Software Design Document

for

Operationalize Networked Collaboration Features for Moon Trek

Version 0.1

Prepared by: Sean Chung, Aldo Gil I, Tommy Lay, Allen Marquez, Tam Nguyen, Alex Sahakian, Andy Tsan, Srivats Venkataraman, Jian Wu, Anna Yesayan

Sponsored by NASA JPL

Dec 3, 2021

Revision History

Name	Date	Reason For Changes	Version
Sean Chung	11/6/2021	Inserted all the information for section #2	0.1
Sean Chung	11/7/2021	Architected the layout for section #3 and began inputting the first half of the data according to the previous SDD.	0.1
Srivats Venkataraman	11/8/2021	Added Section 6.6 - 6.7	0.1
Tam Nguyen	11/10/2021	added section 5-6.3	0.1
Allen Marquez	11/10/2021	Added part 2 of section 3	0.1
Alex Sahakian	11/10/2021	modified section 8 and 9	0.1
Srivats Venkataraman	11/11/2021	Completed section 1, Introduction	0.1
Anna Yesayan	11/11/2021	Added to section 8-9	0.1

1. Introduction

1.1 Purpose

This document will focus on the software design used to build the collaborative features for Moon Trek Lite (MTL). The purpose of the collaborative features is to allow users to share in real time exploration of planetary bodies using actual data from the Jet Propulsion Laboratory (JPL).

1.2 Document Conventions

- The title for each section is Times New Roman, with font size 20.
- The subtitle for each section is Times New Roman, with font size 14.
- The body and bullet points for each section is Times New Roman, with font size 14.

1.3 Intended Audience and Reading Suggestions

This document is for project managers, developers, users, document writers and people with some background in computer science. This includes staff, faculty, advisors, NASA JPL liaisons. The recommended sequence for reading is start with the introduction then move to a topic the user is interested in.

1.4 System Overview

This software is identified as the Networked Collaboration Features for Moon Trek (NCFMT). NCFMT shall provide collaborative markup of 3D solar system terrain such as the creation of waypoints, rapid navigation toward waypoints, text annotations, and freely drawn “ink” annotations on the new Moon Trek Lite application. NCFMT shall also provide “rooms" that let the user markup and share different states of 3D solar system terrain simultaneously, communicate via a text communications system, and create waypoints for rapid navigation. Upon release, NCFMT shall be completely open to the public and be used for scientific research, mission planning, educational purposes, and general exploration.

2. Design Considerations

2.1 Assumptions and Dependencies

- Consumers of the software are expected to have consistent access to a stable internet.
- Consumers are also expected to be familiar with using the internet and the basic input devices of computers: mouse, keyboard, etc.
- Consumers may need to install the WebXR Emulator extension since certain browsers may not simulate the WebXR API. The extension allows users to run WebXR content without the need to use an XR device.

2.2 General Constraints

- **Hardware Limitations:** Since web browser may need to render a large amount of graphics, users will need to get a more powerful GPU component if they wish to run the software smoothly.
- **Network Constraints:** Users will need a stable internet connection because WebSockets drowns in performance during high latency.
- **WebXR Limitations:** While WebXR does provide users with VR capabilities, web browsers may struggle to keep up with the software because of a lack of stable/unstable technologies.

2.3 Goals and Guidelines

Because our group is carrying out the AGILE process, we focus more on usability rather than readability. As a result, we may be sacrificing good documentation to create more prototypes and usable software.

2.4 Development Methods

The project uses the AGILE development model. As a result, we perform weekly SCRUM meetings with our advisor and the JPL team via ZOOM to relay and receive information. We are organized into small teams that can continuously put

out new features and updates. Additionally, we can also output new changes based on demand.

3. Architectural Strategies

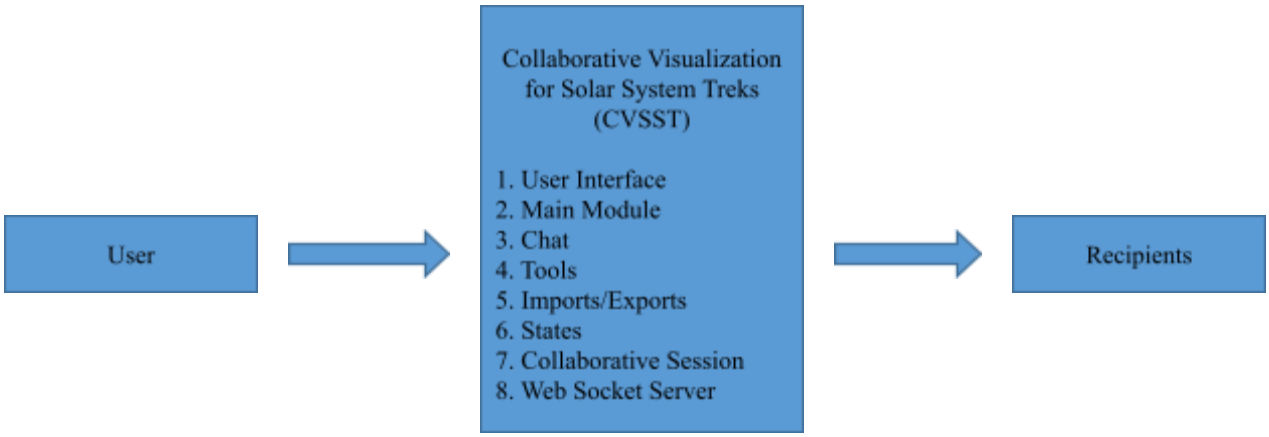
- Programming Languages
    - HTML/CSS
    - Java
    - JavaScript
- Database
    - Types of information used by various functions
        - Coordinate Data
        - HTTP/JSON Requests
        - Mesh Data
        - Model Layer Data
        - Session Data
        - Text Data
        - WMTS Map Data
    - Frequency of use
        - On every host request in the session, data is pulled.
    - Accessing capabilities
        - Database access is done according to the current session host and members.
    - Data Entities
        - Solr
        - ArcGIS
            - Accumulo
        - Oracle
            - Postgres
- Library
    - Bootstrap
    - Express
    - WebSockets
- Reuse of existing software components to implement various parts/features of the system.
    - Reuse of existing software

        - MoonTrek
    - Components to implement
        - Sessions
            - Collaboration in session rooms
            - Textbox communication
                - Users can communicate with each other through the system via text.
            - List of attendees in the current session
            - Session authentication (Password Entry)
                - The host has the option to create a password for visitors to use to enter the session.
                - The host has the option to turn off/on the password feature.
            - Session States
                - The software will keep track of both present and past layers along with entities from the on-going session. (Session Owner Privileges)
            - The software will give the users the ability to become session admin, which allows them to determine what participants can and cannot do.
                - Admins can change the session's current layer/entity.
                - Users can only use tools such as polyline or annotations on layers/entities.
- Future plans for extending or enhancing the software
    - Implement a software update giving users capabilities to render 3D models
- Error detection and recovery
    - Input Validation
        - The system will validate session passwords
            - If the password is incorrect, it will prompt the user with an error response.
- Memory management policies

    - Data retention policies
        - Data will be temporarily saved during each session
        - The system will allow users to revisit previously used data from the current session
- External databases and/or data storage management and persistence
    - Not applicable
- Distributed data or control over a network
    - A java servlet will be used for WebSocket connections
    - A separate server will run a database connected to the java servlet
    - The java servlet will be run on a Tomcat server
- Generalized approaches to control
    - SSH access to the java servlet servers
- Concurrency and synchronization
    - Java servlet for the WebSocket server is asynchronous in that it can handle multiple WebSocket connections in parallel
        - While the WSS connection is active, synchronization between the client and web server shall be achieved with serialization and processing
- Communication mechanisms
    - WebSockets
    - HTTP Requests
- Management of other resources
    - Not applicable

4. System Architecture

Figure 4-1. Context Diagram, DFD Level 0



On surface-level, the software application must receive input from a user then respond by fetching the correct modules. The recipients would receive the information delivered from the modules. The system was architected so that each module represents a core feature in the user experience.

4.1.1 The User Interface Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.2 The Main Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.3 The Chat Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.4 The Tools Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.5 The Imports/Exports Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.6 The States Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

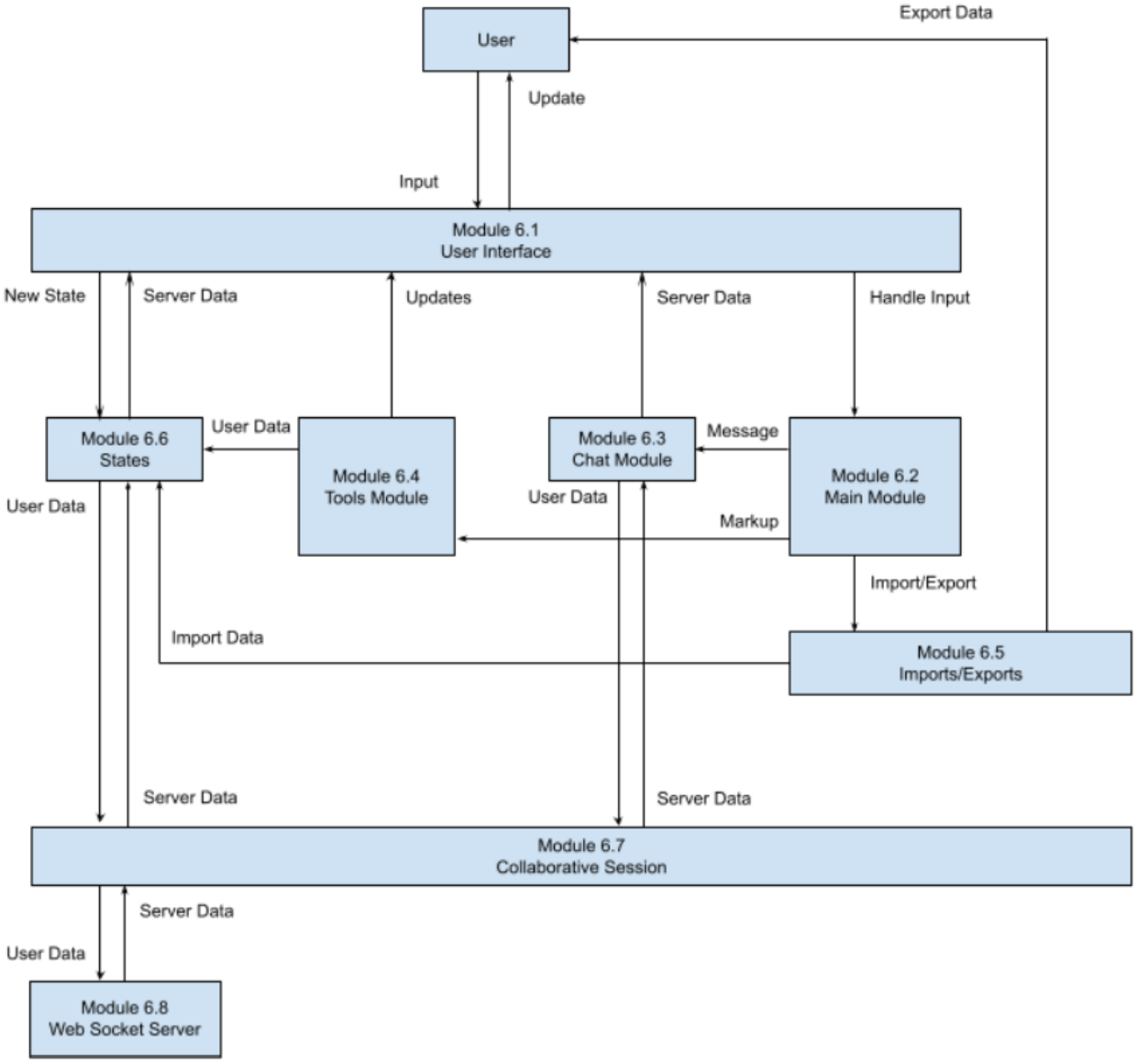
4.1.7 The Collaborative Session Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

4.1.8 The WebSocket Server Module

Check DFD 0 or DFD 1. A comprehensive explanation can be found in Section 6.

Figure 4-2. Level 1 DFD



5. Policies and Tactics

5.1 Specific products used

- IDE: Visual Studio Code
- Database: We decided against using MongoDB. There could also be a better database and database design than MongoDB. A temporary solution is to have users export data occasionally. A database will be chosen later. Refer to Section 8 for more details.
- Library: Dojo Toolkit, Express, WebSockets, Bootstrap, CesiumJS

5.2 Requirements traceability

- Requirements are often discussed in meetings with JPL. After coding and testing, the team goes back to the SRS document to see what requirements have been met. The requirements not met will be brought up with JPL.

5.3 Testing the software

5.3.1 Testing the features of the modules

- Sessions
    - Testing max capacity of session set by host.
        - Display error messages to users attempting to enter a full session.
    - Test the various platforms the software supports.
        - PC, Mac, mobile
    - Test input textbox
        - Have clients spam textboxes and provide invalid input.
            - Discover limitations to be fixed and debugged.
    - Test Administration privileges
        - Ability to control what users can do in a room.
        - Go back and view past layers/entities.
            - Test limitations of memory.
            - Speed and consistency.
        - Ability to remove users from a session.
        - Ability to swap admin roles between a host and user.

5.3.2 Engineering trade-offs

- Sessions do not currently support VR/AR capabilities.

5.3.3 Coding guidelines and conventions

- Coding Guidelines
    - Code must have uniform indentations for readability.
    - Functions must have a comment at the head describing their purpose.
    - If possible, avoid brute force search algorithms. (Look for more optimal methods) ● Conventions
    - Commented description of the script's purpose at the head of the code. 5.3.4 The protocol of one or more subsystems, modules, or subroutines

- Due to this software being built upon existing software, communications between interfaces shall occur through JavaScript code and various implemented modules such as: ○ JSON parsers
    - Database queries
    - API responses
- The software shall require a web browser and use WSS over HTTPS, ensuring fully encrypted communication via SSL. The software shall also query data from JPL's APIs in the form of JSON and/or text/file format. While the WSS connection is active, synchronization between the client and web server shall be achieved with serialization and processing. Although WebSockets allow for a low-latency and full-duplex communication system, data transfer issues may arise when used in large-scale systems. The software system's backend web server shall also require a fast database, which shall assist in the prevention of network bottlenecks. This software shall also include a chatroom service, allowing users to communicate via text in real time.

5.3.5 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

- N/A

5.3.6 Plans for maintaining the software

- The system shall use libraries compatible with the existing SST system. This is done to ensure a smooth transition from the testing server to JPL's server in final transition deployment.
- JPL is responsible for maintaining the software upon delivery.

5.3.7 Interfaces for end-users, software, hardware, and communications

- Some software interfaces include:
  - **Tomcat 7.0.59, https://archive.apache.org/dist/tomcat/tomcat-7/v7.0.59/bin/**
    - Used for testing the software environment through localhost.
  - **Esri ArcGIS API for Javascript Version 4.X, https://developers.arcgis.com/javascript/3/jssamples/**
    - Used for 2D visualization.
  - **WebGL, latest version found at, https://www.khronos.org/webgl/**
    - Used for rendering advanced inter-converting 3D and 2D graphics in any compatible web browser without using plug-ins.
  - **WebSockets, latest version found at https://socket.io/**
    - Used for transferring data from client to server in real-time and with low latency.
  - **Existing APIs in use in the existing Trek Lite application.**
    - Used for transferring data from client to server in real-time and with low latency.

5.3.8 Hierarchical organization of the source code into its physical components (files and directories).

- Most of the code is under src/jpl/dijit/
- There is no need to look at the other directories since they are mostly library files used to create a build.
- Src/jpl/dijit/
  - /css - This is the collection of individual styling sheets for each module. ○ /images - Any images you want to include locally can be included here. ○ /templates - These are all the HTML files for the different modules.
  - /ui - These are more JS files that have to do more with user input interaction. ○ /events - For events that transcend several modules their event strings can be found here.
  - /plugins - These are files that are from a third party library but had to be modified to work with Trek Lite.

- /utils - These are files that don't exactly create a module but provide some sort of logic, for example label formatting or WKT conversions.

- Src/jp/dijit/ControlBar.js - This module contains all the overlay buttons seen on the Trek Lite interface and their event handlers.

5.3.9 How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

1. Steps

1.1. Installing Node.js and Npm

1.1.1. Follow the directions in this link https://www.npmjs.com/get-npm to install. Make sure that you install the version 14.15.4. Installing the LTS version of Node will also install npm.

1.1.2. Run the MSI and go through the installation with default settings. 1.1.3. Test if Node is successfully installed by opening your terminal (command prompt) and typing "Node". It should say something along the lines of "Welcome to Node.js v14.15.4". Exit the Node terminal by pressing CTRL+C twice or by typing ".exit".

1.1.4. After exiting out of the node terminal, type "npm version" in the command prompt. It should display all of the npm packages you have installed along with their versions.

1.2. Cloning the CVSST repository using Git

1.2.1. Create a new folder in a directory you would like to store your application and name it "cvsst-test-1.0" .

1.2.2. Using Git Bash, navigate to the directory of your newly created folder and enter the command (without quotation marks):

"git clone https://github.com/srivats22/moontrek_frontend.git".

1.2.3. Using your command prompt, navigate to the directory of your cloned application and enter the command, "npm install".

1.2.4. After entering "npm install", type "npm start". This will launch the application on port 4000, which is the application's default number.

1.3. Installing Moseif CORS Changer

1.3.1. To use the CVSST application, a Chrome extension called

Moseif CORS changer must be installed. Navigate to the Chrome Extension Store and
install the Moseif Origin & CORS Changer extension.

1.3.2. Turn on the extension when you are in development.

Warning: Do not visit any sketchy website, as this extension opens up many security risks such as cross site requests. Turn off when finished.

1.4. Accessing CVSST

1.4.1. Once the application is launched using “npm start”, you can test it out on your browser by navigating to “https://localhost:4000”. Note that it is “https” not “http”.

1.4.2. Go to “https://localhost:4000/src/” to run it.

5.3.10 Tactics such as abstracting out a generic DatabaseInterface class

- Database structure was not set up. Refer to Section 8 for details.

6. Detailed System Design

6.1 User Interface Module

6.1.1 Responsibilities

The User Interface Module serves as the messenger between the user and the Main Module. It provides a graphical user interface (GUI) for the user and allows the user to interact with the entire system.

6.1.2 Constraints

Some constraints may include a limitation of space for additional widgets and/or possible obstruction of views from other elements in the application. There may also be a very high number of entities which may require additional graphical processing power.

6.1.3 Composition

The User Interface Module shall be composed of various widget-like components following the Dojo Toolkit standards. Each User Interface Module shall consist of simple HTML and CSS elements, all of which can be accessed by their corresponding lower level modules.

The components are listed below.

1. Chat Component

a. The Chat Component will consist of a main chat box for containing all chat messages.

b. The Chat Component shall consist of a text input box for sending messages.

c. The Chat Component shall consist of a single button used to send messages.

2. Tool Component

a. The Tools Component shall consist of a dropdown bar consisting of all the collaborative states in the session.

b. The Tools Component shall consist of buttons indicating a variety of tools: free-hand “ink” drawings, polylines drawings, text, circles, squares,

and triangles.

c. The Tools Component shall reveal additional attributes for each button allowing for additional modifications on the markup. Some attributes include drawing sizes, font size, color, and width of lines.

3. Import/Exports Component

a. The Import/Exports Component shall consist of an import button and an export button.

4. States Component

a. The States Component shall consist of all of the markups and entities that have been placed on the 2D and 3D terrain.

5. Collaborative Sessions Component

a. The Collaborative Sessions Component shall consist of a single button which reveals a modal.

b. The modal shall have two buttons: joining and creating a collaborative session.

6.1.4 Uses/Interactions

The user will input data into the system via click events, scroll events, and key press events.

6.1.5 Resources

This module requires a miniscule amount of memory within the user's browser and a sufficient GPU to render additional entities or markups.

6.2 Main Control Module

6.2.1 Responsibilities

The Main Control Module serves as the main component where all of the Collaborative Visualization for Solar System Trek (CVSST) modules are instantiated and initialized. It also handles all user input and directs the data to the correct modules for additional processing.

6.2.2 Constraints

There have been no constraints identified at this time.

6.2.3 Composition

The Main Control Module shall consist of functions which initialize modules 6.3, 6.4, 6.5, 6.6, and 6.7.

6.2.4 Uses/Interactions

The Main Control Module will handle events from the UI. Based on the commands, it will interact with the Chat Module, Tools Module, Imports/Exports Module, or States Module.

6.2.5 Resources

This module requires a miniscule amount of memory within the user's browser.

6.3 Chat Module

6.3.1 Responsibilities

The Chat Module shall obtain data directly from the user interface and update the user interface with new data from module 6.7. This module shall handle all chat events from the UI, update the chat box with data from the session, and send new data to the session, which will go to the server.

6.3.2 Constraints

The constraints are character limits and number of messages that can exist in a collaborative session at once. Depending on the amount of data being sent, network latency and memory usage may increase. A maximum of 500 characters per message shall suffice.

6.3.3 Composition

The Chat Module shall consist of functions which send data to the collaborative session. It will also contain functions which handle new data that is sent from the session, functions that update the UI, and functions that obtain data from the UI.

6.3.4 Uses/Interactions

When there is a message being sent via text, the Main Control Module will handle the event and call functions within the Chat Module which retrieve user input from the UI. Once the input is validated, the data is sent to module 6.7 for processing then to module 6.8 for storage into the database. The WebSocket server will recognize the data, store it in the database, and send the new entry to all users in the same collaborative session. When the WebSocket Server Module receives this new entry, the data will be sent back to the Chat Module to update the UI accordingly.

6.3.5 Resources

This module requires a chat box which will contain all chat messages, an input text form for user chat messages, and a send button to send the messages.

6.4 Tools Module

6.4.1 Responsibilities

The Tools Module shall provide collaborative functionality for tools such as creating markups, drawing on 2D terrain, and creating waypoints. This module shall receive data from the user via inputting it into the user interface. It shall update the tools component and send all necessary data to the states module to carry out the necessary update. The module shall also display which tool is currently in use.

6.4.2 Constraints

The user will be limited to a certain number of tools that can be used concurrently. There will be limits to the attributes of the markup data pertaining to the colors, line types, and size. The data that is input for waypoints will also need to be valid as well.

6.4.3 Composition

This module will consist of components that help to create the various types of markups. There will also be a set of functions that sends and receives data of the waypoints.

6.4.4 Uses/Interactions

The user shall select a tool to use with the 2D terrain. Currently, it is only supported for 2D, but there will be support for 3D terrains in a future version. Depending on the tool the user selects, they may have to select a location on the terrain to mark or draw on. The tools module keeps track of the tool selected, as well as the attributes and location of the markup defined by the user. The module will send the data of the attributes and location of the markup to the State Module to be rendered, as well as the Collaborative Session Module to display to other users.

6.4.5 Resources

This module will require the use of existing SST functions including polyline, entity placement, and fly to functions. The user's GPU will be used to render the markups and drawings.

6.5 Imports/Exports Module

6.5.1 Responsibilities

The Imports/Exports Module will handle all configuration imports and exports for states. The module shall allow users to export their configurations in formats such as text format or a properties file to be imported for future use.

6.5.2 Constraints

There may be limits to how much a file can store. Files may also grow very large in size, which makes exporting and importing the data take a longer length of time. This may also slow down the system.

6.5.3 Composition

The module shall consist of functions that handle the reading and parsing of the configuration file to import the data. This data may also be passed onto the States Module to update and show the preset configuration. The module shall also consist of functions that collect data and write it to the file so that the data can be exported.

6.5.4 Uses/Interactions

The user is able to download and export their data from their current session and save it into a configuration file. The user may also import previous configurations from a file into the session and display the previous configurations.

6.5.5 Resources

The module will read data to obtain the configurations using the browser's memory.

6.6 State Module

6.6.1 Responsibilities

Users can choose to save their sessions for future references. A user can also go back to a previous session by choosing a state. For the definition of a state, please refer to Software requirement documentation.

6.6.2 Constraints

1. Where to save the states
2. The number of states we can save
3. The time it takes to load a previous state
4. How we want to save the state

6.6.3 Composition

N/a

6.6.4 Uses/Interactions

The state module is used by the main module. This would make use of our backend communication to save and retrieve a user's saved session.

6.6.5 Resources

Utilizes our backend communication module and the memory of the machine to save the state.

6.6.6 Interface/Exports

Will be updated later once we get more info.

6.7 Collaborative Session Module

6.7.1 Responsibilities

The collaborative session module is the main source of data transfer. It's responsible for sending and receiving data from our backend. It also transmitted the following: Chat Module, Tools Modules, and Import/Export module.

6.7.2 Constraints

Large sessions might take a long time to save and read. Network speed might be another thing to consider for a smooth experience. Running multiple sessions on our servers would be another thing to consider. Apart from this, we might also want to consider saving some sessions in cache memory to ensure faster loading.

6.7.3 Composition

This module will contain methods that can send and receive data to the backend server module.

6.7.4 Uses/Interactions

When change is made in the current session, data is sent to our backend server and the new data is passed on to other users in the same session. This will happen in real-time so there is no delay.

6.7.5 Resources

A backend server is required. Based on a timestamp the UI can get updated. We also need to ensure multiple data is not sent at once cause this could lead to a crash.

Section 7 onwards will be completed after more development is done

7. Detailed Lower level Component Design

7.1 Collaborative Tools Module
Will be filled out as software is developed further

7.2 Chat Module
Will be filled out as software is developed further

8. Database Design

When the back end of the project is developed further on, we may be using JPL's own database or a cs3 database provided to us by CSULA's computer science department. After studying last year's team's project and what worked best for them, we understood that it is best for us to use a relational database design. Using relational SQL may be slower than a NoSQL database, however it will be more efficient and easy to work with.

9. User Interface

9.1 Overview of User Interface

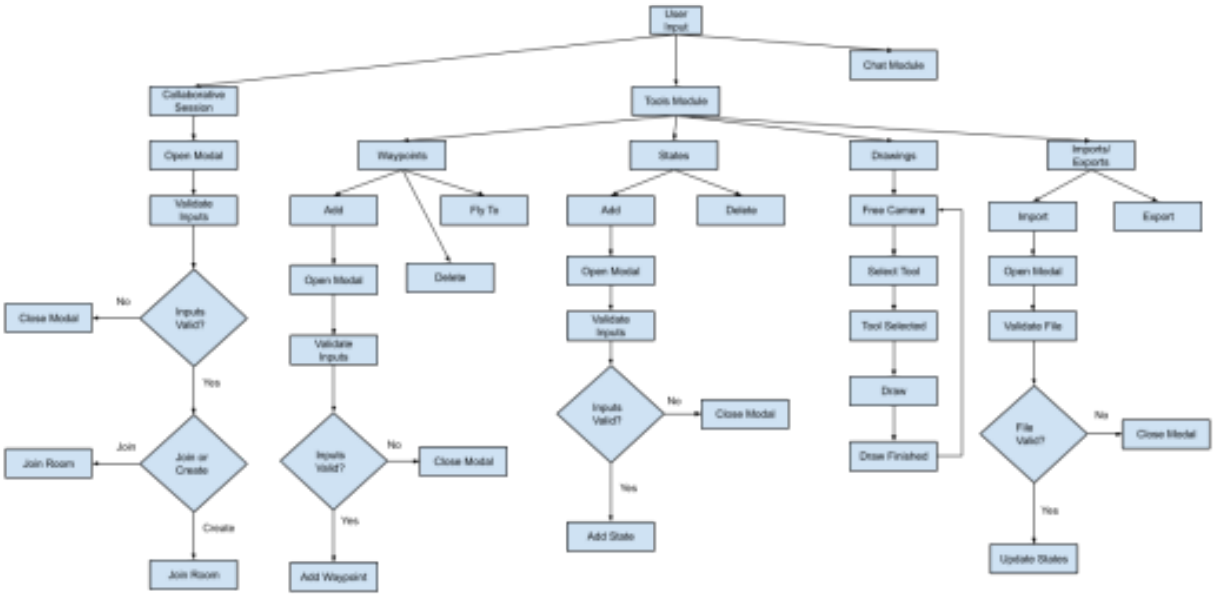
The user would enter the website and be welcomed by the website which would offer a small tutorial to familiarize the user with the website's user interface. The website would present a series of buttons all around the screen that carry out a function. There would be buttons to show and present small interesting facts around the moon which also point to specific parts of the moon where the event was made. In addition, there is the ability to login and create rooms where you can have a live chat with everyone you invited.

9.2 Screen Frameworks or Images

The screenshots will be added when the front end is finished.

9.3 User Interface Flow Model

provided user input.



10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the requirement has been met.

11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Brad Appleton <brad@bradapp.net> http://www.bradapp.net

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc