Jet Propulsion Laboratory Download Manager
(JPLDM)

CS496 Senior Design

Project Document

Prepared by:

Team Members:
Abdias Andres
Rowan Edge
Mariah Martinez
Gregory Miles
Adrian Rendon
Kevin Tu

Faculty Advisors:
Kang, Elaine
Zhu, Yuqing

May 5, 2016

CALIFORNIA STATE UNIVERSITY LOS ANGELES

Los Angeles, California

JPL Download Manager
(JPLDM)

Table of Contents

Section 1: Executive Summary

For this project, our client was Jet Propulsion Laboratory(JPL). JPL is the leading United States center for robotic exploration of the solar system. They have multiple spacecrafts and instruments to fulfill their space based astronomy missions. JPL currently runs a web portal called the Lunar Mapping and Modeling Project(LMMP) that provides access to collections of lunar data such as image mosaics, digital elevation models, hazard assessments maps, lighting maps and models, gravity models, and resource maps. One perspective of LMMP is as an initial step in an interplanetary geographic information system, such as Google maps. Currently, the LMMP is hosted via an online portal that is cumbersome to navigate, download files and search for lunar geographic information files. Our group was tasked with creating an download manager that would streamline, optimize and increase the efficiency of lunar geographic information file search and download.

The LMMP makes heavy use web technologies that have fallen out of vogue, such as Adobe Flash. Because Flash lacks the dominance and popularity on the web it used to enjoy, it is beginning to lack browser support, such as in versions of Chrome coming out this year. For the LMMP project this presents a problem since the primary technology used to navigate the portal is a Flash based user interface. Additionally, as an initial step in an interplanetary geographic information system, LMMP will need the ability to scale to increased customer and data usage. One of the biggest buzzwords in the current tech world is the cloud. One of the main selling points of cloud services is scalability and responsiveness to a web product's traffic without having to build, maintain, configure and provide space for new infrastructure. Because of these benefits of cloud services our team designed the download manager around an Amazon Web Services (AWS) backend.

AWS is a wide variety of services and we settled on AWS Simple Storage Service (S3) with CloudFront as the primary cloud services. AWS S3 would allow for storage of the lunar geographic files while CloudFront provides a Content Distribution Network (CDN) for increased file transfer speeds. Another aspect of our project was to build a Representational State Transfer (REST) Application Programming Interface (API) utilizing AWS to allow for
scripting and other developers to be able easily integrate the LMMP into their projects. For the client side we developed a Graphical User Interface (GUI) that would provide the opportunity for our team to customize the download manager capabilities to interact with server side geographic information files.

Our project has delivered the basic components of a download manager, such as LMMP file search and retrieval along with providing a CDN. For our GUI we used the JavaFX software platform. JavaFX can be customized with Cascading Style Sheets (CSS) and provides the functional programming aspects of Java 8, although we did find a learning curve. Our code is structured so that future developers and maintainers of the project can quickly take advantage of our lessons learned and code base to extend and update the download manager. The JPL download manager was also a wonderful

opportunity to work with the Jet Propulsion Laboratory and as a team we were grateful for the time our liaisons took out of their busy schedules to assist us.

Section 2: Introduction

For our project, JPL Download Manager(JPLDM), in collaboration with Jet Propulsion Laboratory(JPL) to create a desktop application that allows users to download and access JPL's extensive lunar mapping images.

JPL's Lunar Mapping and Modeling Portal(LMMP) has their own downloading process however it was shown to be too troublesome to use. As well as creating a strain on JPL's servers if there was too many downloads running at once. Our fix is the JPL Download Manager which will display all lunar images available for download to users in a compact graphical user interface. Users will be able to pick and choose the images they would like to download from the JPL database hosted at Amazon's storage services. Users will also be able to preview the images before downloading to ensure the images are correct before proceeding. The JPLDM will give real-time updates to the status of downloads such as the time remaining and download speed along with full control over how the user would like to download the images.

Section 3: User Guide

3.1 Developer Instructions

Tools Needed

- Eclipse IDE https://eclipse.org/downloads/

Once Environment is Ready:

1. Unzip [download-manager..zip] file
2. Open Eclipse
3. File > Import > Existing Projects Into Workspace
4. Select the root directory of the unzipped file [.../download-manager]
5. Click the run button to launch the application

3.2 User Instructions

1. The application is in a runnable JAR. Double clicking on JPL-DownloadManager.jar launches the app.

3.3 Navigation Through the Download Manager

Video of navigations through the application

- *https://youtu.be/fi-LTpOALPs*
- *https://youtu.be/_JHrVrxeUvo*

3.2.1 Logging In

- *Note: There is currently a hard coded user since the backend for users is not setup. These credentials are "Username: jdoe, Password: password"*

- To log into the application click on the "Users" menu option on the menu bar.
- Click on "Login"
- Once login is clicked, the user should now have access to the encrypted files if they have access to view them and there exists any encrypted files.

3.2.2 Viewing and Editing Profile

- To view and change the user profile, navigate to "Users" in the menu bar and select "Profile"
- From this popup, a user can see their profile information.
- To edit a profile, click on "Edit Profile" and change the necessary information before clicking save.

3.2.3 Changing downloads path

- By default, the current downloads path is a user's local downloads folder.
- To change this, click on the "Settings" menu item on the menu bar.
- Click on "Download Settings"
- Click the button with the three dots to open a file system window.
- Select your desired folder and click "ok"
- Once back to the application, click on "Save"

3.2.4 Searching / Previewing / Downloading a file

- To **search** for an image, type a keyword in the search bar and click on the magnifying glass. At the moment the only search criteria are the words in the file name.
- To **preview** an image before downloading, click once on the item you would like to download in the database table.
- Once clicked, information should appear on the bottom pane of the application.
- To **download** a file double click on the file you would like to download from the database table.
- Once double clicked, the file should show up in the downloads table with the download status.

3.2.5 Pausing / Resuming / Deleting Files

- To **pause** a downloading file, click on the file from the downloads table. Once selected, click the pause button on the toolbar. Since files download in parts, the pause will not occur right away. The part that is currently downloading needs to finish first.
- To **resume** a paused file, click on the file from the downloads table. Once selected, click the play button on the toolbar. The file should start downloading right away and the status information should start updating.
- To **delete** a file, click on the file from the downloads table. Once selected, click the 'X' button. This will remove your file locally.
-

3.4 Uploading the meta-data to AWS Elasticsearch

3.4.1 Obtain needed files

- After setting up AWS Elasticsearch obtain the URL for the Elasticsearch endpoint and save in the first line of the file "es_link.txt" in the same directory as the script "scrape_es_upload.py".
- Save the LMMP API URL with username and key included in the file named "lmmp_api.txt".

3.4.2 Build the Elasticsearch mapping

- Run the script "mappings.sh" with the Elasticsearch URL endpoint as the command line argument.

3.4.3 Upload meta-data via web-scraping using LMMP API

- Run the script "scrape_es_upload.py". This should take about an hour to complete.

3.4.4 Test Elasticsearch API

- Run the command :

```
curl -XPOST '<YOUR ES ENDPOINT>' -d '{
  "query": { "match": { "_all": "LRO" } }}'
```

- If the meta-data has successfully been web scraped and uploaded the output should be all matches for LRO.
- Kibana is also a feature of AWS Elasticsearch which allows exploratory data analysis. The Kibana link should be available in

the AWS Elasticsearch web console. When using remember to change the dates which are being searched in the Dashboard setting upper right hand corner.

Section 4: Lifelong Learning

Each member of the group has given their learning experience below.

4.1 Abdias Andres

The knowledge and skills I have attained about the field in computer science do not stop once I graduate. As with any aspect of life, lifelong learning continues. This also applies to my educational experience in the field of computer science. Through this process I have learned various systems and tools. However, from working in my senior design project the most important tool I have learned to use is Git and GitHub. I learned this by asking my teammates as well as doing some research online showing the various ways to apply it into our project. By being part of this working project, I also learned important skills that can be transferred to potential jobs. This is one of the most valuable lessons I learned from learning.

Though, I learned about different tools and learned to apply it during the span of this project. What I felt the most important tool that helped was Git and GitHub. Git is a version control system that helps during the process of software development allowing teammates to keep track of changes in the project. GitHub allows developers to store their projects and also interact with teammates. The way this was applied to our project was to assign teammates with a certain task to complete on the project. For example, I was assigned to develop the settings option of the download manager, others were assigned to do user options and download options. Once we finish our part of the project, we used Git to push our contributions to GitHub allowing the rest of the team to view and download the new version. Reviewing if everything works, as it should. Once we find that the changes work we then merge all of the code into one main branch called the master branch. Where we now have all of our code put together nicely. The luxury of this tool is crucial in software development because it makes keeping the code organized very easy. Before I knew about this, for previous classes when we have a project that was group work we had to assign one person to put the code together. Everyone had to do their part then email the person with our parts then he had to manually go and put it together which is a hassle and makes its very disorganized. With this tool we can go back to a previous working point of the project in case the new changes had too many errors. For this reason, it was crucial for all of us to learn about Git commands and how to create our own branch in GitHub, so that the main project can move along effectively.

Furthermore, the process of learning Git and GitHub was not in one particular way. At first, I asked my teammates who had previously used the tools on why we would

use this for our project and how can it help. Once I knew the importance of the tool, I asked for some guidance on Git commands, which did help, but I found that I learned that command for a short time that caused me to ask again. Then I turned to Google and various other reliable websites and started looking up examples of when to use a command and how. By doing this I knew more about the meaning of the command and had an actual understanding of it. Before I just memorized the command, but learning and going into details, I knew more than just memorize. Then once I knew the "why" part I had to apply it; this is when I practiced using the commands. In GitHub, I practiced making my own branch once that was done I made a simple text file and tried pushing that onto GitHub using the Git commands I had just researched. Doing this repetitively I understood when and how to apply what I learned to the right situations.

Moreover, I learned that I am always going to continuing learning on my own, especially since I am now going to be done with school. Which will mean I will not be able to rely on professors help. During this class I was able to find an effective way to learn new technologies and that gives confidence when going into the professional field. I will be able to apply this with either asking a coworker or simply doing my own research applying it. Learning new technologies can be a bit intimidating at first but once you have an effective way to understand new technologies you gain confidence that you can learn it. Just have to accept that it takes patience and it will not happen overnight. Also, Git and GitHub is widely used by various companies. Knowing that I learned a tool on my own that is a standard at many companies gives me confidence. Hearing some advice from professionals already in the field, the most common thing they say is, you never stop learning. With the advancement of technologies growing at a rapid pace, its good to know I have process which I can rely one to learn new technologies.

In conclusion, in this experience I have learned an important tool, Git and GitHub. I have learned it is an essential tool that helps during the process of software development and is widely used by many professionals. I learned how to research information about a new technology and not be intimidated. In this case, it was Git and GitHub, which I did by asking my peers questions, researching this tool online, and applying it to my own projects. By doing so, I also learned the skills I need so that I continue to learn on my own (e.g., doing my own research and asking colleagues questions). From this experience I have learned an essential tool that I may encounter again in the future. I have also learned skills necessary that will continue to help me prosper in my continuing education and prospective job.

4.2 Rowan Edge

We're always led to believe that the single most important consideration in working with a group of engineers is whether they're technically capable of doing their jobs. The technology world calls itself a meritocracy, valuing knowledge and talent above all else. But over the course of this academic year, I've learned that every member of a team project must also take into account many other factors in order to help the rest of their team achieve its goals. Most notably, I found my social skills tested perhaps even more than my technical knowledge, as I learned to walk the fine line between effective guidance and counter-productive micro-management.

When we began working on this project last fall, I assumed that the single largest obstacle would be our general lack of familiarity with Amazon Web Services. That prediction could not possibly have been any less accurate. Due to the difficulties and delays we suffered in even getting access to Amazon's tools, we were behind before we even started, and had to strike all but the most basic functionality from our plan, which was so simple to implement that none of us really learned anything. In spite of the adverse timeline, however, I learned a huge amount about the lifecycle of group projects as a whole.

The first major life lesson I learned from my time on this project is blaming others for your own lack of progress is ultimately a waste of everyone's time. While we did eventually get access to AWS through our own CS department, rather than through JPL as originally planned, that access came far too late to allow us to implement most of the features we'd intended to include. As far as I'm aware, nobody blames my team under the circumstances, but the fact remains that we're unable to deliver a project meeting our own specifications. In hindsight, we—more specifically, I—should have recognized that simply waiting for JPL to deliver access as promised was folly, and that it was necessary to try to get the tools we needed another way. Had any of us realized this sooner, we would have been able to deliver a much more complete application; the next time I find myself facing delays beyond my control, I will be much more proactive in finding ways around the cause of those delays.

Regrettably, I think JPL's apparent indifference to our project was contagious. We all struggled to find the motivation to continue working on other aspects of the project that would be completely irrelevant if, as seemed likely, we weren't able to build a backend to tie the entire thing together. But from this I learned yet another life lesson, which I touched on in my introduction: there are many human factors to consider other than the immediate technical ability of one's teammates. This became increasingly important as the project dragged on with little in the way of progress. Our inability to even start work on the largest component turned into complete apathy toward the few things we were able to complete independently. I myself am guilty of that, but I think many of these motivational issues could have been largely avoided had any of us been more experienced in managing a team of not just engineers but people. It took us all far too long to think of each other by name and personality rather than by official position. As we got to know each other better, we became more comfortable talking about both this project and entirely unrelated things. Not only did befriending my teammates provide some intrinsic motivation to work on what seemed to be a doomed project, it also facilitated substantially more effective communication about the project itself. We

learned to work with one another as human beings, rather than just as engineers, and we and our project are both better off as a result.

Computer science—and software engineering in particular—has always been primarily data-driven and reliant on quantifiable metrics. But there is another component that is never taught in classes and only addressed by the most successful technology companies—which is perhaps why they are so successful—and that is the human element. My experience with software over the past academic year is largely irrelevant; the most valuable things I've learned have not been object- but people-oriented. It's a shame "social engineering" has such a negative connotation.

4.3 Mariah Martinez

Being part of the JPL Download Manager project was a great learning experience for me and brought a variety of new tools to the table. Some of the new tools that were introduced in this project were AWS Services (Cloudfront, RDS, Lambda, Glacier, and IAM), Apache Maven and AXET/Wget library for Java. These were the tools that were essential to making our project function efficiently. JPL currently has their data stored in an S3 bucket hosted by Amazon, so our group had to find a way to efficiently retrieve and allow the public to have access to that data. JPL is also using their own servers to handle to requests which would cost too much once this application is disbursed to the public. Using other Amazon cloud services like Lambda and cloudfront would fix these issues. To make calls to download the data, we would integrate the AXET Wget library for java. While two of our team members did have experience with such libraries, I found it necessary to take the time and learn some of these tools on my own. Cloud storage is the future and this was the perfect opportunity to dip into this technology.

To get a handle on these new tools, we had one of the more experienced members give the group a short presentation on the different Amazon services. Since we had a group of six people, we found it unnecessary and inefficient for everyone to be working on and learning the cloud tools. This is where we decided that we would split the team between cloud and GUI implementation. Being team lead, I worked on both teams and had to learn how to use cloudfront and the AXET/Wget library since that was the only thing that I was unfamiliar with and would have to implement. While the short presentation from my team member did prove useful in that we got an overall overview of what the different services of AWS were and how they worked together, it was not enough to start thinking about implementation. On my own time I slowly went through the documentation and tutorials that Amazon Web Services provide. A lot of these tutorials were more about getting started, running a database and hosting. What helped me the most was the fact that during my current internship, we just happened to be moving things around in our S3 Bucket, so I was able to ask a lot of questions and have first hand experience in the console. JPL was not able to get us access to AWS until Spring quarter and even then, I was not one of the ones who was given access. Therefore my only hands on experience was dealing with the console and command line at work. There was also a tutorial on Lynda.com that proved to be very helpful in learning the essentials of Amazon Web Services, specifically data services. I was not able to work through many of these tutorials since I did not have access to JPL's account but watching

the videos help my understanding of the way these services work together and the basic calls one needs to make in order to retrieve data. Learning how to use the AXET/Wget library was not difficult at all. The github page had many samples, one of which we directly used with our application. There still was a bit of a learning curve with setup, but Stackoverflow was helpful.

Although it seems odd to say, this learning experience taught me a lot about learning new things. I learned that learning new technologies takes time and dedication and an interest. Taking on a new technology means making a commitment to take the time and learn something from the ground up. Making a commitment like this will not seem like a big deal if it is something that interest you. Watching videos, going through tutorials and reading documentation takes a great amount of time. If the interest is not there and your heart is not in it then it just takes longer for you to get comfortable with the technology. Another thing I learned from this experience was that reading documentation and watching videos is not enough to learn a new technology. In order to actually learn something new, you need hands on experience with it. Luckily I had a little bit of that hands on experience at work, but it did not touch every service that we were using for this project. When learning things in the future I will definitely make sure that I follow along with the tutorial first hand so that I can learn more efficiently and run into problems that I need to learn how to fix that do not come up when watching or reading tutorials.

4.4 Gregory Miles

Before starting the senior design year long experience, I had very little knowledge about the cloud buzzword. Prior to senior design my knowledge of cloud services extended to the idea that servers are remotely provided for by a third party company. My general assumption is that one would SSH into the server. My senior design team, the Jet Propulsion Laboratory (JPL) Download Manager, made use of Amazon Web Services (AWS) to provide the server side services and infrastructure. We made use of AWS to provide for storage and content distribution of very large GeoTIFF images which are part of the JPL Lunar Mapping and Modeling Portal (LMMP). Additionally, prior to senior design my knowledge of web design and web based application level services was mostly limited to what I had learned in CS320 and CS120. By reading through documentation, listening to teammates and watching AWS youtube tutorials I was able to gain a much clearer idea of how cloud services actually work. On a more abstract level I learned about learning technical information beyond the buzzwords.

AWS provides much more than just remote servers to SSH into, they provide a whole range of services, all of which have Application Programming Interfaces (API) to allow automation via software of the services. Included in these services are AWS S3, AWS Lambda, AWS RDS, AWS EC2 and many others. AWS EC2 (Elastic Compute Cloud) comprises what I initially thought cloud services were about, they provide servers which one can SSH into and build services through. What I wasn't aware of before beginning senior design is all the other aspects of AWS, including the API and Software Development Kits (SDK)s that are provided. For example, with AWS EC2 one can use the API to control how long a given server is up and to spin up a new instance of a server

based on different events through the API. Additionally, there are services such as AWS Lambda, which provide microsecond priced computing, so one does not even need to take care of an actual server instance, which might include security issues such as SSH key safety or time needed to setup a environment, and only worry about uploading the code needed for a given event. While my team initially had a much more complex plan we ended up focusing on AWS S3 for storage services and AWS Cloudfront for a Content Distribution Network (CDN). AWS S3 would store the images and provide as the distribution point for the initial download from the client, and further downloads would be made from servers which have cached the download as part of AWS Cloudfront CDN.

To learn about AWS services, the primary place I read through was the various AWS documentation pages, while I was also assisted with by my teammate who had prior knowledge of AWS services, Rowan Edge. Rowan Edge was already very knowledgeable about AWS services prior to the start of the project and was the architectural lead along with being the main decision maker behind technology decisions. During team meetings Rowan Edge would discuss various aspects of AWS a great length, after the meetings I would look up in the documentation what he had discussed and what was required to implement the AWS technology discussed in the meeting. In general, I learned that by investigating the buzzwords and figuring out what actual services are behind them that it is a great way to build one's ability to discern the actual practical technological aspects of a project.

Learning new technical things requires day to day dedication and allocating time to go through documentation, manuals and specifications. Software initially made as a demo can turn into an advanced and complete software project. By building on a software project everyday what starts off as a simple piece of software base can grow to be very advanced. A prerequisite of the AWS side of the project was to go through the API documentation and online examples. It might seem that reading documentation is a waste of time and it is better to just start writing code. I have learned that documentation is very important and spending extra time going through technical documentation, specifications and standards can save time and provide a better end product. While going through documentation however, it is important to skim through the parts which will likely not be used, make note of interesting portions which might be used and spend a large amount of time on documentation which will definitely be used. Reading technical documentation and manuals becomes a skill which is vital to learning new technologies.

Additionally, I have learned it is better to cope with problems instead of complaining about them. For example, for our senior project we had many delays for various reasons. The main reason was that our customer, JPL was going to provide AWS access but was unable to. While this might seem like something to complain about and blame other people, it can also be looked on as a way to learn new things. I have learned that it is better to figure out ways to remedy technical setbacks instead of blaming people. For example, a code base can be built, documentation can be gone through, modules can be planned out in more detail and planning out exactly who will be responsible for what can be done without requiring any external help. By the time I realized this however, our team had spent a great deal of time waiting for the sponsors to help setup the AWS access. I realized that it would have been much more productive if we had started on

building the other parts of the project than on blaming the sponsors. My main lifelong learning takeaway is that on technical projects it is best to get started doing as much work as possible as soon as possible even if there are resource setbacks.

Buzzwords are often very powerful technologies but it is important to understand what the pros and cons of those technologies are. For example, AWS cloud services are very useful and can be made to quickly build scalable web services and can provide the back-end for many software services which otherwise might require large amounts of infrastructure to be built. I learned that AWS has many APIs which abstract away from the infrastructure beneath and provide a developer with a quick and easy way to integrate their software with a AWS service, such as building a API by using an AWS API endpoint and AWS Lambda. Cons to AWS include that it might be difficult for some organizations to reconcile AWS with their security and funding requirements, the long term pricing structure might not appeal to some organizations and AWS APIs can end up in "vendor lock-in". Vendor lock-in happens when a convenient API is included in a large software project and there is a sudden price jump in the cost of the API service but it is unfeasible for an organization to modify the software to not use the API. Another con of cloud services is that an organization might prefer to use preexisting infrastructure instead of cloud computing. I have learned that while there are many pros to using cloud services keeping these cons in mind is important when designing a large software system and it might be feasible to include a back-up design in the plan. For example, it might have been feasible for us to have included a design which did not use cloud services, particularly when there were resource setbacks. I also learned that going through the documentation with a mind to solve the problem at hand will save time in the long run. Additionally, I have learned that it is best to start working around setbacks instead of letting setbacks set the schedule. I am now confident that I understand cloud services and particularly AWS services. I can understand the line that divides the buzz and the marketing of the buzzword.

4.5 Adrian Rendon

Four to five years of going to school at a University, in the field of a major such as computer science, can leave a senior student like me with a lengthy array of things they learned. Some concepts or technologies are more important than others but, nevertheless, I value everything I learned throughout my years of school here at Cal State LA. From the basics of web and Java programming to the complicated and painful world of Computer Graphics, I value it all. Of course there are a couple technologies in the world of Computer Science that I have learned and find to be more important than others. Out of all the technologies I deem to be more important, I would have to say that the most important was being able to understand and use Git and GitHub to manage source code for my school and work projects. The understanding and importance of these two technologies was further stressed thanks to the senior design project.

Git and GitHub are two technologies that are different but at the same time can go hand in hand with each other. Git is a revision control system, which basically means it is a tool to manage source code history. GitHub is a hosting service for Git repositories; it offers all of the distributed revision control and source code management functionality as

Git in addition to adding its own features. I never thought it would be such an important tool. The only reason I took it upon myself to learn how to use them is because I realized that many developer companies use Git and/or GitHub to manage their software. It was not until I used it in a couple of my classes, at work, and Senior Design that I realized how useful and powerful these two technologies can be. One clear example of this for me was when my senior design group and I were using it to create the main GUI component of our project. Our team lead created the "skeleton" of the GUI and delegated the rest of the work to three of us, myself included. Among us three we split the work evenly and decided to each create different branches of the main GUI skeleton and add our own work in there. When we finished, our team lead merged all three branches together and just like that our GUI component was almost nearly done. When I pulled the master branch onto my computer and ran the program, I was amazed to see how the features I added were all there along with what my other teammates added on their own. Furthermore, I was amazed by the fact that the process of merging our three branches together was really quick and easy. This was when I realized how important of a tool this was not only for school other school projects but also why many developer companies rely on this technology to manage their source code. At my current work I created a ticketing support form system along with administrator page that manages it. I originally created it without the use of GitHub and, if I had to be honest, it was hard to keep track of everything I changed or fixed. After I realized how important these technologies are, I decided to put my source code in a Git repository and manage it through GitHub. Now every time I add a new feature or make some changes to the system, I use branches to manage the changes I have made to the source code. It helps to keep me organized and at the same time makes me feel secure about changing the source code that currently works.

As I stated earlier, I never thought Git and GitHub would be as important as I now believe they are. I first heard about those technologies my second year during my CS203 class but never thought much of it out of ignorance. It was later brought up again by one of my close friends during the Spring Quarter of my third year. I made an account at that time but never used it or thought about using it until the following Fall Quarter of my fourth year and the start of senior design. It was during this time when I realized I should put forth more effort to learn it for my own good and for the good of my senior design project. At first I found it very confusing. The brief online research I did on it didn't really do much to help me so called on my friend to help me learn how to use Git and GitHub. He had already been using those technologies for about two years so he knew his way around how to use the Git commands on the terminal and how to incorporate the use of GitHub. With his help I was able to get a proficient understanding on how to use these technologies and, more importantly, when to use them.

Learning new things is generally exciting, at least for me personally. When it comes to learning something new I learned you need to have the motivation to take the time and actually learn whatever it is you want to learn. If you say you want to learn a new technology but lack the motivation to follow through and learn it, then you will be in the same state you were before. When you have motivation to learn something new then you have to be able to use all the resources you have available to help you learn. This should not be limited to self research that can be found when using the internet because it does not always have the best or easiest answer. Sometimes it can be easier to rely on

asking help from fellow peers. In my case I asked my close friend for help and I also asked for help from my team lead. It made learning how to use Git and GitHub much faster and easier. I honestly believe this approach is the best when it comes to learning new technologies. I had a job interview not too long ago and was asked how I would go about learning a new language. I more or less stated the same thing I did here and my interviewer later told me that is probably one of the best ways to go about it.

I learned about a few new technologies because of senior design. I learned about JavaFX and the many services that Amazon Web Services provides. But as I have stated, Git and GitHub were the most important for me. It is something every serious computer science major looking for a real life job should have a proficient understanding of. I will use everything I have learned throughout my years at school and the real life experience given to me by my senior design project to help me improve my own skills and have a healthy approach to learning new things.

4.6 Kevin Tu

I have learned so much throughout the year in Senior Design. But what I've learned in the way of software tools have been helpful and I would imagine be useful in future projects I encounter. The main tool or software library I've had to use during the year has been JavaFX on E(fx)clipse. It is similar to Java to which I was familiar with given all the courses I have taken over the years. JavaFX is a library that was created to specialize in creating graphical user interfaces for desktop applications. Another tool was GitHub where everyone could merge their code in an efficient and effective manner. I was not entirely comfortable in using github as at the time it was pretty new to me, however, it proved to be very useful. Communication through Slack and Google Hangouts were extremely helpful in coming up with plans for the project.

Starting with JavaFX, learning it was simple since it is very similar to Java. There was even a simple to follow tutorial on the JavaFX website which made the transition even easier. The tutorial started with the windows and progressing to tables, buttons, and pop ups so it was a great start to learn before creating the GUI which involved most of those elements. The main thing to learn was its large pool of available classes for each part of the GUI that we needed. Classes for features such as the tables, the text boxes, and the assortment of buttons were full of customization options. Choosing the right classes that would be adequately useful for our GUI required a Google search and trial and error during implementation. Sometimes I would look up online videos to get a better visual understanding of the classes and the capabilities of JavaFX.

I programmed using E(fx)clipse which provided JavaFX tooling and framework for the Eclipse IDE. I have used Eclipse for most if not all of my Java programs over the years so I was most familiar with its environment. Overall making the start of the project's coding much easier to begin. Since coding was not on completely new grounds

or concepts, it did not require as much learning as picking up a new programming language.

Github was a tool that I recently started using going into the project earlier in the year. I have not found myself using it as much before since team projects were infrequent and often did not rely on coding as a big part of the project. Opposed to manually syncing up code line by line, GitHub definitely makes it less of a hassle. However, I honestly found it tricky to work with at first. The terminology and certain functions were a bit vague so I could not use github to its fullest. After many attempts I stumbled upon EGit where it is the integration of Git for the Eclipse IDE. In one place, I was able to push code to and pull code from my team's repository within E(fx)clipse.

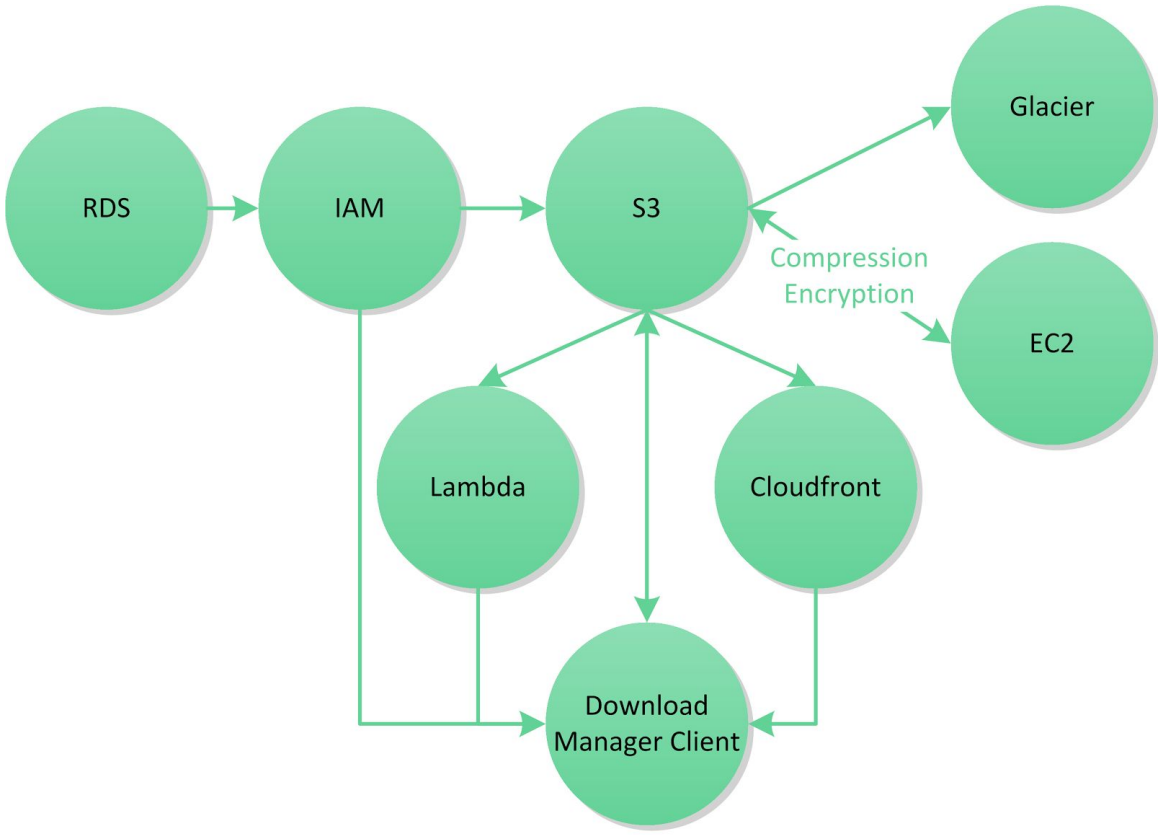
Communication is important between the team and between clients and advisors. The tools, Slack and Google Hangouts definitely made that much easier than being in person or back and forth emails. Slack was new to me but with much needed communication at all times, it was immensely convenient. Notifications on my phone made sure messages were available to me right as it was created. It was relatively simple to setup with no hassle. Next thing I know the group was able to communicate whenever and wherever with a connection to the internet, by phone or computer. I was new to Slack and the team basically introduced me to it. I can see myself using it for future projects or among other things that would require quick communication. Google Hangouts was our communication with the liaison and advisors for the Spring Quarter. Another simple way to communicate across pretty much anywhere and with its voice chat and face cam features it created a quick way to simulate an in person meeting. Google Hangouts was nothing new to me, but Google applications including Google Hangouts were also extremely useful. Google Drive, Google Docs, Google Slides, Google Calendar were used together and provided an easy work environment. I was familiar with these Google applications so again there was no real learning to do other than using them to complete our assignments. But similar to GitHub it allowed us to work on written portions of the project like presentation slides and documentation together at the same time on the same document.

What I learned about learning during this project is that it could either take a long time or a short time to learn certains tools and libraries. I would imagine if I had not learned Java over the years JavaFX would have been troublesome to use. But having learned Java played a big part in why it was used in the first place. Although obvious, the more documentation and tutorials a tool has the easier it is to understand to use it. JavaFX had plenty on its own website and tutorials expanded even further thanks to Youtube where user create their own tutorial from the basic to the more niche features of the library. EGit had a more vague tutorial and its user base from what I understand was not as large so other tutorials were far and in between. I was able to work with EGit however, I feel like I could use it more effectively if there was a more clear tutorial. As of now, I

use some of its features that I am able to understand and kind of workaround certain aspects of it. Learning is important to grow as a person and further improve one's efficiency in their work.

This project has had me learning many things from tools, software libraries, and among others involving working in a software project with a team and clients. I can definitely say that what I have learned throughout the year will be used to improve myself and my future endeavors in the field of Computer Science.

Section 5: Architecture and Design



The main components of this Download Manager, shown in the DFD above, each belong to one of three major categories: security, storage, or distribution.

Security
Security and permissions, for both users and files, are managed and enforced by IAM. RDS is used to store metadata and encryption keys for each file; access to RDS is also enforced by IAM.

Storage
By default, all files are located in S3. EC2 can be used as necessary to compress and/or encrypt files in-place. If a file is not accessed frequently, it can be archived to Glacier to reduce costs.

Distribution
Files cannot be served directly from S3; a CloudFront distribution ensures that any download request will be checked against the local edge node's cache; if the file is not already in the cache, it will be added during the download. Lambda is used for all other communication between the client and the Amazon services.

REST API
The meta-data REST API was accomplished via AWS Elasticsearch (Not shown in diagram). All meta-data was web scraped from web pages accessed via the LMMP API and uploaded to AWS Elasticsearch. Elasticsearch provides advanced query capabilities and data analysis.

Section 6: Conclusion

Just like many other senior design groups in our graduating class and all the ones before us, we as a group faced many challenges throughout the course of this Senior Design project. The requirements for our JPL Download Manager were many and essentially each were a challenge for our team in one way or another. Now that we have reached the end of this project, we can look back at all that we accomplished and all that we unfortunately could not accomplish. We now have the insight all the lessons we learned throughout this project and be able to say what could have been changed or added to the project to help make it more complete.

Our main goal was always to be able to download a file through a type of download manager. And we succeeded in accomplishing that main goal. We were able to create a visually appealing GUI that is able to pause, resume, and stop downloads. It is able to handle filtering of items listed in the database and the ability to handle editing individual user accounts. Requirements we failed to produce would include automatic scheduling of downloads, the integration of individual user accounts through AWS IAM, encryption of specific files, and the ability to limit download speed in order to prevent the use of the entire available bandwidth.

There is no way to mention the requirements we failed to produce without mentioning the challenges we faced. Without a doubt, the most difficult challenge we faced was the amount of time we had to wait to gain AWS access. Not having AWS access really slowed down the productivity of our team during Winter Quarter. We were on schedule the first 4 to 5 weeks where we spent ample time completing the GUI component of the JPLDM. Once we got AWS access, only 2 of our 6 members were able to access it, and only 1 of the 2 actually had the full ideal permissions needed to test our project. Among other challenges, a common challenge we all faced was being able to figure out times we can meet given our busy school and work schedules.

All in all, regardless of challenges faced, there many important lessons we as a team learned because of this senior design project. The main lesson being that delays in a big project can happen. As a team, once we were able to gain AWS access, we worked as best as we could to do as much as we could before our deadline date. Although we were not able to implement everything we proposed we were able to experience something that

can frequently occur in industry. And that experience is one of the most valuable real world experiences gained because of this project. Thanks to the experience we all gained from working on the JPLDM we are more prepared to put these experiences to use when we begin our professional careers as computer scientists.

Section A: ACRONYMS

AM	**Authentication Module**
AWS	**Amazon Web Services**
CDN	**Content Distribution Network**
DMC	**Download Manager Client**
DM	**Distribution Module**
ECM	**Encryption/Compression Module**
JPLDM	**Jet Propulsion Laboratory Download Manager**
RA	**Rest API**
SM	**Storage Module**
LMMP	**Lunar Mapping and Modeling Project**