Software Requirements Specification

for Bounce Mission Control (NASA API Explorer)

Version: 1.1
Date: 18 June 2025
Author: Gemini
Project Name: Bounce Mission Control

1. Introduction

1.1 Purpose

This document provides a detailed specification of the requirements for the "Bounce Mission Control" web application. Its purpose is to serve as a foundational guide for developers, designers, and testers, ensuring that the final product aligns with the objectives outlined in the "bounce Insights" coding challenge. It defines the system's features, constraints, and success criteria.

1.2 Project Scope

The project entails the development of a full-stack web application that allows users to explore space-related data provided by NASA's Open APIs. The application will consist of a React frontend, a Node.js/Express backend, and will be deployed to a live environment. The Minimum Viable Product (MVP) will focus on two core features: the **Astronomy Picture of the Day (APOD) Viewer** and the **Mars Rover Photo Explorer**.

1.3 Intended Audience

This SRS is intended for:

- **Software Developers:** To understand the technical requirements, architecture, and features to be implemented.
- **Project Evaluators:** To assess the completed project against the specified requirements and evaluation criteria.

1.4 Definitions, Acronyms, and Abbreviations

Term	Definition
APOD	Astronomy Picture of the Day
API	Application Programming Interface
MVP	Minimum Viable Product

NASA	National Aeronautics and Space Administration
SRS	Software Requirements Specification
UI	User Interface
UX	User Experience
Sol	A Martian day, as measured by NASA for rover missions.
CI/CD	Continuous Integration / Continuous Deployment

2. Overall Description

2.1 Product Perspective

Bounce Mission Control is a self-contained web application that acts as a client to NASA's public data services. It will feature a decoupled architecture where the React frontend communicates with a dedicated Node.js backend. The backend serves as a secure proxy and intermediary to the external NASA APIs, abstracting the data-fetching logic from the client.

2.2 Product Features

The application will provide the following primary features:

- **APOD Viewer:** Users can view the Astronomy Picture of the Day for any selected date.
- **Mars Rover Photo Explorer:** Users can browse photos taken by the Mars Curiosity rover, with the ability to search by Martian day (sol).
- **Data Visualization:** Data will be presented in a visually engaging and intuitive manner, focusing on high-quality imagery and clean typography.

2.3 User Characteristics

The target user is a member of the general public with an interest in astronomy and space exploration, as well as the technical evaluation team from "bounce Insights". Users are expected to have a basic understanding of web navigation.

2.4 Constraints

- **C-1: Technology Stack:** The frontend must be built with **React**. The backend must be built with **Node.js** and **Express**.

- **C-2: Deadline:** The project must be completed and submitted within two weeks of receipt of the challenge.
- **C-3: Deployment:** The application must be deployed to a live, publicly accessible URL.
- **C-4: API Source:** The application must exclusively use the NASA Open APIs (api.nasa.gov) as its data source.

2.5 Assumptions and Dependencies

- **A-1:** The NASA Open APIs will be available and functioning correctly during the development and evaluation period.
- **A-2:** The developer has been provided with a NASA API key or will use the DEMO_KEY.
- **D-1:** The project is dependent on third-party services for deployment (e.g., Vercel for the frontend, Render for the backend).

3. System Features (Functional Requirements)

3.1 SF-1: APOD Viewer

- **3.1.1 Display Today's APOD:** Upon loading the APOD page, the system shall fetch and display the current day's Astronomy Picture of the Day.
- **3.1.2 Display APOD by Date:** The UI shall provide a date picker, allowing the user to select a past date. Upon selection, the system shall fetch and display the APOD for that specific date.
- **3.1.3 Handle APOD Media Types:** The system shall correctly render the APOD whether the returned media type is an image or an embedded video (e.g., from YouTube).
- **3.1.4 Display Metadata:** For each APOD, the system must display its title and the full text of the explanation provided by the API.

3.2 SF-2: Mars Rover Photo Explorer

- **3.2.1 Display Latest Photos:** By default, the system shall display the most recent photos available from the Mars Curiosity rover.
- **3.2.2 Search Photos by Sol:** The UI shall include a search input field where a user can enter a Martian day (sol) number. Upon submission, the system shall fetch and display all photos taken by the Curiosity rover on that sol.
- **3.2.3 Image Gallery View:** The photos shall be presented in a responsive gallery format (e.g., a grid).
- **3.2.4 Image Detail View:** Clicking on a photo thumbnail in the gallery shall display a larger, high-resolution version of the image, for instance, in a modal overlay.

3.3 SF-3: Backend API

- **3.3.1 APOD Endpoint:** The backend shall expose a GET endpoint (e.g., /api/apod) that accepts a date parameter, retrieves the relevant data from the NASA APOD API, and returns it to the frontend.
- **3.3.2 Mars Rover Photo Endpoint:** The backend shall expose a GET endpoint (e.g., /api/mars-photos) that accepts a sol parameter, retrieves the relevant photos from the NASA Mars Rover Photos API, and returns the data to the frontend.
- **3.3.3 API Key Abstraction:** The backend shall manage the NASA API key. The key must not be exposed to the frontend. It will be injected into requests made from the backend to the NASA API.

4. Non-Functional Requirements

4.1 NFR-1: Performance

- **4.1.1 Loading States:** The UI must display a clear loading indicator (e.g., a spinner) while data is being fetched from the backend. This is a mandatory requirement.
- **4.1.2 Image Optimization:** The Mars Rover photo gallery should lazy-load images or use pagination/infinite scroll to ensure a fast initial page load.

4.2 NFR-2: Usability & UX

- **4.2.1 Responsiveness:** All UI components must be fully responsive and provide an optimal viewing experience on desktop, tablet, and mobile screen sizes. (Bonus Point)
- **4.2.2 Intuitive Navigation:** The application's navigation should be clear and simple, allowing users to easily switch between the APOD and Mars Rover modules.

4.3 NFR-3: Reliability

- **4.3.1 Error Handling:** The application must gracefully handle errors. If an API call fails (either from frontend to backend, or backend to NASA), a user-friendly error message must be displayed on the UI.

4.4 NFR-4: Code Quality & Maintainability

- **4.4.1 Code Structure:** The code shall be well-structured, readable, and organized into logical components and modules. The repository must be organized with separate frontend/ and backend/ directories.
- **4.4.2 Documentation:** The repository must include a comprehensive README.md file with setup instructions. Code should be commented where

complex logic is implemented.

4.5 NFR-5: Deployment

- **4.5.1 Live Application:** The frontend and backend must be deployed to live, publicly accessible URLs.
- **4.5.2 DevOps (Bonus):** To showcase advanced skills, the backend should be containerized using Docker and deployed to the hosting service. A CI/CD pipeline using GitHub Actions should be implemented to automate linting, testing, and deployment.

4.6 NFR-6: Testing (Bonus Point)

- **4.6.1 Backend Testing:** The backend API endpoints shall have unit tests (e.g., using Jest) to verify their functionality.
- **4.6.2 Frontend Testing:** Key frontend components shall have unit or integration tests (e.g., using Jest and React Testing Library).