System Overview:

The Nebula Net interactive Feed hosted website on local servers is a website and supporting software that allows users to view up-to-date JWST photos and the completed mission compiled photos. This is done with two web pages. The first is the home landing page that will display the last mission photo taken from JWST. This page will have a list of informative information and the current mission this photo is a part of. The second page is a mission timeline that will provide the mission information from the NASA JWST website. These user interactions as well as the user to download and view photos from the JWST are done through a limited user interface so all interactions are click-based and not text-based. The website will be run on a React software framework to allow flexibility in hosting on different devices. The set of Processing modules

Software Architecture:

NebulaNet Component Architecture

Space User Interface (SUI) (The displayed website):

Provides and displays photographic, mission, and informative data from the official James Webb Space Telescope (JWST) Database and website hosted by the National Aeronautics and Space Administration (NASA). This allows users to view the mission photo of the day that is part of a larger mission that is used to create the large and compiled beautiful photo that most are familiar with. The website will display only the current mission photo set and the mission timeline with the compiled photos from each completed mission.

Image and Mission Processor (IMP) (The information gathering)

This Component is broken down into 3 modules that will be discussed in detail in the software module section. This Component is responsible for fetching the photos, mission, and informative data that is listed on the NASA JWST database, and collection of the mission timeline.

Photo Converter - Fits to Jpeg (PC-FJ)

Mission to Image Information Packaging (MIIP)

Mission Information Gatherer (MIG)

MAST Validator (MASTv)

Web Hosting Architecture

AWS Components

Software Modules:

Space User Interface (SUI) Module:

Role and primary function:

This Module is responsible for the implementation of the website's landing page. Starting at the header section of the page, the header will contain 4 main elements: The name of the webpage (NebulaNet), and links to the Calendar, About, and Sources pages. Next, the homepage will display the corresponding daily James Webb telescope photo directly below the header. This image will be rendered at full size in color so that the user can easily find the daily picture. Towards the bottom of the page, a small preview of pictures taken within the previous three days will be shown at the bottom with another button link that will take the user to the Calendar Gallery.

Interface to other modules:

As index.html will be the landing page, this module will be the foundation for connecting all other modules within the software. All backend data will be called and displayed on the screen in an aesthetically pleasing manner. Links to all other pages within the website will have links embedded within the landing page as well.

Static model:

Dynamic model:

Design rationale:

The design rationale for the landing page was to ensure that the daily observation from the James Webb telescope was the main feature for the user to see upon entering the site (as that is the site's purpose). Next, a body of text is located below containing the Name of the object being seen in the photo, the date when it was taken, and a brief description of how the photo was taken and what mission it was taken during. This will allow the viewer to take in the photo and its beauty and once they are done observing, the user can read an explanation of what they just viewed. As the user scrolls down the site, a small preview of a set of photos from previous days will be displayed next to a button that will redirect the user to the Calendar module which will display all photos in a calendar pattern. The goal of the landing page is to be easily navigable and ensure that the main feature is highlighted and dominant.

Photo Converter - Fits to PNG (PC-FG):

Role and primary function:

This module is designed to convert Flexible Image Transport System (FITS) files, which are the standard format for astronomical data into PNG (Programmable Network Graphics) image formats. It uses the astropy library to read and process FITS files. It then utilizes matplotlib to convert the

FITS files into a 2D array where each element is a value that corresponds to the pixel's shading. These numerical values are further translated into gradations of shading, which renders them into detailed grayscale or color as a PNG image.

The MAST Query serves as an intermediary between the local file system and the Mikulski Archive for Space Telescopes (MAST) data. It gives access to a wide range of astronomical data hosted by MAST. It performs data queries into the MAST database using parameters such as target name, observation iDs, or dates. This module uses the astroquery library to execute these queries, which handles authentication, data retrieval, and error checking.

Interface to other modules:|

This module interacts with SUI to retrieve the data requested for the current mission.

Static model:

Dynamic model:

Design rationale

Mission to Image Information Packaging (MIIP):

Role and primary function

Interface to other modules

Static model

Dynamic model

Design rationale

Mission Information Gatherer (MIG) Module:

Role and primary function:

This module is responsible for managing and collecting requests from the James Webb Space Telescope website. It collects data specifically from the observing schedules data and saves it to a list. The data on the list is then written to and output to a json file.

Interface to other modules:

The design rationale for the web scraper is to scrape the current data from the observing schedule. The observing schedule data can then be used for the landing page and for information about previous images that have been taken by the telescope.

Static model:

Dynamic mode:

Design rationale:

The design rationale for the web scraper is to scrape the current data from the observing schedule. The observing schedule data can then be used in other modules. For the web scraper, Beautiful Soup was not needed because the URL is a .txt link. Instead, all that was needed was requests. Once getting the requests, the web scraper just needed to convert it to .text, that way it could be appended to the list and returned. Once it is returned, another function writes this returned data to a JavaScript Object Notation JSON file and a third function converts it to a Python list.

MAST Validator (MASTv)Module:

Role and primary function

Role: The MAST Validator Module acts as a verification mechanism for both the web scraping and MAST API modules to determine if the MAST API library is functioning as intended for the current expected implementation and to verify that the JWST observation website is online. It accomplishes this by trying several different API requests by importing the MAST API Module and running defined unit test cases. There is also a separate test to verify that the JWST observation website is online before the web scraper runs.

Function: The MAST Validator is a part of the system admin tools and runs during initial website setup or reset. It's intended to verify that the scheduled back-end Python programs work, but also to detect any changes that may have occurred resulting in bugs and issues, whether that be changes to the code implementation or the website/python library dependencies breaking or changing significantly in how they function.

Interface to other modules

The MAST Validator directly imports the MAST API Module Python code and runs several unit tests using the imported functions. There is no direct communication with the other modules, as the MAST Validator module only runs during the website setup or reset process.

Static model:

UML Class Diagram for MAST Validator

Dynamic model (Sequence Diagram):

UML Sequence Diagram for Start Shell Script

UML Sequence Diagram for Start Shell Script

Design rationale:

Choosing Nose extends the default unit testing functions that Python provides. A majority of the group has prior experience with using the testing library from previous computer science courses. The Library also supports easy shell scripting and coordination if there are multiple different tests to run. Because the MAST API module and WebScraping module are written in Python, the MAST Validator should also be written in the same language to avoid programming language incompatibilities. The reason for the modules' existence is to verify that the functionality of the current module works as intended, help debug issues during the implementation step of the software design lifecycle process, and to detect any issues with dependencies that the project relies on such as libraries or websites/API's.

Extra Module Placeholder (if needed):

Role and primary function

Interface to other modules

Static model

Dynamic model

Design rationale