Software Design Document for NASA Psyche Mission Simulator

Version 2.3

CSE 397 Team 3 – Ryan Funk, Levi Johnson, Jerry Lane, Cayleigh Leishman, Colton Pritchett, Kristin Thumstedter, Gerin Wilde, Brycen Williams

NOVEMBER 9, 2024

Contents

Software Design Document (SDD) for NASA Psyche Mission Simulator

1. Introduction

1.1 Purpose

The purpose of this SDD is to provide a comprehensive overview of the design and architecture of the web-based spacecraft simulator for the Psyche mission, ensuring it meets the outlined requirements in the SRS.

1.2 Scope

This document details the software architecture, system design, user interface, and interactions necessary to implement the functionalities described in the SRS. It serves as a guide for developers, testers, and project stakeholders.

2. Overall System Architecture

2.1 Architectural Style

The simulator will adopt a Model-View-Controller (MVC) architecture, separating concerns among data management, user interface, and user interaction.

2.2 Component Diagram

Frontend:

Model: Handles data, simple logic, and provides forms.

View: Graphic display for simulation and data visualization.

Controller: Manages user input and interacts with the Model to update the View.

Backend:

Model: Contains the static data used in physics calculations, API return format for data.

View: API.

Controller: Business logic, and physics calculations.

Viewpoint:

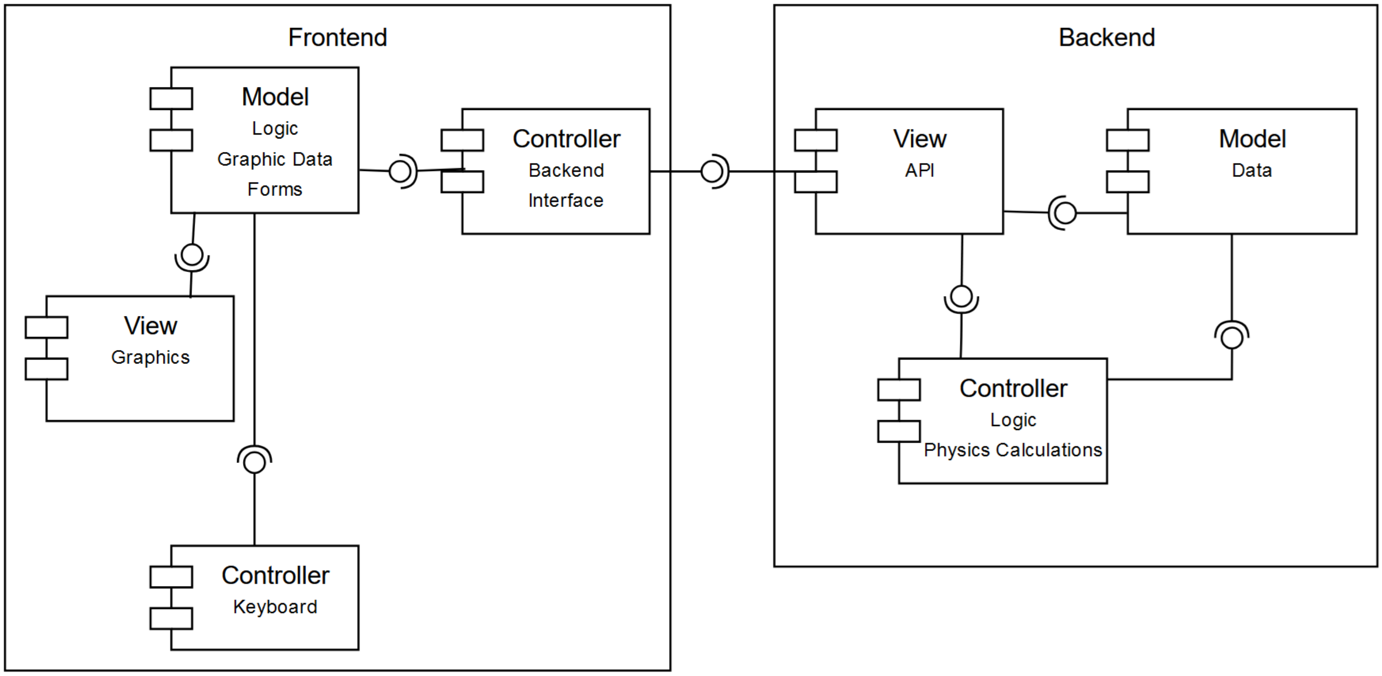

Figure 1 – Architecture Overview

2.3 Technologies

Frontend: HTML5, CSS3, JavaScript (with React, ReactDOM, Vite, App libraries)

Backend: ASP.NET Core

Simulation Engine: Three.js, WebGL for rendering of simulation view

3. Detailed Design

3.1 User Interface Design

3.1.1 Home Page

SRS Requirements: 3.1.1.1, 3.1.1.4, 3.1.10.3, 3.2.1.2, 3.2.2.1

Components: ***Activate Simulator*** button and ***Learn more about NASA's Mission to Psyche*** button.

Design Guidelines: Adhere to WCAG standards for accessibility.

Viewpoint:

Figure 2 – Home Page

3.1.2 Main Menu

SRS Requirements: 3.1.4.3, 3.1.7.1, 3.1.7.3, 3.2.2.1

Description: The Main Menu will be the first menu option the user sees upon access.

Elements: The Main Menu will have a dropdown menu for difficulty options and five button controls as listed:

- Difficulty dropdown menu
  - Expert
  - Intermediate
  - Novice

- Buttons for five options:
  - Load Mission
  - Write Mission
  - Configuration
  - Start Simulation
  - Exit

Viewpoint:

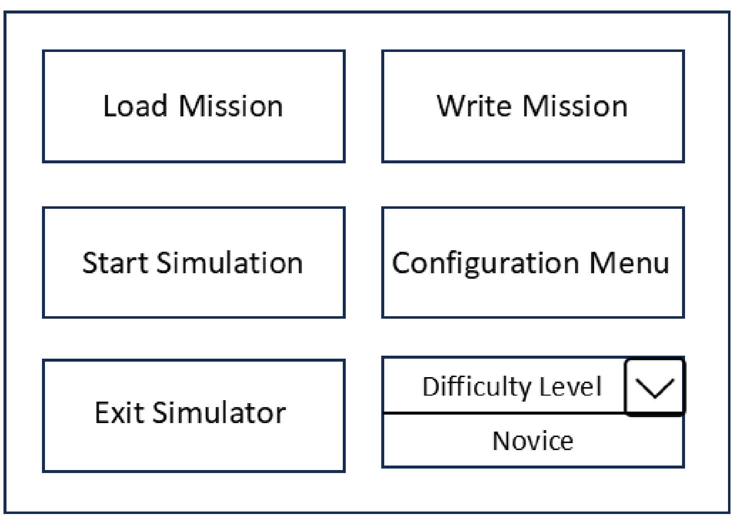

Figure 3 – Main Menu

3.1.3 Configuration Menu

SRS Requirement: 3.1.7.2, 3.2.2.1

Description: After the Configuration Menu button is pressed on the Main Menu, the Configuration Menu is shown, allowing users to set their parameters themselves.

Viewpoint:

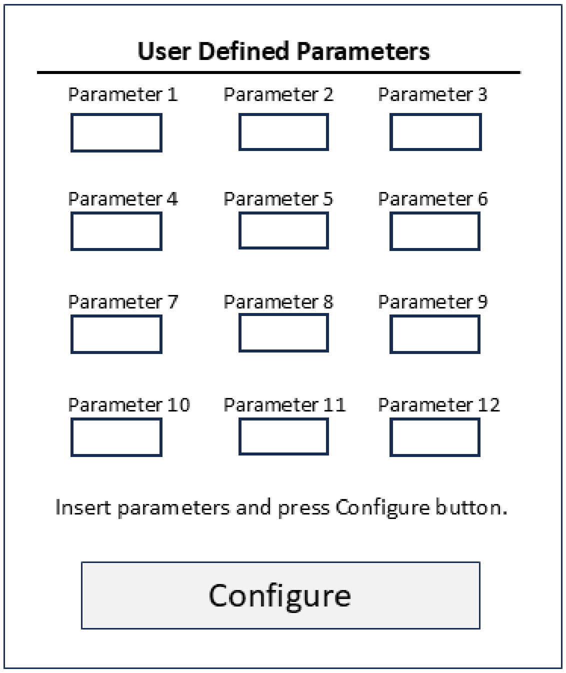

Figure 4 – Configuration Menu

3.2 Functionality Design

3.2.1 Cookie Management

SRS Requirements: 3.1.2.1, 3.1.2.2, 3.1.2.3

Cookies will only be used for functionality, so there is no need to offer users a choice to opt out. In this application, the cookies will be used to store the session ID. We may also use the cookies to track unique visits to our application, so NASA could evaluate public interest in the Psyche Mission.

3.2.2 Error Handling

SRS Requirements: 3.1.3.1, 3.1.3.2, 3.1.3.3, 3.1.8.1, 3.1.9.1

Purpose: Implement a centralized error handling mechanism for user information and feedback.

Description: Handle all errors through the following:

- Display server outage with a popup window requiring user acknowledgement.
- In case of guest sign on, auto-set limited features to novice default settings.
- When input configuration data is invalid, warn user with popup and reset to default value.

- Should the lander go too far off course or sustain damage, the user is prompted by popup window that the mission has failed. The application is set to default parameters.

3.3 Simulation Controls

3.3.1 Simulation Parameters

SRS Requirements: 3.1.5.2, 3.1.5.4, 3.2.2.1

Input and Display Handling

Create a form to input and display the below parameters validated in real-time.

Developer Defined (Constant) Parameters

These parameters are to be defined by the developer and unable to be tweaked by the standard user.

- Acceleration due to gravity of Psyche (float)
- Average diameter of Psyche (float)
- Rotation speed of Psyche (float)
- Mass of the spacecraft lander (float)
- Amount of fuel for thrusters designated for landing (int)
- Minimum force from landing impact that the lander can sustain (int)
- Diameter of landing feet (float)
- Minimum/maximum radial distances the lander can start at (float)
- Maximum level of thrust that can be exerted (int)
- Maximum distance of the lander from Psyche that's considered too far (out of bounds threshold) (float)

User Defined (Configurable) Parameters

These parameters are to be adjusted by the user before (and perhaps during for some) the simulation.

- Starting location of the lander: The location of the lander will be determined using the polar coordinate system. The coordinates for such are as follows.
    - Radial distance (ρ) (float)
    - polar angle (θ) (float)

- Facing angle of the lander (same conditions from lander location apply here)
- Distance of the lander from Psyche in which to turn on thrusters (float)
- Level of thrust to decrease gravitational acceleration (float)
- Angle of thrusters relative to the lander (float)

Real-Time (Variable) Parameters

These parameters are those that update in real time during the simulation, not defined or adjusted by a developer or user, and displayed to the user as information during runtime.

- Current acceleration of the lander (float)
- Current velocity of the lander (float)
- Current level of thrust (float)
- Amount of fuel remaining (int)
- Amount of damage the lander has sustained (int)
- Distance between the lander and Psyche (float)
- Time elapsed (float)

Conclusion

With these parameters considered, implications can be made regarding the overall design.

- The simulation will run in two dimensions.
- Psyche's surface won't be perfectly symmetrical, making the facing angle of the lander and its rate of descent conditional by its starting location.
- The lander will start at a fixed position without any additional velocities due to orbit or otherwise.
- The lander must make a soft landing below the force of impact sustain threshold.
- The lander must make a secure landing without losing balance and toppling over.

Viewpoint:

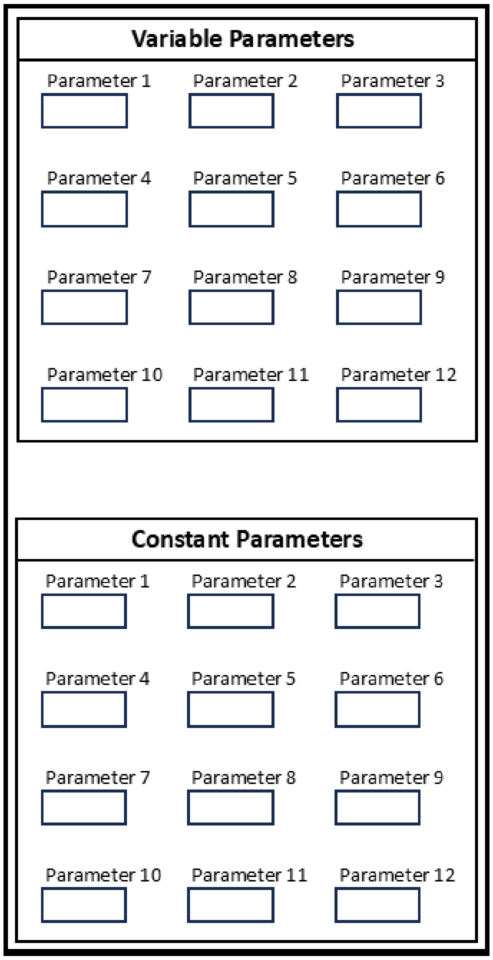

Figure 6 – Parameter Panel of Simulator

3.3.2 Keyboard Controls

SRS Requirement: 3.1.7.4, 3.1.7.6

Description: User will have access to keyboard controls for real-time adjustment of thrusters. Specific keys will be identified as individual thrusters, and the cursor left and right keys will act to increase and decrease thruster angles, with the cursor down arrow acting to turn the thruster on and the cursor up key to turn the thruster off.

3.3.3 Simulation Management

SRS Requirement: 3.1.7.5, 3.2.2.1

Control Functions: Implement start, pause, exit, and resume features in the application using state management.

Viewpoint:

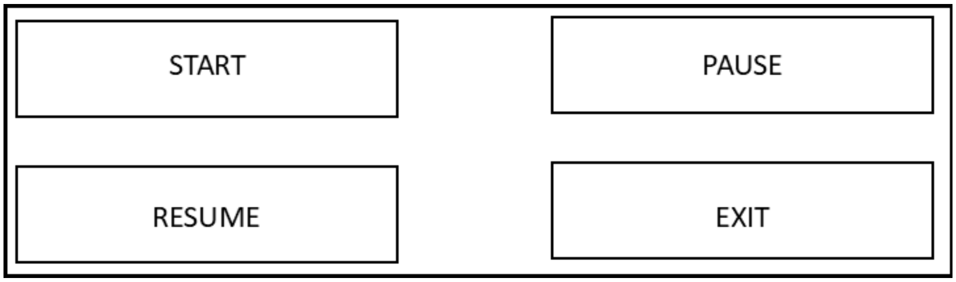

Figure 7 – Simulation Control Panel of Simulator

3.4 Data and Simulation Visuals

3.4.1 Visualization Windows

SRS Requirement: 3.1.5.1, 3.1.5.2, 3.1.5.3, 3.1.5.4, 3.1.5.5, 3.1.7.3, 3.2.1.3, 3.2.2.1

Window Components: Two main windows, one for data readout and one for simulation visuals. (Note: Viewpoint includes the simulation controls from Section 3.3.3 in addition.)

Data Parameters: Display constant metrics like spacecraft weight, Psyche gravity, Psyche rotation, and maximum landing speed. Show real-time data such as distance from Psyche, descent speed, fuel remaining, rotation direction, rotation speed, and all thruster actions.

Viewpoint:

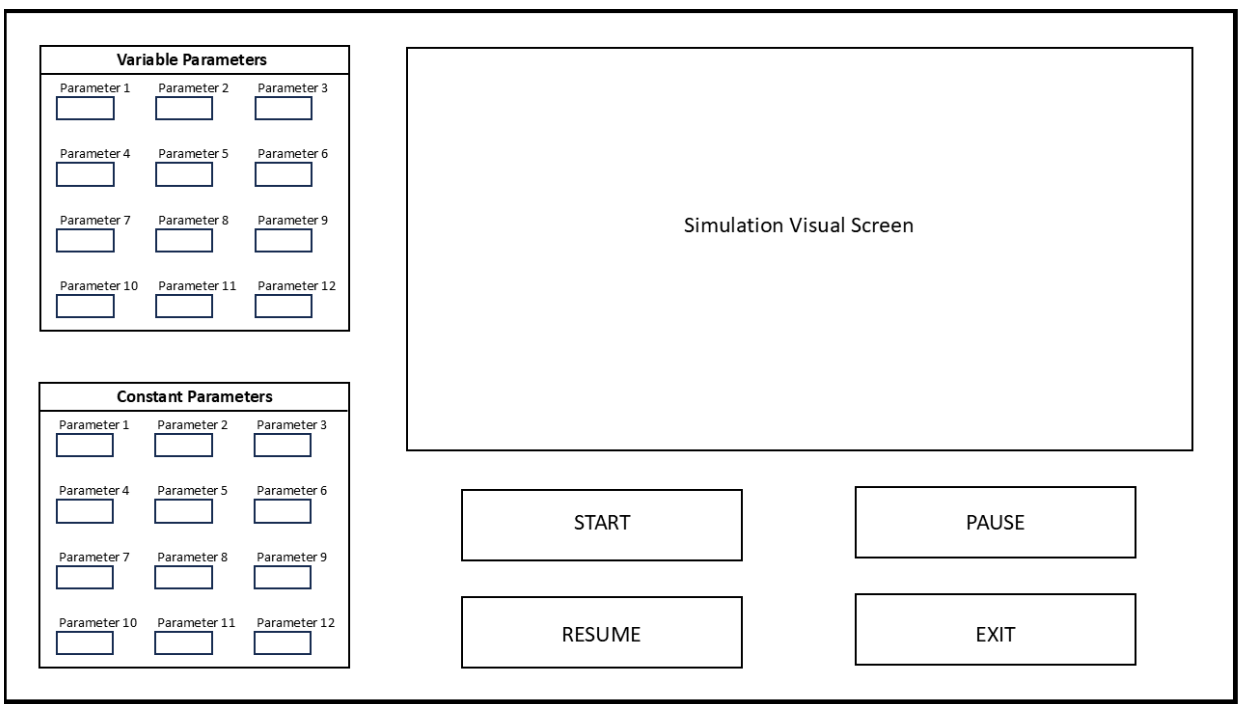

Figure 8 – Simulation Screen with Parameter Panel, Simulation Visualization Screen, and Simulator Controls

3.5 File Access

3.5.1 Parameter Load and Save Functions

SRS Requirement: 3.1.4.4

Description: Allow storage of and access to parameter data on local machine and external storage medium.

Viewpoint: (See Figure 2)

3.6 Front-end Structure

3.6.1 Home page

SRS Requirements: 3.1.1.1, 3.1.1.4

Description: When user navigates to the IP address where the front-end code is stored, the browser window will show the home page as noted in section 3.1.1 of this document. If the user clicks on the ***Use Simulator*** button, a User class object (see section 3.6.2) will be created and the methods contained therein will drive all front-end actions until the application is terminated by the user.

Viewpoint: (See Figure 2)

3.6.2 API request JSON format

SRS Requirements: 3.1.5.1, 3.1.5.2

Description: During an active simulation, calculations will be performed requested by the front-end and performed in the back-end via an API call. The format of the JSON request will be as noted in this section.

Viewpoint:

API Request JSON:

{
  "user_id": [{
    "time": "time simulation began",
    "current": "current time",
    "last": "time last response received",
    "thruster": "on/off",
    "altitude": "current altitude in meters",
    "prior": "old altitude",
    "fuel": "current fuel in kg",
    }
  ]
}

Response JSON:

{
  "user_id": [{
    "calculation": " time calculations performed",
    "velocity": "falling velocity",
    "fuel": "fuel remaining",
    "damage": "0% / actual damage %",
    "height": "new altitude",
    "elapsed": "time since simulation began",
    }
  ]
}

3.6.3 User class

SRS Requirements: 3.1.3.1, 3.1.4.3, 3.1.4.4, 3.1.5.1, 3.1.5.2, 3.1.7.1, 3.1.7.2, 3.1.7.3, 3.1.7.4, 3.1.7.5, 3.1.7.6, 3.1.8.1, 3.1.9.1, 3.1.10.3, 3.2.1.1, 3.2.1.2, 3.2.1.3

Description: The User class will support all the required actions as outlined in SRS v2.3. A User object will be created when the user selects the ***Use Simulator*** button on the home page. This class will support the following methods:

- Load mission (file read function)
- Save mission (file write function)
- Configuration (user parameter input function with error checking)
- Skill level (automatic parameter assignment by application according to selection)
  - Novice
  - Intermediate
  - Expert
- Start (begin simulator and activate simulation controls
- Exit (end simulation immediately – same as closing the browser window)

Upon selecting Start from the Main Menu, the simulator will use the parameters set by the system or those modified by the user, whichever occurs last and the simulation window will be displayed as shown in Figure 8. The User class will then support the following simulation controls:

- Start (start the simulation graphics and update variables in real-time)
- Pause (temporarily halt the simulation)
- Resume (close the Main Menu and continue simulation)
- Exit (the Main Menu will be shown in case the user wishes to save the current mission data)

Viewpoint:

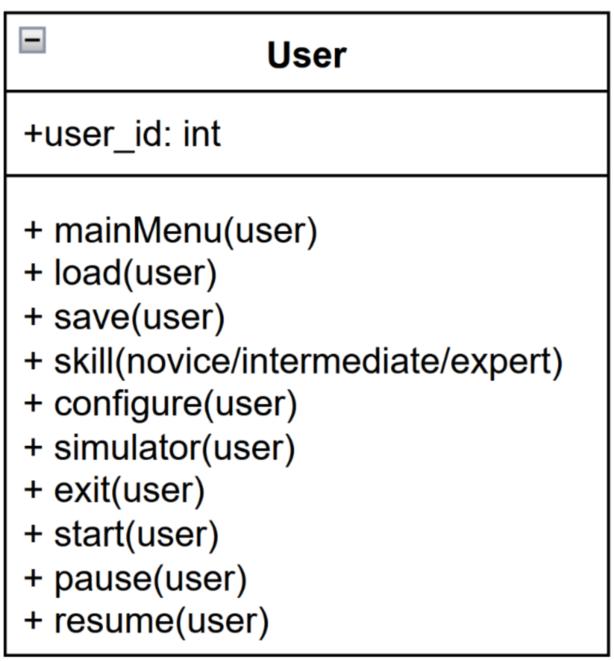

Figure 9 – User class diagram

4. Non-Functional Requirements

4.1 Performance

Load Testing: The system should handle 100 simultaneous users without performance degradation.

Response Times: Home page load within 3 seconds, real-time updates on parameter changes.

4.2 Usability

User Testing: Regular usability testing with target user groups to refine the interface, particularly regarding accessibility.

5. Validation Strategy

5.1 Validation Techniques

Unit Testing: For individual components.

Integration Testing: For interactions between components.

User Acceptance Testing (UAT): Involving end-users to validate against SRS.

5.2 Success Criteria

All critical requirements must be verified.

User feedback must indicate satisfactory usability and engagement.

6. Conclusion

This SDD outlines the architectural and design approach for the NASA Psyche Mission Simulator, providing a clear roadmap for development and implementation. By adhering to these specifications, the project aims to deliver a robust and engaging educational tool for various user groups.