Team Leads

First off, take a deep breath and get ready to read and learn a lot of new things. Take it 1 step at a time and meet your team. We had 10 people, so we split everyone up into 4 teams. Team lead role is to look at all the stops and plan/manage similar to a scrum master. There's a lot to do such as actually working on the telescope and troubleshooting. #1 was just setting a weekly schedule to meet with both liaisons and advisors. Play to people's strengths and weaknesses and where they want to go. We spent 1 semester researching, reading, and preparing sprints, and the 2nd semester working on the code and keeping track on Jira. We saw the previous team use a Python Virtual environment, and it didn't work properly so we just swapped to Anaconda with explicit instructions, (Note the YAML file). Take a realistic expectation and try to chip work on a portion of the project. Also as team leads don't forget to work on where they need a body like I jumped into Telescope Integration by helping set up the Telescope and working on hands-on experience with the hardware/software. If someone needs help, you go help them and don't be an ass or make conflicts. When it was time to graduate, we had to merge all files together and test run it. Don't forget to take time to look for an internship/job.

3D Modeling:

To the next team focusing on the 3D modeling, I'd like to note the following:

Last thing we did in regard to 3D modeling was fixing the camera angle toward the nearest point on the Moon using an API provided by JPL (Please refer to *CSULA Telescope Project API Calls*). Second, we left comments on the code we wrote to leave information on the approaches we took to either add or improve 3D modeling features. In addition, we optimized the code for the side menu that allows users to toggle between data layers. We were also attempting to extract Exif data from a picture so that we could more conveniently process the data without having users fill out HTML forms, but couldn't implement this feature due to time constraints.

If you took the computer graphics course, chances are you used WebStorm or Visual Studio to write your code. I would recommend using one of these programs because they highlight the code, making it easier to read and understand the purpose of the code as opposed to PyCharm, which doesn't highlight the JavaScript portions of a code file.

Next, I'd recommend also getting a picture of the Moon, while also noting down the time and location (lat, lon) of the screenshot in case the picture doesn't have that information available in the form of Exif data. We ran into issues during the process of making sure that the images we took had the data inside the image so that we could pass them as parameters for the generation of the 3D models. Embedding this data into the image is an important step in achieving Exif data extraction.

When replicating the view of the Moon at a given geo location and time, we saw that the 3D render wasn't as accurate as we wanted it to be, so we considered aligning the 3D models of the Moon and Earth to get better results. For aligning the models, we were going to use quaternions, but time constraints created a roadblock for us to learn the implementation, so for the future team, we recommend researching quaternions and to consider using them if the 3D render of the Moon isn't an accurate replication of what is provided in the user's screenshot.

User Interface/User Experience:

To the next team focusing on User Interface:

While we were able to make a lot of improvements towards the User Interface/Experience, there were a few shortcomings with our overall goals for the project. The main issue to keep in mind for this year is the mobile implementation. Unfortunately for us, we learned a bit later on in the project that all the CSS for the application as a whole was on one page, meaning we had to deal with over 11,000 lines of code. What we recommend is to take a bit of time to recognize what portions of code are needed for each page and try to splice it in a way to allow for each page to have it's own CSS page (EX: About.html would have its corresponding About.css, etc… for every other page)

We had originally tried to attempt to do this ourselves; however, we began trying to separate everything too late in the project's lifecycle, thus we were unable to get any progress towards it done. We did have to add new CSS in order for the newly implemented about page to work as well as the Connect page for the Telescope Integration team, however that was all added to the bottom of the css page and commented accordingly. If this team is unable to splice everything to work properly, at least try to understand what everything does and comment on anything that you feel is important.

While reviewing our implementation, you will see inline bootstrap. We were trying to stray away from this because it makes it difficult for future teams to understand. Taking out the bootstrap and using strictly CSS was our goal. However, we were not able to do so due to time constraints. You will notice that the layouts in each page are in a grid format. There are features in Connect.html that need to be reviewed. Keep in mind that this page is unfinished and ready to be built upon. Lastly, read up on the CSRF token. This is a crucial piece in the integration of Django and the html pages.

Telescope Integration:

Our work was done on the Connect page which holds a container meant for the live feed from the telescope. Currently, there’s a placeholder image of the moon which you can screencap, save, then upload it with the right container which allows for image input. We had planned for a way to display the live feed directly to the Connect page, be able to take a screenshot of what is being displayed, and have a button that directly uploads to the MoonTrek DB and grabs that last uploaded image to navigate the user to the registered image. The problem with this approach is that we could not figure out how to obtain the live feed from the telescope we worked on in a way where we can add it to the Connect page. Additionally, we could not have a button grabbing the last uploaded photo from MoonTrek’s uploaded images because of a built-in security issue. From what we researched, this has nothing to do with the implementation, it’s just how the code and file manager works to prevent code from being able to access an application’s files; a security risk.

What we decided to do is leave a placeholder image representing the live feed for you to implement. We had to figure out a solution to working with the telescope after running into issues, which meant we had to put the live feed display functionality on the backburner. We used other software and hardware to display the telescope view, but capturing a photo or video would have to be done aside from MoonTrek rather than through MoonTrek. We would recommend finding a better way to view the telescope’s pov in such a way that it can be displayed onto MoonTrek. So now let’s talk about the telescope and how we ended up working with it to view what’s being displayed.[]

what you recommend for next year's team
- -closing remarks
- -what things you would like them to know on what to build upon
- -Things you didn't get to and list them
- -ideas you couldn't finish

<u>Image Registration Processing:</u>

To the next team focusing on Image Registration Processing:

After removing some lines of code from the views page there should be nothing outside of the imageProcedure page. All of the process that makes up the image registration process is in that page inside of the processUserImage function. The function can be divided into two parts. The first part is the circle detection. In this section, you are trying to look for circles inside of the image given to you. We pick the largest circle as our the one used for the rest of the process. There is some improvement that can be done in this section. Specifically, trying to add tresh holds to allow for some images to process, image 88. Another thing is the opening and closing, I believe this got cut due to time but it can help with some images in the circle detection process.
Code for threshold and opening/closing

```
thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
```

```
kernel = np.ones((25, 25),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
```

The other are of improvement would be the image registration. From our point of view, there is not a lot that can be done here but you can try to implement FLANN or AKAZE instead of SIFT to see what you get. The issue with those is that it can be hard to integrate with the warp portion of the code.

The thing that needs to be a focus to improve the image registration is the model_3d function and what comes with it. Since the other group in the 3D model manage to create the model of the Moon and Earth, you can now try to connect their work with the image registration process in order to really increase the probability of an image being register. We currently only use one image of the Moon in different sizes as our base for all comparisons but with the 3D portion of the project we might be able to get more images that can be use as the base.