Culver, and Addison Sweeney

Section: U
Date: 4/30/2019

Table of Contents

Proposed Solution

Introduction

Problem Statement

How might we better deliver digital information and assets to places with no or little to no traditional internet access?

According to a study reported by the Washington Post, 4.4 billion people do not have access to the Internet [1]. This includes some countries that have more than 95 percent of the population is without internet. This really inhibits the population of these countries to be involved with economic and social activities. According to the Brooking research group, a 10% increase in internet usage of a country results in “1.7 percent increase in export, and 1.1 increase of imports” [2], that what was in 2004, and the internet has only gotten more important since then. Additionally, countless educational opportunities are online, and more are being added all the time. According to a study done by the Babson Survey Research Group, online education has grown 150.6% [3] from 2003 to 2013. It is clear that it is getting increasingly important as our world gets more connected to enable people in 3rd world countries to be involved.

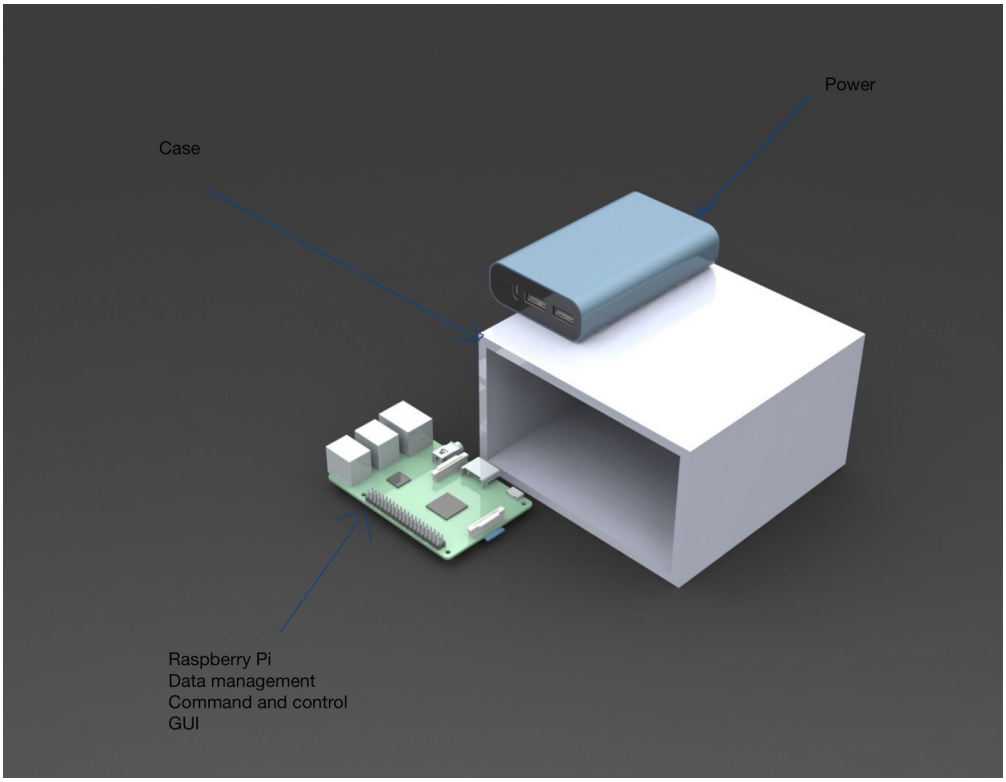

Figure 1: Solidworks Isometric. 2019-04-30 Jake Vossen [37][38]

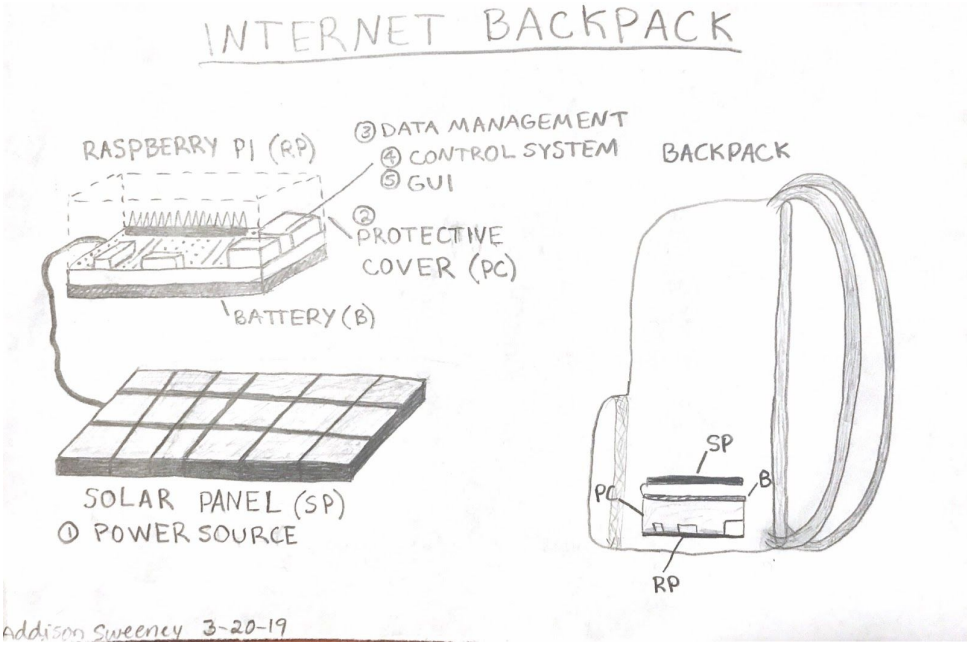

Figure 2: Sketch of System Addison Sweeney 3-20-19

Stakeholder Input

The team is targeting the market of people who already have low-cost smartphones or other wifi-enabled devices, but the cost per GB is prohibitively expensive (or non-existent). The team wants to make requesting and obtaining internet data easier for the user by using 3rd parties to get the information requested, as well as providing curated lists of information that otherwise might not have been found. Wifi-enabled devices are devices that can connect to the internet in places with a local wifi network connection. Since not everywhere has a local wifi network connections available, the system designed by the team can bridge the gap. These devices can act as relay stations, which allows for more remote access and greater impact. Greater impact would mean an increase in quality of life, which can be seen through education. In places with increased access to knowledge and information, which is easily accessed on the internet, quality of life improves. Teachers use online materials to prepare lessons and internet access allows students to extend their range of learning [9].

In an interview with Anthony Wanjiru, who lived in Kenya for the first 30 years of his life, he confirmed that a device to deliver internet access to areas affected by poverty would be able to greatly benefit the people of those areas[4]. His main concern was the ability for the people of the area to repair it and make it on their own. He said that often people would come in and set up a project, but once they left, the people did not know how to operate and repair that. With that input, a conscious effort was made to ensure that all of the subsystems were repairable and expandable. The source code for the software is licensed in such a way that allows for free redistribution and modification, and the components are interchangeable and standardized (any computer can work in place of the Raspberry Pi, the power is over standard Micro-USB cable, etc).

Overview of Subsystem Communications

This diagram describes a good overview of how the subsystems communicate with each other.

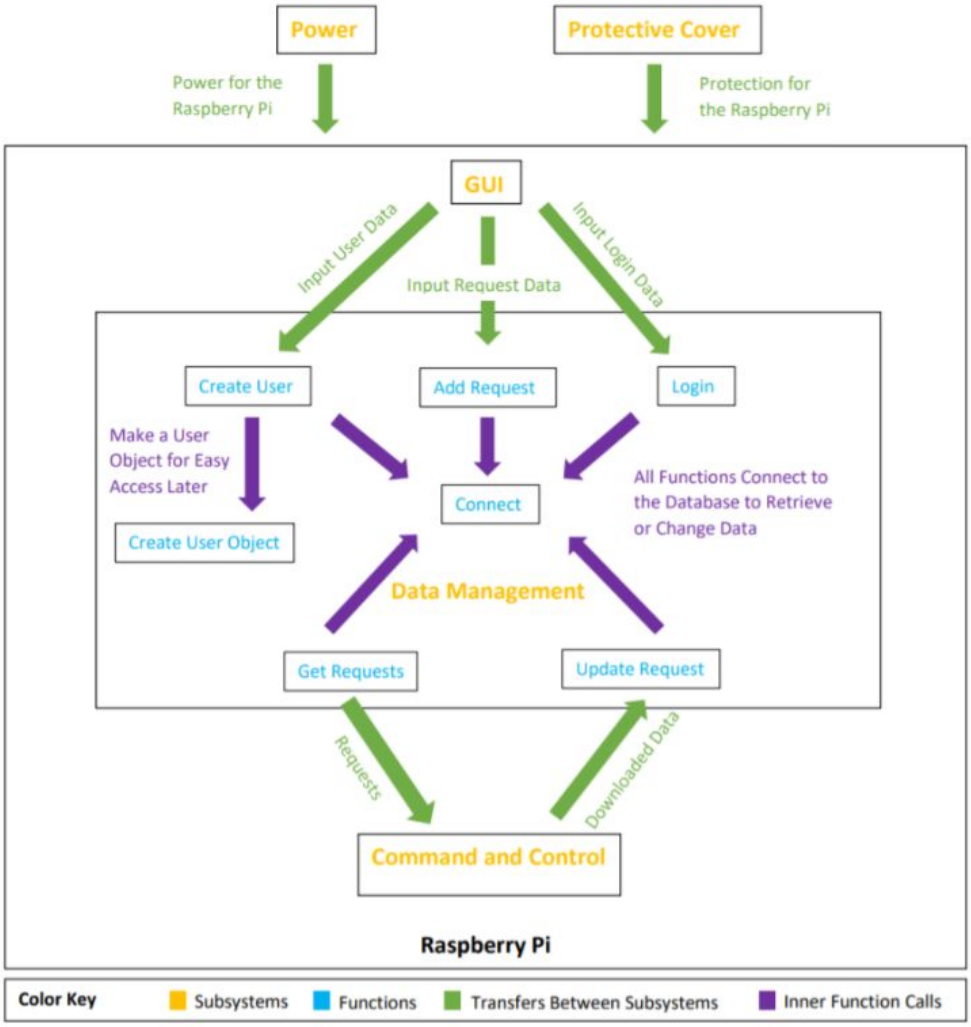

Figure 3: Subsystem and Function Relations. William Culver, 4/2/2019

Full Scale Solution Benefits

Our solution will provide value by decreasing the **cost** of accessing digital assets in places where traditional internet solutions are expensive. You could measure this by comparing the cost of the device (Raspberry Pi, solar panel, etc) compared to the access cost of the internet (economically viable vs the alternatives). We are targeting the market of people who already have low-cost smartphones or other wifi-enabled devices, but the cost per GB is prohibitively expensive (or non-existent). Access to the internet generally results in a quality of life improvement [10], including **education, job count, national GDP, national imports, and national exports**. This value could be measured by analyzing the population of areas this product could be productive in to see if the increase in internet access will be an increase in quality of life benefits. We also want to make requesting and obtaining internet data **easier for**

the user by using 3rd parties to get the information requested, as well as providing curated lists of information that otherwise might not have been found. The value could be measured by the usage of those curated lists and testing on different age groups for ease of use along with measuring time saved by using a 3rd party to get that data.

Concept Validation

Subsystems

The system contains five parts that all must function together in order to work. This includes the power source, the physical container, the data management, the command and control system, and the interface.

The first subsystem is the power source. We will be using a solar panel as our source of power. The power source will connect to the Raspberry Pi and give it enough power to complete all of the requests and functions needed. The power source is the battery and/or solar panel which will connect to the Raspberry Pi using a USB port. Raspberry pi's use 1.21 Watts of power [11] and solar panels (60 cells) contain power output anywhere between 270 to 300 Watts [12]. The power from the solar panel will give the Raspberry Pi power and when that power source is no longer viable, there is a battery for power.

The physical container will house the Raspberry Pi and the battery. The physical cover for the whole system is the second subsystem. It is meant to protect the Raspberry Pi during transportation and use. The cover must be durable and consider weather conditions in rural areas in order to protect the contents of the system in various situations.

The data management subsystem will allow external storage sources to connect to a database that will store user requests as well as the downloaded data. This subsystem will need to check that the data is in the correct format before storing it in the database.

Command and control is a subsystem that will process the requests that are stored in the database. This can process generic search terms, aka "Textbook about Calculus by John Doe", to specific URLs such as "https://www.nytimes.com/2019/03/20/business/google-fine-advertising.html". It will also be able to process different protocols, such as git and IPFS, which will allow for greater possibilities and increase efficiency in the whole system. Once the item has been downloaded, it will connect with the data layer to store the information.

Another subsystem is the interface, which will be the GUI. This is the subsystem that allows the user to interact with the other components of our problems and influence them to the users' needs. This will help send user information to be stored in the data management subsystem, and allow it to know what the user wants to be downloaded.

The command and control, the interface, and the data storage are all coded into the Raspberry Pi. The power source and the physical container are separate from the coding systems in the Raspberry Pi and will have to interact for the system to function. The physical container must have a compartment for the battery and an opening for the power chords to come out of when charging the solar panel or when downloading data.

Power

The team will be using a lithium-ion battery to power the Raspberry Pi. The lithium battery stores 20,000 mAh[36]. In addition, as our back-up power source we will use a solar panel. The solar panel possesses 200 Watts[34]. Moreover, when looking into the climate of our primary area where we want to implement our internet backpacks, Kenya, the weather ranges from cool to warm/hot every day [35]. Knowing that solar panels generate electricity off sunlight and not heat, weather will not be a issue for powering the Raspberry Pi.

Case

As a part of deciding what materials to use for the physical container, research was done to determine what materials would be the strongest and most cost-efficient to use.

Table 1: Container Details [13]

Material	Composition	Physical Properties	Strength (1-5, 5 is highest)	Cost per square foot
Paperboard	Thick paper-based material	Foldability, rigidity, 0.012 in thick	1	$2.08
Plywood	Wood (hardwoods and softwoods)	Stability, high strength, flexibility, fire resistance	3	$2.40
High-density polyethylene plastic	HDPE (thermoplastic polymer)	Hardness, rigidity, ⅛” thickness	2	$1.35
Polymer concrete	Aggregate mixture bound with polymer	Low permeability and corrosive resistance	4	$5
Zinc sheet metal	Zinc	Non-sparking, corrosion resistant, recyclability	4	$3.6

As seen in Table 1, polymer concrete and zinc sheet metal are the two materials that are viable for construction, meaning they are cheap, strong, and water-resistant materials. Zinc metal sheets are significantly lighter and cheaper than polymer concrete, which is why the container is made out of zinc sheet metal.

Table 2: Protective Padding Details [14]

Type of Padding	Composition	Physical Properties	Cost per cubic foot
Packing peanuts	Styrofoam pieces	Filler, protect fragile items, cushion effect	$3.57
Padded divider sets	Polyurethane foam covered with nylon	Suited for small objects, help ship two incompatible things in one package	$1.30
Bubble wrap	little air-filled sacs made from plastic	cushion items, takes great amounts of energy and pressure to pop enough bubbles to render a mailed item unsafe	$3.28
Packaging foam	Denser foam that can be cut into any shape	no definitive shape, firmness, customizable	$2.10
Packing paper	Specialized paper (butcher paper, kraft paper and other kinds)	Dense, sturdy, large volumes provide stability and protection	$1.20

Zinc sheet metal alone will not protect the Raspberry Pi and the battery, so there must be padding inside the metal box. There are various materials that can be used for padding (Table 2). Packaging foam is the best option because it is cheap and will protect the Raspberry Pi, which is vulnerable in the container.

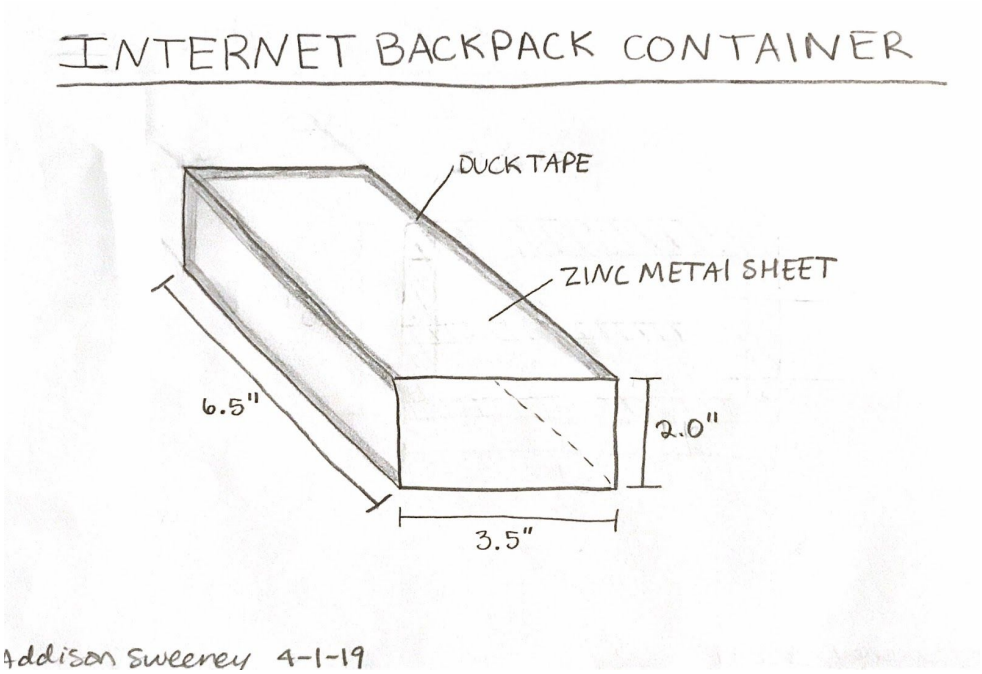

Figure 4 - Physical Container Sketch Addison Sweeney 4-1-19

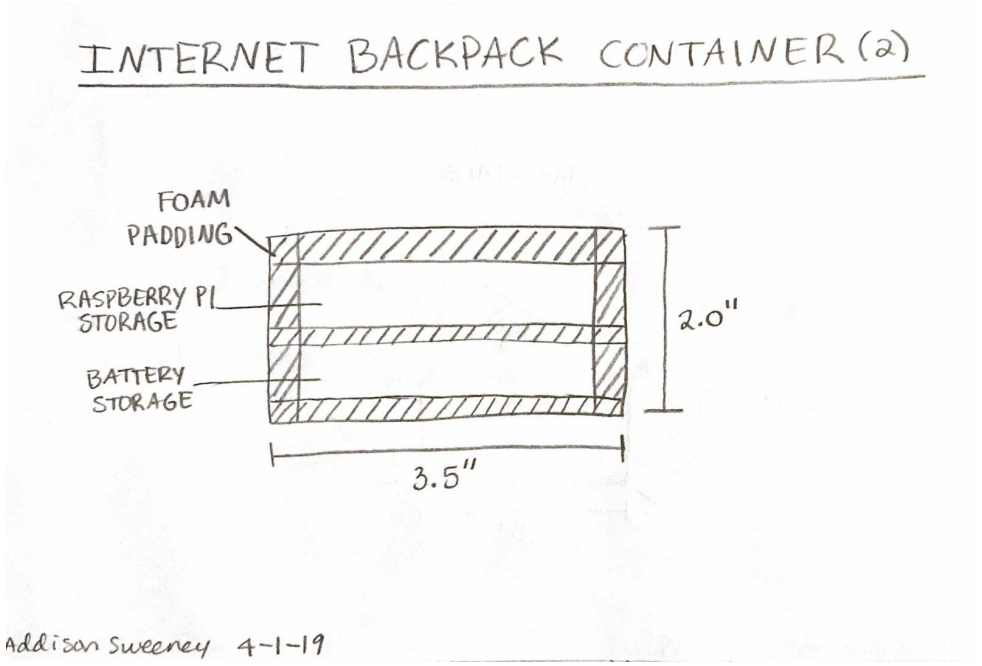

Figure 5 - Physical Container Sketch (Inside) Addison Sweeney 4-1-19

Data Management

Subsystem Description

The data management subsystem is responsible for keeping track of the data associated with users and requests. This subsystem connects to a PostgreSQL database that contains a table for user data and a table for request data. Python code will be run on a Raspberry Pi in order to accomplish tasks associated with the data.

There are seven functions written in Python that are associated with data management. These functions are `connect`, `add_request`, `get_all_requests`, `update_request`, `new_user`, `login`, and `create_user_object`.

The `connect` function is needed for the six other functions to connect to the database. This function uses the psycopg2 library and a PostgreSQL database.

The `add_request` function first connects to the database and creates a new request object. The values for the request object include the ID for the user making the request, the type of request, the request itself, the download status, the location where the requested data will be downloaded to, and the date the request was made. A script command is executed to store the input data into the requests table in the database which can be sorted and accessed later based on any of the request object values. The input values will fill the columns for the request type and the request. The user ID will be filled based on the current user's ID (see Figure 6) and the date will be filled using the date and time from the Raspberry Pi. The download status and location will be populated with default values which will later be updated.

The `get_all_requests` function simply connects to the database and executes a script command that returns the requests table from the database.

The `update_request` function can be called to alter the request object by changing the download status and file location.

The `new_user` function connects to the database and creates a user object based on the input data as long as the entered username is available. This object is then stored in the user table in the database through another script command. Each user object has values for a unique ID, the user's first and last names, their username, and their password. Each password is encrypted before it is stored to provide more security for the users (see Figure 6).

Passwords (Hash Algorithm)

Password	Hash
pa$$word	$pbkdf2-sha256$29000$.1/L.f/fe29t...
wordpass	$pbkdf2-sha256$29000$sHbuXcvZ2xsj...

Users (Database Table)

ID (uuid)	First Name (text)	Last Name (text)	Username (text)	Password (text)
7e12e697...	william	culver	willlynx	$pbkdf2-sha256$29000$.1/L.f/fe29t...
cdaf0eba...	billy	bob	bbob12	$pbkdf2-sha256$29000$sHbuXcvZ2xsj...

Requests (Database Table)

UUID (uuid)	User ID (uuid)	Kind (text)	Value (text)	Download Status (text)	File Location (text)	Date (timestamp)
a38844ef...	7e12e697...	URL	https://www.google.com	True	C:\\Users\william\Downloads\File1	2019-03-31 15:50:03.3537
d9dc60ee...	cdaf0eba...	search	What is the value of pi?	False	NULL	2019-03-31 18:02:55.342
83644514...	7e12e697...	youtube	https://www.youtube.com/watch?v=dQw4w9WgXcQ	False	NULL	2019-03-31 18:05:21.4656

Figure 6: Database Relations. William Culver, 4/2/2019

The login function takes the entered username, if it exists, and pulls only the data associated with that user from the user table in the database. The user's password is then compared to the password that was entered using the same hash algorithm. If the password is correct the function will call the create_user_object function and return true. If it is incorrect the function will return false.

The create_user_object function creates a user object for the current user after they log in for ease of access to their data. Instead of connecting to the database each time the user's data needs to be accessed, this function allows for one database access to the users table while the internet backpack is in use. See the DataManagement.py file in the appendix for code of each function.

Interfaces with Other Subsystems

The GUI subsystem will call functions from the data management code in order to access the database and perform actions with the data. A user will log in by entering their username and password into the GUI which will call the `login` function. When a new user is created from the GUI, the `new_user` function from the data management subsystem will be called. Requests for data can also be made from the GUI which will call the `add_request` function from the data management subsystem. The GUI and data management subsystems were designed with the same user and request objects in mind.

The command and control subsystem will call the `get_all_requests` and `update_request` functions from the data management code in order to download and update requests. Each request will be downloaded in order based on the date and time the request was made so that older requests are downloaded first. Once the data is downloaded, the download status and file location can be updated from their default entries.

The relationship between the data management subsystem and all others can be seen in Figure 6.

Stakeholder Considerations

Since users may be uncomfortable providing their first and last names when making an account, these fields can be left blank. A unique username and a password are all that is required from the user to create an account and log in later. To ensure the protection of the user's requests, all passwords are encrypted before they are stored in the database. The passwords are hashed using the passlib hash library and the PBKDF2-SHA256 algorithm [15]. Hashing prevents passwords from being stolen even if the database containing the passwords is compromised [16]

Validation

In order to validate this subsystem's functionality, a separate Python program was written to allow for user input, to call each data management function, and display the database tables before and after they change. This program was run starting with an empty database and the output can be seen in the DataManagementTests.py file in the appendix. This program tests each function and each special case for functions. First, a new user is created with a hashed password stored in the database. Next, an incorrect password and an incorrect username are entered when logging in which results in an error until the correct combination is entered. A request is added next and updated to change the status and location of the file. Finally, another user is created trying to use the same username which results in an error until a unique username is entered. This user is also created without the first and last name fields to show that they are optional.

Idea Generation and Decision Making Tools

The decision to use Python as the programming language for this subsystem was made based on the device and operating system the program will be run on. This project will be using a Raspberry Pi with the Raspbian operating system installed on it. The Raspbian operating system comes with Python and runs it well [17], so Python was an easy choice for a programming language. Python is also simple to understand yet powerful enough to perform all the tasks for this subsystem.

When choosing a database platform, it came down to either PostgreSQL or MySQL since both use SQL code which is the standard language for databases [18]. Both platforms are free, open source, and powerful enough for this subsystem [19] [20] but PostgreSQL was chosen because it has more features than MySQL which would allow this system to improve and expand in the future [7].

For the password encryption algorithm, the PBKDF2-SHA256 hash algorithm was chosen since it is a top tier algorithm and works on all operating systems [22].

Command and Control

Subsystem Description

The Command and Control subsystem is the subsystem responsible for converting the requests that have been collected into downloaded data to be distributed to users. It starts by receiving a list of request objects - a structure for containing information about each request in Python.

With this list of requests, the first thing it does is use the Python multiprocessing[23] library to split work up between the different threads on the computer. While this software is designed for low-end machines to be more accessible to developing areas, most computers[24] in recent times will have more than 1 CPU core (including the Raspberry Pi[25]). This allows for the processor to split up all the requests, and execute them in parallel, instead of waiting for each one to finish individually, which can provide a decrease in time spent downloading files, as explored in section 4.2.

When downloading a request, it determines the type of request. The types are URL, search, YouTube, and ipfs. Figure 7 is the flow chart to determine the method used for downloading an asset.

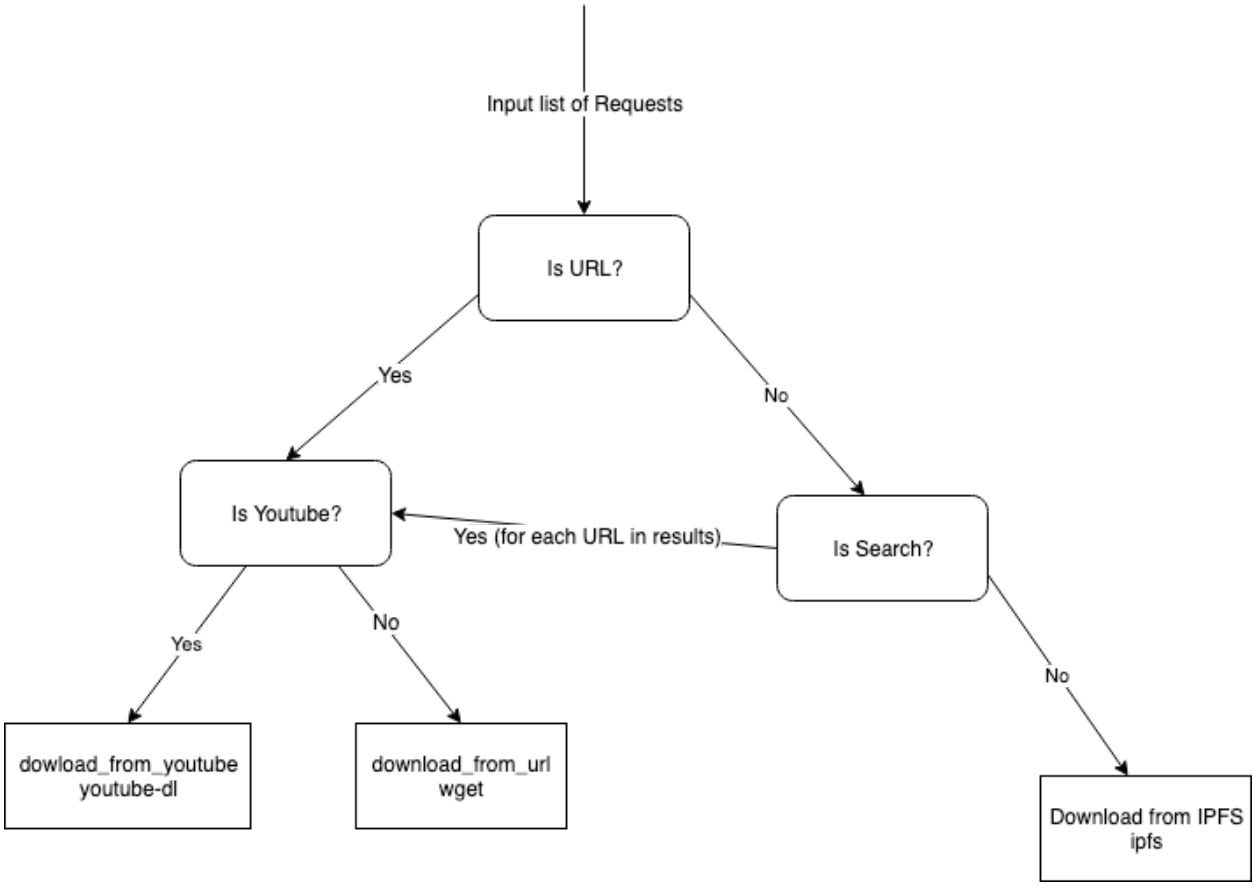

Figure 7: Diagram describing flow chart of downloading each request. Jake Vossen, 2019-04-01

The steps for each type of request are omitted here but can be found in the appendix.

Interfaces with Other Subsystems

Software is all about abstraction, so it is important to clearly define where the Command and Control subsystem will interface with the other subsystems. Ideally, all the other subsystems will work independently and a couple of links will get everything working together. These links are shown in Figure 8, and explained in further detail below. The interfaces are left inside the Python code in the appendix of this document, just commented out with the `#` character.

Most of the meshing is done through the request object, which can be found in the appendix.

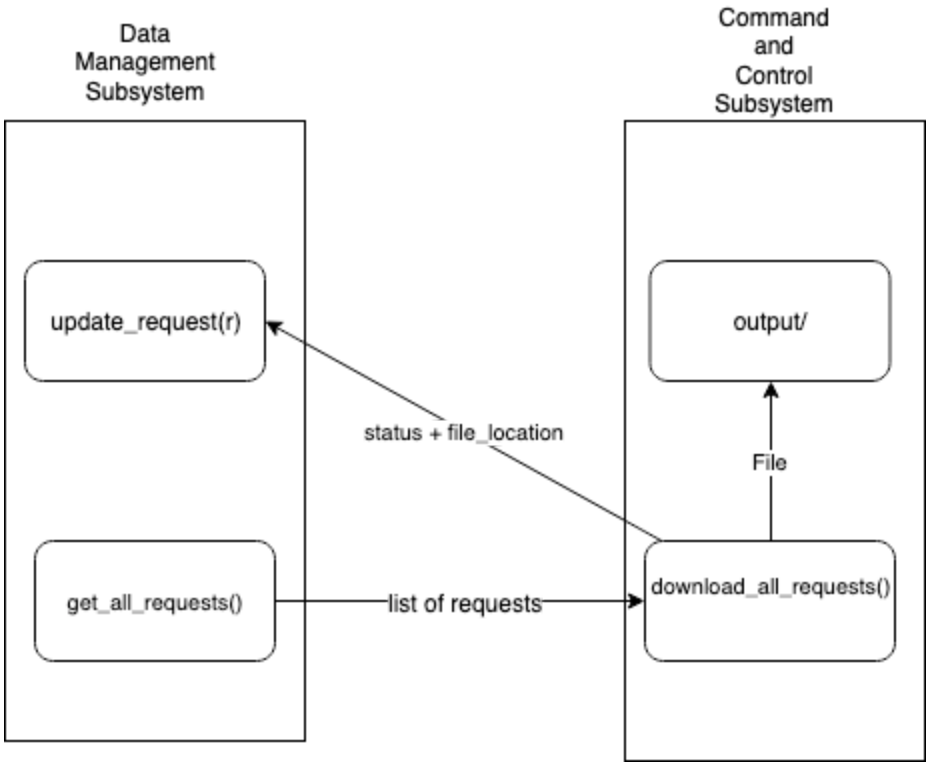

Figure 8 Diagram describing the inputs and outputs of the Command and Control subsystem request. Jake Vossen, 2019-04-01

Input - List of Requests

Python, like all programming languages, only stores objects in memory while the program is running. That means when the program is shut down (or the machine is powered off), the data generated must be saved somewhere on the device or else that information would be lost. In this case, what is important is to be able to store the request objects. This is completed by the data management subsystem. This means my subsystem will call `get_all_requests()` which will retrieve the data from the hard drive of the machine about the requests, create the objects again (as they were destroyed from memory when the program shut down), and return that information to Command and Control.

Output - Download path and status

Once this subsystem completes its download, it needs to update the database about the new status. This is again through the Data Management subsystem. To ensure each object is downloaded only once, each request has two properties: `file_location` and `downloaded_status`. Once Command and Control has completed a download, it will call the Data Management method `update_request(r)` where `r` includes the changes to `file_location` and downloaded_status. So the Data Management subsystem knows which database entry to update, a Universally Unique Identifier (UUID)[10] is used to identify each request.

Stakeholder Considerations

The primary concern for the Command and Control subsystem is ensuring that the primary forms of retrieval are supported. In an interview with Anthony Wanjiru, who lived in Kenya for 35 years, he said the most important types of content are either a URL (either as a web page, pdf, etc), YouTube, Google Searches, and while IPFS isn't widely used, it has a lot of potential[4]. He also said that local content is on the rise, and a method to upload content would be really beneficial. While this was not implemented at this time, due to specifics with integration with the other subsystems, this could be implemented later.

Validation

File Retrieval

Testing is one of the most fundamentally important things in software development. It is easy to write code, but it is not useful if it can't work under a variety of difficult tests. A handful of curated tests designed to test the boundaries of the code have been picked. To verify the results, there are two primary methods. The first is a SHA-512 sum, which is a cryptographically secure way to ensure two files are the same[11], the first file being originally from a manual download, and the second one from the Command and Control Subsystem Python program. If the two hashes are the same, the subsystem was successful. However, this cannot be used for all tests, because when downloading from a website, the original HTML files are modified for browsing offline (see the -k option on wget). This makes it impossible to cryptographically ensure that the requested file is the same as the downloaded file, so a visual analysis is performed to analyze the two documents and ensure that they are identical. The table below explains the tests performed, where the "Analysis Method" is either visual or hash to perform the check.

Table 3: File Download Tests Performed, Jake Vossen, 2019-04-01

Type	Value	Expected Result	Analysis Method	Expected = Actual?
URL	https://www.gutenberg.org/cache/epub/2265/pg2265.txt	Text of “Hamlet” By Shakespeare	Hash	Yes
URL	https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46507.pdf	PDF of Google Machine Learning Research Paper	Hash	Yes
URL	https://en.wikipedia.org/wiki/Monty_Python_and_the_Holy_Grail	Wikipedia entry for “Monty Python and the Holy Grail”	Visual	Yes
YouTube	https://www.youtube.com/watch?v=Gbtulv0mnlU	NASA Shuttle Recovery Video	Visual	Yes
search	What Is the Airspeed Velocity of an Unladen Swallow?	Google search results	Visual	Yes
ipfs	/ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme	IPFS introduction document	Hash	Yes
ipfs	/ipfs/QmVLTMHtLRhnft3QspDx4qTJeXY6hiib1j77UfQmY54CGe/mosaic.png	300MB picture of the moon	Hash	Yes

Multithreading

Table 4 Tests Performed, Jake Vossen, 2019-04-01

1 Core	2 Cores	4 Cores	8 Cores
161.277 seconds	53.927 seconds	45.984 seconds	46.448 seconds

From this data, it is clear that the multithreading is successful in decreasing the time required to download multiple requests. Further work could be done to multi thread the `googlesearch` library, as the current multi threading solution does not support sub threads.

Idea Generation / Software Choices and Requirements

Below is the reasoning for each software library and tool picked. It also acts as a requirements list.

- Python - Easy for beginners to understand, has a lot of libraries, powerful
- `googlesearch` - It is really difficult to parse the URLs and submit a search to Google through plain HTTP requests, and this library takes care of this as well as providing updates if the format changes in the future
- `youtube_dl` - Written in Python, does exactly what is needed, and also supports sites that aren't YouTube which expands the usefulness of this project
- `wget` - Open source, widely used, and can create a snapshot as the website exactly appears, instead of just the file at the specific URL.
- IPFS - Allows for the decentralization of content and therefore increases download speed

GUI

This subsystem serves as the connection between the user and the rest of the functionality of the project. The subsystem will, when fully implemented, will have several functionalities. The GUI needs to be able to keep track of the requests they have made, as well as see the items they have ready for download and also allow the user to download those items. The user also needs to be able to input a request, and the request type, such as a URL link, or description of the item desired. In order to accomplish these requirements, the GUI subsystem will be implemented using a python script that runs a web server that the user would be able to access by connecting to the Raspberry Pi. This script will create the two input areas for the user to input a request, as well received the receipts for the requests so that the user can go and look at what they have requested once it is ready for download. The final design can be seen below (see Figure 9).

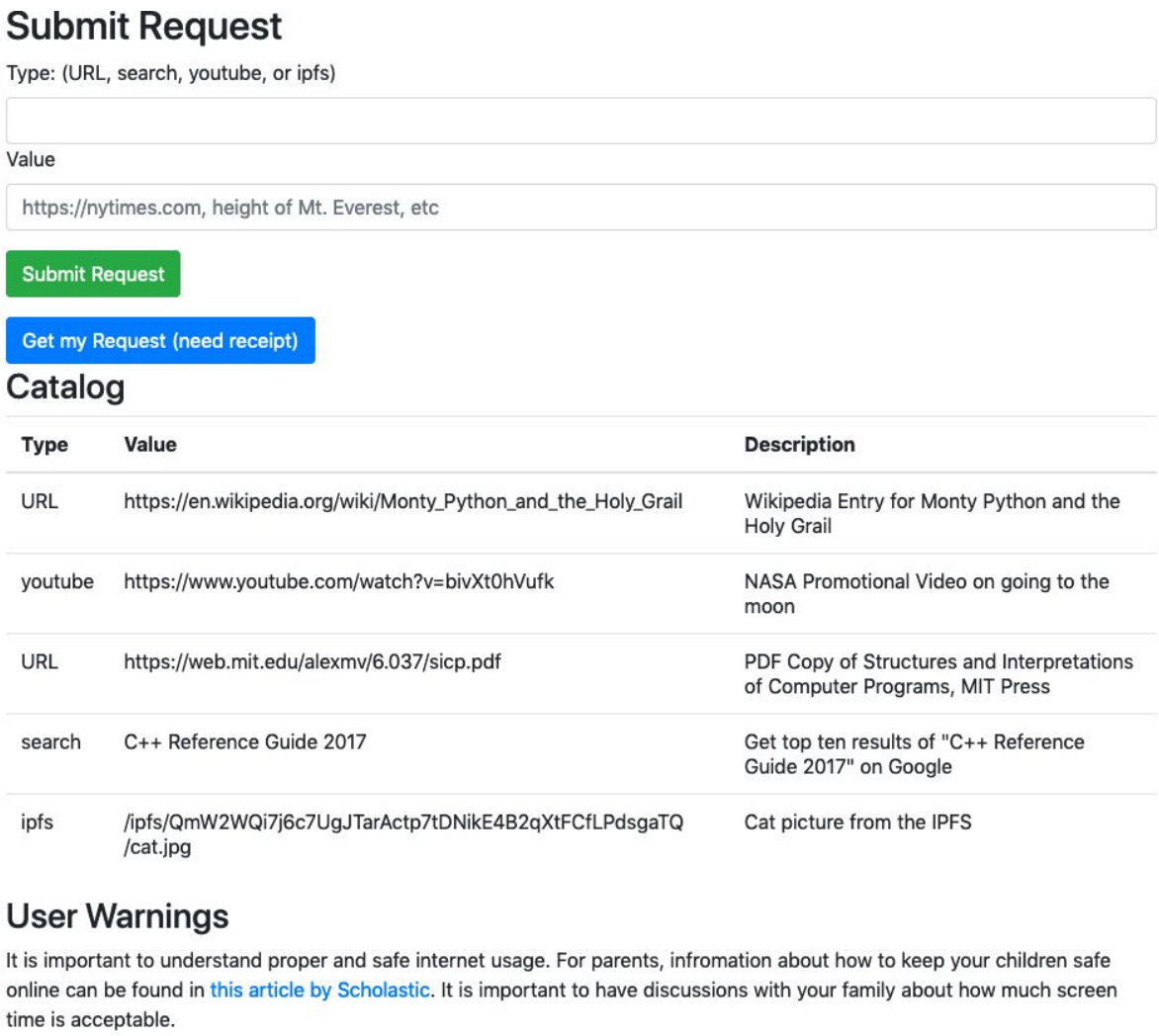

Figure 9, GUI Design.

The way that the GUI subsystem connects to the other subsystems is fairly simplistic; it is represented by the transfer of information between the GUI and the database management subsystem. There is also the connection to the power subsystem in the sense that the GUI needs electricity to function properly. The GUI initially starts out by receiving data from the database management subsystem and then once the user is looking at the right receipt, shows the user the right information. As the user gives the GUI requests, they are also sent from the GUI to the database management subsystem to be processed by command and control and stored.

Since the stakeholders for the GUI subsystem are just people who would be using it, the stakeholder consideration consists of user testing the GUI. These tests were conducted with two peers who have significant experience with using GUIs and other computer applications. These two stakeholders said that as an overall subsystem, it was good [6], but would need some tweaks and a few new features to be more user friendly and simplistic [6]. Other than the minor changes they suggested, they said that the GUI worked well and had no found errors in how it functioned.

Several of these suggestions were implemented and can be seen in the final design. They also tested the functionality by running through the system in the mindset of someone with little experience with computers.

One of the methods of testing the GUI was through the use of the Validation and verification technique. This technique of testing is where the tester first validates the program by making sure that it has all of the required functionality, then the tester can use either static or dynamic verification techniques [8]. In the case of this subsystem, a static verification method was used, in which the tester went through the program and its code to look for any flaws or errors in the system [8]. It is often helpful to have people who had nothing to do with the creation process, be the tester for this 'V and V' testing method [5]. This is because they don't know what the creators did to prevent the user from doing something they aren't supposed to, and will likely try to do these things, and find ways to work around the barriers in place. This was proven with having the two stakeholders test the program because they were able to find multiple issues that could have caused the GUI to not function properly. As an example, the request system allowed for the input of an empty string [6], which would take up a spot in the database but not have any downloadable information.

A good way of coming up with ideas for a program is to model existing ideas, and observe how they work with what users expect, make changes based on these observations, and keep cycling through this process [7]. This method, along with the method shown in Table 2 were used when coming up with the ideas and specifics of the GUI. Each requirement of the GUI was thought of, and then initially there was a few ideas for how to implement each of them. This was narrowed down using a method discussed in class, as seen in Table 5, and then these ideas were refined through the process of observation and changes.

Table 5 GUI Decision Matrix

	Idea	Feasibility	User-Friendly	How it looks	Total (max 15)
Lists	Making the Lists show entries by pages	4	2	5	11
	Making the Lists show entries in an ordered list	4	3	3	11
	Showing them on separate pages, accessed by receipt	4	3	5	13
Type-selection	Bubble selection	5	3	4	12
	Dropdown menu	5	3	4	12
	Text-Field	5	4	5	14

Works like Validation

Once the subsystems were complete and individually tested, the whole solution had to be tested. Most of this work was done in application.py, which can be found in the appendix. We ran the server online, and used the same tests and testing strategies as in table 3, and verified that all of the information was transferred successfully. Instead of hard-coding the information, we ran the server on a Raspberry Pi, and entered the requests through the GUI.

Conclusion

By creating an affordable and robust system for connecting people to the internet, more people can now have access to education, communication, and the connection to the greater world. As discussed in the value proposition, an increase in access to the internet is associated with an

improvement in life in just about any measurable metric, including education, GDP, and national imports and exports. Everything in this device is interchangeable, and the software is free to use and distribute, to ensure maximum use.

References

[1] R. A. Ferdman, "4.4 billion people around the world still don't have Internet. Here's where they live," The Washington Post, 02-Oct-2014. [Online]. Available: https://www.washingtonpost.com/news/wonk/wp/2014/10/02/4-4-billion-people-around-the-world-still-dont-have-internet-heres-where-they-live/?utm term=.b81eab12af53. [Accessed: 31-Jan-2019].

[2] F. Dews and F. Dews, "How the Internet and Data Help the Developing World," brookings.edu, 29-Jul-2016. [Online]. Available: https://www.brookings.edu/blog/brookings-now/2014/02/06/how-the-internet-and-data-help-the-developing-world/. [Accessed: 31-Jan-2019].

[3] E. I. Allen and J. Seaman, "GRADE LEVEL TRACKING ONLINE EDUCATION IN THE UNITED STATES," Feb-2015. [Online]. Available: https://files.eric.ed.gov/fulltext/ED572778.pdf. [Accessed: 27-Apr-2019].

[4] J. J. Vossen and A. Wanjiru, "Discussions of Content Distribution in Kenya."

[5] C. T. Fitz-Gibbon and L. L. Morris, How to design a program evaluation. Newbury Park, CA: Sage, 1994.

[6] R. H. Thompson, R. F. Erickson 03-Apr-2019, "GUI Stakeholder Interview."

[7] Zaraté Pascale, Tools for collaborative decision-making. London: ISTE, 2013.

[8] A. Funes and A. Dasso, Verification, Validation and Testing in Software Engineering. IGI Global, 2007.

[9] "Internet Access and Education: Key considerations for policy makers," Internet Society. [Online]. Available: https://www.internetsociety.org/resources/doc/2017/internet-access-and-education/. [Accessed: 03-Apr-2019].

[10] F. Dews and F. Dews, “How the Internet and Data Help the Developing World,” brookings.edu, 29-Jul-2016. [Online]. Available: https://www.brookings.edu/blog/brookings-now/2014/02/06/how-the-internet-and-data-help-the-developing-world/. [Accessed: 31-Jan-2019].

[11] “How Much Less Power does the Raspberry Pi B use than the old model B?,” RasPi.TV, 16-Jul-2014. [Online]. Available: https://raspi.tv/2014/how-much-less-power-does-the-raspberry-pi-b-use-than-the-old-model-b. [Accessed: 14-Mar-2019].

[12] A. Sendy, “How much output do you get from solar power panels?,” Solar Reviews, 28-Apr-2018. [Online]. Available: https://www.solarreviews.com/blog/what-is-the-power-output-of-a-solar-panel. [Accessed: 14-Mar-2019].

[13] “The Home Depot,” The Home Depot. [Online]. Available: https://www.homedepot.com/. [Accessed: 03-Apr-2019].

[14] “5 Types of Protective Packaging - Grainger Industrial Supply,” Grainger. [Online]. Available: https://www.grainger.com/content/supplylink-types-of-protective-packaging. [Accessed: 03-Apr-2019].

[15] “PasswordHash Tutorial,” Passlib v1.7.1 Documentation, 2017. [Online]. Available: https://passlib.readthedocs.io/en/stable/narr/hash-tutorial.html. [Accessed: 31-Mar-2019].

[16] “Safe Password Hashing,” ISU. [Online]. Available: http://ld2015.scusa.lsu.edu/php/faq.passwords.html. [Accessed: 31-Mar-2019].

[17] “Download Raspbian for Raspberry Pi,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Accessed: 02-Apr-2019].

[18] “What is SQL?,” SQLCourse, 20-Aug-2000. [Online]. Available: http://www.sqlcourse.com/intro.html. [Accessed: 02-Apr-2019].

[19] “About,” PostgreSQL. [Online]. Available: https://www.postgresql.org/about/. [Accessed: 31-Mar-2019].

[20] MySQL Editions,” MySQL. [Online]. Available: https://www.mysql.com/products/. [Accessed: 02-Apr-2019].

[21] “Databases,” Full Stack Python. [Online]. Available: https://www.fullstackpython.com/databases.html. [Accessed: 02-Apr-2019].

[22] J. Salvatierra, “Encrypting passwords in Python with passlib,” The Teclado Blog, 06-Dec-2017. [Online]. Available: http://blog.tecladocode.com/learn-python-encrypting-passwords-python-flask-and-passlib/. [Accessed: 31-Mar-2019].

[23] “multiprocessing - Process-based parallelism,” multiprocessing - Process-based parallelism - Python 3.7.3 documentation. [Online]. Available: https://docs.python.org/3.7/library/multiprocessing.html. [Accessed: 03-Apr-2019].

[24] “Hardware Survey - CPU Cores,” Hardware Survey - CPU Cores. [Online]. Available: https://www.pcbenchmarks.net/number-of-cpu-cores.html. [Accessed: 03-Apr-2019].

[25] “Raspberry Pi 3 Model B,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 03-Apr-2019].

[26] “A Universally Unique IDentifier (UUID) URN Namespace,” IETF Tools. [Online]. Available: https://tools.ietf.org/html/rfc4122.html. [Accessed: 03-Apr-2019].

[27] C. H. Romie, “Secure Hash Standard (SHS)” Aug-2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf. [Accessed: 02-Apr-2019].

[28] “GNU Wget 1.20 Manual,” GNU Wget 1.20 Manual. [Online]. Available: https://www.gnu.org/software/wget/manual/wget.html. [Accessed: 03-Apr-2019].

[29] MarioVilas, “MarioVilas/googlesearch,” GitHub, 06-Mar-2019. [Online]. Available: https://github.com/MarioVilas/googlesearch. [Accessed: 03-Apr-2019].

[30] Ytdl-Org, “ytdl-org/youtube-dl” GitHub, 03-Apr-2019. [Online]. Available: https://github.com/ytdl-org/youtube-dl/. [Accessed: 03-Apr-2019].

[31] “IPFS is the Distributed Web” IPFS. [Online]. Available: https://ipfs.io/. [Accessed: 03-Apr-2019].

[32] “Client-Server Mode,” [Online]. Available: https://web.cs.wpi.edu/~cs513/s07/week1-unixsock.pdf. [Accessed: 02-Apr-2019].

[33] “IPFS Documentation,” Install IPFS – IPFS Documentation. [Online]. Available: https://docs.ipfs.io/introduction/install/. [Accessed: 03-Apr-2019].

[34] B. Zientara, “How much electricity does a solar panel produce?,” *Solar Power Rocks*, 11-Feb-2019. [Online]. Available: https://www.solarpowerrocks.com/solar-basics/how-much-electricity-does-a-solar-panel-produce/. [Accessed: 04-Apr-2019].

[35] “Climate - Kenya,” *Kenya climate: average weather, temperature, precipitation, best time*. [Online]. Available: https://www.climatestotravel.com/climate/kenya. [Accessed: 29-Apr-2019].

[36] “Anker PowerCore II 20000, 20100mAh Portable Charger with Dual USB Ports, PowerIQ 2.0 (up to 18W Output) Power Bank, Fast Charging for iPhone, Samsung and More (Compatible with Quick Charge Devices),” *Amazon*. [Online]. Available: https://www.amazon.com/Anker-PowerCore-20100mAh-Portable-Compatible/dp/B01LQ81QR0/ref=sr_1_1?keywords=anker powercore II 20000&qid=1556564100&s=hpc&sr=8-1. [Accessed: 29-Apr-2019].

[37] *3D CAD Model Collection | GrabCAD Community Library*. [Online]. Available: https://grabcad.com/library/raspberry-pi-3-board-1. [Accessed: 30-Apr-2019].

[38] *3D CAD Model Collection | GrabCAD Community Library*. [Online]. Available: https://grabcad.com/library/belkin-6600mah-battery-pack-1. [Accessed: 30-Apr-2019].

Appendix

Risk Analysis and Mitigation Plan

Table 6: Risk Analysis Matrix

Likelihood of Risk				
	Very likely	Dangers of travel		
	Likely	Less social interaction and family time	Exposure to dangerous or illicit content	
	Unlikely			Battery malfunction
		Minor	Moderate	Major
		Level of Risk		

In order to mitigate these risks, several warnings will be included with the product as well as guidelines in place for the use of this product. There will be a warning downloaded on the Raspberry Pi informing the user of the dangers of the internet before they submit requests. This doesn't guarantee that they won't become exposed to unwanted content, but it will likely decrease the chances. In this warning, there will also be a notice telling the user of the importance of family time. Since internet access provided from this product will not be continuous or on demand, the risk of less social interaction is low but still important to warn the user about.

The final product will also include a warning for the battery since it is a lithium-ion battery and could be dangerous if mishandled. Misuse of the battery is unlikely but could result in serious injuries.

Lastly, if this product is to be transported between locations by motorized vehicles, there will be guidelines for the drivers. These guidelines will include restrictions on how long they can drive each day and what times they can and can't drive to prevent fatigue and other risks associated with the road.

The risks of this product did not lead to adding a section on the GUI about safe and responsible internet usage, especially for children.

Prototype Cost

Table 7: Cost of Materials for Prototype

Item	Cost	Notes
Raspberry Pi	$38.10	Donated
Solar Panel	$49.99	Donated
Battery	$25.95	Donated
Zinc Metal Sheet	$19.47	24in x 36in 26-gauge
Ottertex Nylon Casing	$3.95	1.9 OZ
Hot Glue, duck tape	$0.00	Used design lab resources

Request Object(request.py)

```
import uuid
class Request:
        'This is the main object that stores information about somones request'
        def __init__(self, kind, value, user, date):
        self.kind = kind # options: "search, url, IPFS hash, git URL
        self.value = value # the value associated with the type
        self.user = user # The user associated with this request. We can make a user object later if we feel like it
        self.date = date # This should be a date-time object
        self.downloaded_status = False # This is not included in the constructor because state should always be false when starting
        self.uuid=str(uuid.uuid4())
        self.file_location = "" # There is no downloaded location when the object is created
        def set_downloaded_status(self, state):
```

```
        self.downloaded_status = state # Should be a boolean
        def set_file_location(self, loc):
        self.file_location = loc # Is there a file object that would be better to use than a string?
```

CommandAndControl.py

```
from request import Request
from user import user
from datetime import datetime
import uuid
import subprocess
import os
from googlesearch import search
import youtube_dl
import multiprocessing
from multiprocessing import Pool

def download_all_requests(requests):
        threads_count = multiprocessing.cpu_count() # use all of the threads on the system that we can
        # Below are the hard coded values for the testing of the effectiveness of threads
        # threads_count = 1
        # threads_count = 2
        # threads_count = 4
        # threads_count = 8
        with Pool(threads_count) as p:
        p.map(download_request, requests)
        # The above code is the same as the code below, above will do it with as many threads as possible

        # for r in requests:
        # download_request(r)

def mkdir(path): #mkdir = Make Director, creates a folder if it does not exist
        if not os.path.exists(path):
        os.makedirs(path)

def download_request(r):
```

```
        if r.downloaded_status:
        return
        if r.kind == "URL": # we are dealing with a plain old HTTP request
        url = r.value
        path = "output/" + r.uuid
        download_from_url(url, path)
        if r.kind == "search":
        search_list = list(search(r.value, stop = 10)) # call the google_search library to get the list of urls for the search terms stored in r.value
        for i in range(len(search_list)): # it would nice to multithread this, but we are already multithreaded at the request level. More work could be done to improve this system.
        url = search_list[i]
        path = 'output/' + r.uuid + '/' + str(i + 1) + '/'
        if ('youtube' not in url): # Youtube vidoes are not going to be able to work, those will be for the youtube request
                download_from_url(url, path)
        else:
                download_from_youtube(url, path)
        if r.kind == "youtube":
        path = "output/" + r.uuid + '/'
        download_from_youtube(r.value, path)
        if r.kind == "ipfs":
        path = "output/" + r.uuid + '/'
        download_from_ipfs(r.value, path)
        r.set_file_location(path)
        r.set_downloaded_status(True)

        # mark_as_downloaded(p) # this is where this function will intergrate with data managment to update the file location and downlaoded status

def download_from_url(url, path):
        mkdir(path)
        subprocess.call(r'wget -E -H -k -K -p -P ' + path + ' ' +  url + ' robots=off ', shell=True)

def download_from_youtube(url, path):
        mkdir(path)
        ydl_opts = {
        'outtmpl': path + '%(title)s' + '.mkv' # add title to file name
```

```
        }
        with youtube_dl.YoutubeDL(ydl_opts) as ydl:
        ydl.download([url])

def download_from_ipfs(ipfs_hash, path):
        mkdir(path)
        subprocess.call(r'ipfs get ' + ipfs_hash + ' -o ' + path, shell=True) #this requires the
IPFS daemon running

 def main():
        requests = get_all_requests() # This is where command and control will interafce with
Data Managment. It will return the list of requests from the database.
        requests = [] # Because we can't do the above method, we are going to create a mock
list with requests
        # tests
        requests.append(Request("URL",
"https://www.gutenberg.org/cache/epub/2265/pg2265.txt", user("Jake", "Vossen",
"jakevossen", "asdf"), datetime.now()))
        requests.append(Request("URL",
"https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/46507.pdf",
user("Jake", "Vossen", "jakevossen", "asdf"), datetime.now()))
        requests.append(Request("URL",
"https://en.wikipedia.org/wiki/Monty_Python_and_the_Holy_Grail", user("Jake", "Vossen",
"jakevossen", "asdf"), datetime.now()))
        requests.append(Request("search", "What Is the Airspeed Velocity of an Unladen
Swallow?", user("Jake", "Vossen", "jakevossen", "asdf"), datetime.now()))
        requests.append(Request("search", "Library of Congress", user("Jake", "Vossen",
"jakevossen", "asdf"), datetime.now()))
        requests.append(Request("youtube",
"https://www.youtube.com/watch?v=NtrVwX1ncqk", user("Jake", "Vossen", "jakevossen",
"asdf"), datetime.now()))
        requests.append(Request("youtube",
"https://www.youtube.com/watch?v=Gbtulv0mnlU", user("Jake", "Vossen", "jakevossen",
"asdf"), datetime.now()))
        requests.append(Request("ipfs",
"/ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme", user("Jake",
"Vossen", "jakevossen", "asdf"), datetime.now()))
        requests.append(Request("ipfs",
```

```
"/ipfs/QmVLTMHtLRhnft3QspDx4qTJeXY6hiib1j77UfQmY54CGe/mosaic.png",
user("Jake", "Vossen", "jakevossen", "asdf"), datetime.now()))

        download_all_requests(requests)

# main() # start the program (uncomment when using as library)
```

application.py

```
from flask import Flask, render_template, flash, request, send_file
from wtforms import Form, TextField, TextAreaField, validators, StringField, SubmitField
import CommandAndControl
from request import Request
from user import user
import uuid
from datetime import datetime
from pathlib import Path
import subprocess
import os
# App config.
DEBUG = True
application = Flask(__name__)
application.config.from_object(__name__)
application.config['SECRET_KEY'] = '7d451f27d441f27567d441f2b6176a'
requests = []


class ReusableForm(Form):
        name = TextField('Name:', validators=[validators.required()])
        email = TextField('Email:')#, validators=[validators.required(),
validators.Length(min=6, max=35)])
        # password = TextField('Password:', validators=[True])

@application.route("/", methods=['GET', 'POST'])
def submit_gen_request():
        form = ReusableForm(request.form)

        print (form.errors)
```

```
        if request.method == 'POST':
        name=request.form['name']
        email=request.form['email']
        print (name, " ", email)

        if form.validate():
        r = Request(name, email, user("Jake", "Vossen", "jakevossen", "asdf"), datetime.now())
        requests.append(r)
        # CommandAndControl.download_all_requests(requests)
        flash('Thanks for the submission, your receipt is ' + r.uuid)

        # else:
        # flash('Error: All the form fields are required. ')

        return render_template('hello.html', form=form)


@application.route("/get-my-request", methods=['GET', 'POST'])
def getmyrequest():
        form = ReusableForm(request.form)

        print (form.errors)
        if request.method == 'POST':
        name=request.form['name']
        print (name, " ")

        if form.validate():
        space_seperated_ids = ""
        for input_uuid in name.split(','):
        print("input uuid: ", input_uuid)
        if os.path.isdir("output/" + input_uuid): # only add the elements that have been
downloaded
                space_seperated_ids += input_uuid + " "

        space_seperated_ids = space_seperated_ids[:-1] # remove final charachter
        print(space_seperated_ids)
        return_id = str(uuid.uuid4())
        # subprocess.call(r'pwd')
```

```
        subprocess.call(r'cd output; zip -r ' + return_id + '.zip ' + space_seperated_ids,
shell=True)
        return send_file('output/' + return_id + '.zip', as_attachment=True)

        return render_template('get-my-request.html', form=form)

@application.route("/google-research", methods=['GET', 'POST'])
def googleresearch():
        form = ReusableForm(request.form)

        print (form.errors)
        if request.method == 'POST':
        name=request.form['name']
        email=request.form['email']
        print (name, " ", email)

        if form.validate():
# Save the comment here.
        r = Request(name, email, user("Jake", "Vossen", "jakevossen", "asdf"), datetime.now())
        requests.append(r)
        # CommandAndControl.download_all_requests(requests)
        flash('Thanks for registration, your receipt is ' + r.uuid)



        else:
        flash('Error: All the form fields are required. ')

        return render_template('research.html', form=form)

@application.route("/download", methods=['GET', 'POST'])
def download_requests():
        CommandAndControl.download_all_requests(requests)
        # requests = []
        return render_template('download.html')

if __name__ == "__main__":
        application.debug = True
```

```
        application.run(host='0.0.0.0')
```

DataManagment.py

```
import psycopg2
import datetime
from passlib.hash import pbkdf2_sha256
from request import Request
from user import user

current_user = user(None, None, None, None)
user_request = Request(None, None, None, None)


def connect():
        global connection
        # connect to the database
        connection = psycopg2.connect(
        host="127.0.0.1",  # depends on host
        database="internetdata",
        user="postgres",
        password="uD4ph2stu",
        port="5432"
        )


def add_request(kind, value, user):
        global user_request
        connect()
        # cursor for database
        cursor = connection.cursor()
        now = datetime.datetime.now()
        # create a request object and add data to database
        user_request.__init__(kind, value, user, now)
        cursor.execute("INSERT INTO requests (\"UUID\", \"User ID\", \"Kind\", \"Value\",
\"Download Status\", "
                "\"Date\") VALUES (%s, %s, %s, %s, \'False\', %s)",
```

```
                (user_request.uuid, user_request.user.ID, user_request.kind, user_request.value,
now))
        # make changes and close the cursor
        connection.commit()
        cursor.close()
        connection.close()


def get_all_requests():
        connect()
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM requests")
        rows = cursor.fetchall()
        cursor.close()
        connection.close()
        # returns a tuple where the first index is the row and the second index is the column
        return rows


# I added the user_request parameter so you have to specify which request to update
def update_request(current_request, status, location):
        connect()
        current_request.set_downloaded_status(status)
        current_request.set_file_location(location)
        cursor = connection.cursor()
        cursor.execute(f"UPDATE requests SET \"Download Status\" =
\'{current_request.downloaded_status}\', "
                f"\"File Location\" = \'{current_request.file_location}\' WHERE \"UUID\" = "
                f"\'{current_request.uuid}\'")
        connection.commit()
        cursor.close()
        connection.close()


# I added this function
# This creates a new user from the user object and inserts the data into the users table
def new_user(first_name, last_name, username, password):
        global hash
```

```
        connect()
        cursor = connection.cursor()
        usernameAvailable = True
        cursor.execute("SELECT * FROM users")
        rows = cursor.fetchall()
        for r in rows:
        if r[3] == username.lower():
        usernameAvailable = False
        if usernameAvailable:
        # password is hashed
        hash = pbkdf2_sha256.encrypt(password)
        current_user = user(first_name, last_name, username, hash)
        # first name, last name, and username converted to lowercase so logging in is not case
sensitive
        cursor.execute(f"INSERT INTO users (\"ID\", \"First Name\", \"Last Name\",
\"Username\", \"Password\") "
                f"VALUES (\'{current_user.ID}\', \'{current_user.first_name.lower()}\', "
                f"\'{current_user.last_name.lower()}\', \'{current_user.username.lower()}\', "
                f"\'{current_user.password}\')")
        connection.commit()
        cursor.close()
        connection.close()
        return True
        else:
        cursor.close()
        connection.close()
        return False


# I added this function
# This checks if the username and password are correct in order to login
# It then creates a new user object which can be used throughout the program for different
purposes
def login(username, password):
        connect()
        cursor = connection.cursor()

        userExists = False
```

```
        cursor.execute("SELECT * FROM users")
        rows = cursor.fetchall()
        for r in rows:
        if r[3] == username.lower():
        userExists = True

        if userExists:
        # entered username is converted to lower case to match what is in the database
        cursor.execute(f'SELECT * FROM users WHERE \"Username\" =
\'{username.lower()}\'")
        rows = cursor.fetchall()
        hash = rows[0][4]
        cursor.close()
        connection.close()
        # the entered password is compared to the database password using the hash and
encryption algorithm
        if pbkdf2_sha256.verify(password, hash):
        ID = str(rows[0][0])
        create_user_object(ID)
        return True
        else:
        return False
        else:
        cursor.close()
        connection.close()
        return False


# Creates a user object for the current user so that their data can be accessed easily
def create_user_object(uuid_num):
        global current_user
        connect()
        cursor = connection.cursor()
        cursor.execute(f'SELECT * FROM users WHERE \"ID\" = \'{uuid_num}\'")
        rows = cursor.fetchall()
        current_user.ID = rows[0][0]
        current_user.first_name = rows[0][1]
        current_user.last_name = rows[0][2]
```

```
        current_user.username = rows[0][3]
        current_user.password = rows[0][4]
        cursor.close()
        connection.close()
```

DataManagementTests.py

```
import psycopg2
from DataManagement import *


# add_request test
def add_request_test():
        print("Add Request Test: ")
        kind = input("Kind: ")
        value = input("Value: ")
        add_request(kind, value, current_user)


# get_all_requests test
def get_all_requests_test():
        print("Get All Requests Test: ")
        rows = get_all_requests()
        print("(UUID, User ID, Kind, Value, Download Status, File Location, Date)")
        for r in rows:
        print(r)


# update_request test
def update_request_test():
        print("Update Request Test: ")
        update_request(user_request, True, "C:\\\\Users\\william\\Downloads\\File1")
        print("Done")


# new_user test
def new_user_test():
```

```
        print("New User Test: ")
        first_name = input("First Name: ")
        last_name = input("Last Name: ")
        username = input("Username: ")
        password = input("Password: ")
        usernameAvailable = new_user(first_name, last_name, username, password)
        while not usernameAvailable:
        print("That username is taken please choose another.")
        username = input("Username: ")
        password = input("Password: ")
        usernameAvailable = new_user(first_name, last_name, username, password)
        if usernameAvailable:
        break


# login test
def login_test():
        print("Login Test: ")
        username = input("Username: ")
        password = input("Password: ")

        login_success = login(username, password)

        while not login_success:
        print("Incorrect Username or Password. Try Again.")
        username = input("Username: ")
        password = input("Password: ")
        login_success = login(username, password)
        if login_success:
        break

        print("Login Success")


# to print the current user table from the database
def print_user_table():
        connection = psycopg2.connect(
        database="internetdata",
```

```
        host="127.0.0.1",
        user="postgres",
        password="uD4ph2stu",
        port="5432"
        )
        print("User Table: ")
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM users")
        rows = cursor.fetchall()
        print("(ID, First Name, Last Name, Username, Password)")
        for r in rows:
        print(r)
        cursor.close()
        connection.close()


# tests to test all functionality for the data management subsystem
new_user_test()
print_user_table()
login_test()
add_request_test()
get_all_requests_test()
update_request_test()
get_all_requests_test()
new_user_test()
print_user_table()


# sample test output starting from a blank database
'''
New User Test:
First Name: William
Last Name: Culver
Username: Willlynx
Password: pa$$word
User Table:
(ID, First Name, Last Name, Username, Password)
('36c984dc-5b3d-4733-9c20-b638765b7657', 'william', 'culver', 'willlynx',
```

'$pbkdf2-sha256$29000$E4IQwjiHUEpJae0do9Qaww$xBDNUFsX3G6yVnNlsqpTZzPp0YQ26cCZxWnKqBv3PZg')
Login Test:
Username: willlynx
Password: password
Incorrect Username or Password. Try Again.
Username: will
Password: pa$$word
Incorrect Username or Password. Try Again.
Username: willlynx
Password: pa$$word
Login Success
Add Request Test:
Kind: URL
Value: https://www.google.com
Get All Requests Test:
(UUID, User ID, Kind, Value, Download Status, File Location, Date)
('4a9d96e8-3ec4-40cc-983a-bd92bc05fab9', '36c984dc-5b3d-4733-9c20-b638765b7657', 'URL', 'https://www.google.com', 'False', None, datetime.datetime(2019, 4, 2, 19, 55, 42, 785000))
Update Request Test:
Done
Get All Requests Test:
(UUID, User ID, Kind, Value, Download Status, File Location, Date)
('4a9d96e8-3ec4-40cc-983a-bd92bc05fab9', '36c984dc-5b3d-4733-9c20-b638765b7657', 'URL', 'https://www.google.com', 'True', 'C:\\\\Users\\william\\Downloads\\File1', datetime.datetime(2019, 4, 2, 19, 55, 42, 785000))
New User Test:
First Name:
Last Name:
Username: Willlynx
Password: pass
That username is taken please choose another.
Username: BillyBob
Password: pass
User Table:
(ID, First Name, Last Name, Username, Password)
('6f2f477a-e7c7-4fca-bec4-414ba0d0b579', 'william', 'culver', 'willlynx',

```
'$pbkdf2-sha256$29000$0tq7N2ZsjdEag1AKoTTG2A$GHUQSW5Hd9SvTSmI3flnCCqa924
ftCI9iCufSsh1gKg')
('a9be6453-5d34-4599-bb7e-8e44c52fbb2e', '', '', 'billybob',
'$pbkdf2-sha256$29000$vdeac661thYixLhXqlVqDQ$zHziXsJVwLDsPdo25FQmy3V6UiFq
G4XQr5Sccy0qr70')
'''
```

User.py

```
import uuid
class user:
        def __init__(self, first_name, last_name, username, password):
        self.ID = str(uuid.uuid4())
        self.first_name = first_name
        self.last_name = last_name
        self.username = username
        self.password = password  # in hashed form
```

Team Photo / Biographies

From left to right: Addison Sweeney, Jake Vossen, Nahom Mesfin, William Culver, Nick Kaiser

Addison Sweeney Biography

My name is Addison and I'm a freshman here at Mines studying mechanical engineering. I play on the club soccer team here at Mines and love to ski, hike, and be outside. I'm also a part of Pi Beta Phi Fraternity and love to volunteer and help the community.

Jake Vossen Biography

Hello! I am a first year student, majoring in Computer Science. I am a Colorado native and lived here about my whole life. I enjoy backpacking, as well as hunting and fishing. I also like programming competitions / hackathons, and have a passion for cybersecurity, CS education, and algorithms.

William Culver Biography

Hi, my name is William and I am a freshman from Nederland, Colorado. I am majoring in Computer Science and love all things programming. Outside of school, I'm in two a capella groups at Mines (the Harmonic Miners and Miner Dissonance) and I'm a goalie on the Mines club ice hockey team.

Nahom Mesfin Biography

My name is Nahom and I am freshman from Highlands Ranch, Colorado. My major is Computer Science. Some things I love to do outside of school is play basketball, watch basketball and football, and going to the gym to work out.

Nicholas Kaiser Biography

My name is Nicholas and I am a Sophomore from Lafayette, Colorado. I am majoring in Computer Science. Outside of school I like to go climbing. I have competed in the Mines' bouldering competition every semester since joining and really enjoy it.

Decision Matrix

Team Name:	*OREPACKAGERS*					
Instructions:	Type in your Design Alternatives, your Criteria, the Definition of that criateria, and your weightings					
	Scale 0 = worst 5 = Best					
	Design Objectives (criteria)					**Overall Score**
Design Alternatives	Viability (Financial)	Feasibility	Reliablity	Speed	Scope of Impact	
Internet Backpack	4	3.5	3	1.5	5	**17**
Drones	1	3	2	4.5	5	**15.5**
"Amazon" Power Hubs	3	2.5	4.5	3	4	**17**
Radio Internet	2	2.5	4.5	3	3	**15**
						0
						0
Criteria Definition:						
Viability (Financial)	*If this solution where to be implemented, would it make sense to use over alternatives?*					
Feasibility	*Is this possible to build?*					
Reliablity	*If this solution where implemented, would it be feasible to repair / upgrade*					
Speed	*Does this solution provide "fast" internet?*					
Scope of Impact	*Where can it go, and how many people can be effected by it?*					
Note: there are comments on some cells, look for the orange triangle in the upper right corner and mouse over to view.						

Figure 10: Decision Matrix

Omitted Subsystem Report sections

Command and Control

Steps for downloading different materials

URLs

URLs are your basic websites, such as https://en.wikipedia.org/wiki/Monty_Python_and_the_Holy_Grail, or https://www.nytimes.com/2019/03/27/technology/turing-award-ai.html. This is for users who already know the content they want. In the backend, the Python program is going to use the `wget` utility. Specifically, `wget -E -H -k -K -p -P path url robots=off` where path is the output directory and url is the url that has been requested. Each of those letters does something to modify the request to created the desired behavior. For example, -p tells wget to download pictures and things linked on the website so the page appears exactly as it would in a web browser. More details for each option can be found in the wget documentation[28]. All of those options ensure that downloading the URL requested gets the website exactly as it appears in a browser, including linked images. Additionally, it works with PDF and ZIP files, which is really important to ensure all possible media can be obtained. This method is also used by other parts of the program.

Search

Sometimes the user will not know exactly what they want, so functionally was added to get the first page of Google search results (top 10 results). The `googlesearch`[29] library was very helpful for this. This library provides a list of the URL's on the first page of a Google Search. With that list, the established URL method described in section 1.1 is used to download each URL (it also checks to ensure to use the YouTube method if it is a YouTube link). The results are each in their own folder named based on the Google search rank (1 is the first result, 2 is the second result, etc).

YouTube

It is well known that a lot of quality educational and entertainment content is in video format, and the majority of that content is on YouTube. That is why the users can request YouTube videos (through a link, or a result from the search function). In this case, the `youtube-dl`[30] program allows for content retrieval. A user can either request a YouTube video directly, or use the URL method which will detect the YouTube link and use this method.

IPFS

IPFS stands for “InterPlanetary File System”, which is a “A peer-to-peer hypermedia protocol to make the web faster, safer, and more open”[31]. The internet that is familiar to most people is the client-server model[32], but IPFS changes that so everyone is both a client and a server. Media is distributed based on the cryptographic hash, a unique ID for each object, instead of a URL. Anybody can add objects, and when requesting an object, it can be downloaded from any number of servers, not just the original person hosting the server. The ipfs command line utility[33] is used to retrieve objects. This greatly expands the functionality of our product, because the operator can now do some content retrieval with no access to the internet. If two of these backpacks meet where there was no internet, they can easily exchange content on the IPFS, which could increase speed and decrease the cost of retrieval for those requests.