



Universal Explorer - Documentation

Application Frameworks – SE3040

Assignment 2

IT Number	Name
IT21321436	Gunatilleke M.B. D.S

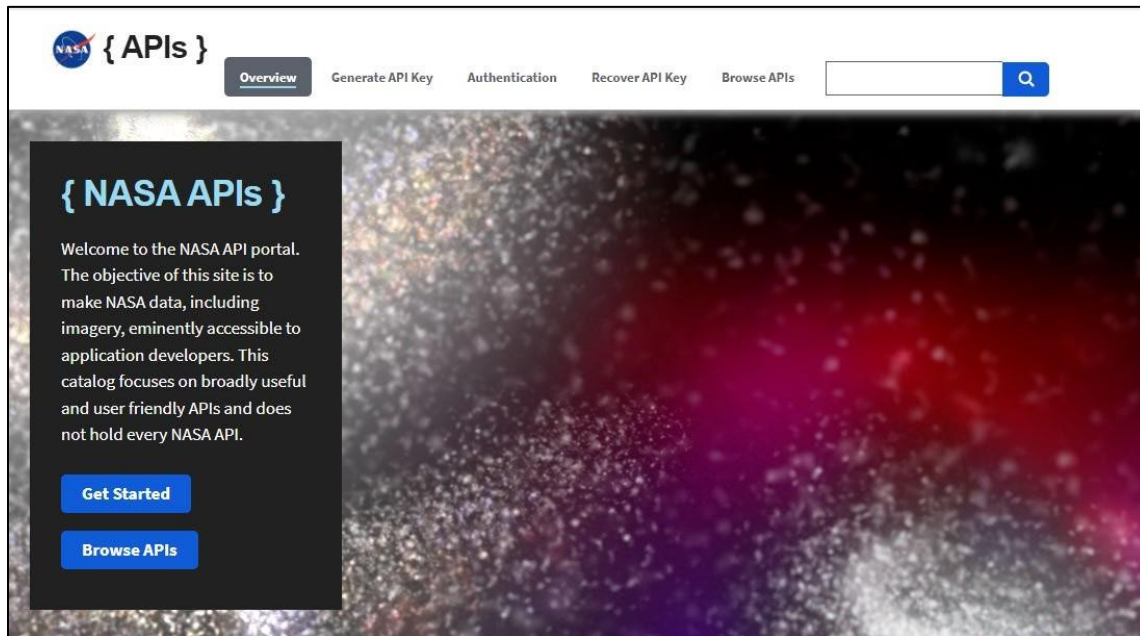
## Table of Contents

1. NASA API's .....	3
How to use the NASA Open API's .....	3
NASA Open API's used in Explore Space.....	4
APOD – Astronomy Picture Of the Day .....	4
Mars Rover Photos .....	5
2. Implementation of the system .....	7
3. User Interfaces .....	9
4. Challenges Faced .....	13
Usage of NASA Open APIs .....	13
Responsive User Interface Creation .....	13

# 1. NASA API's


## How to use the NASA Open API's

The National Aeronautics and Space Administration which is abbreviated to NASA has made several API's available to the public use. They're known as NASA Open API's. They can be found at the website of the NASA API website which is <https://api.nasa.gov>.



***Landing Page of the NASA Open API Website***

After arriving at the landing page of the website, people can give their personal information such as the name and the email. Afterwards, an email containing the 'API KEY' for that specific user will be sent to the email provided. That API KEY should be used while fetching information from the specific API listed on the website.

 { APIs }

[Overview](#)[Generate API Key](#)[Authentication](#)[Recover API Key](#)[Browse APIs](#)

## Generate API Key

Required fields are marked with an asterisk (\*).

First Name \*

Last Name \*

Email \*

How will you use the APIs? (optional)

This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.

**a**

### *How to generate the API KEY*

## NASA Open API's used in Explore Space

### APOD – Astronomy Picture Of the Day

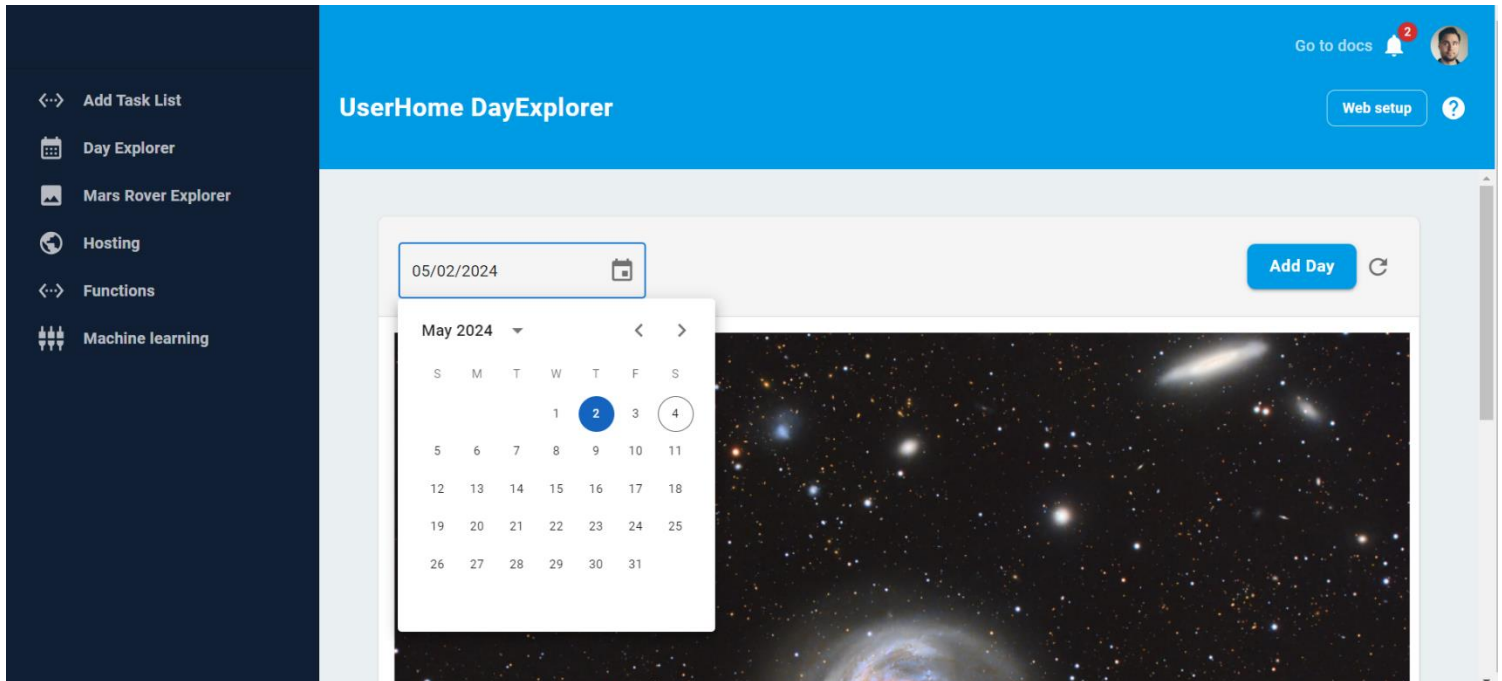
GET <https://api.nasa.gov/planetary/apod>

This API is one of the most popular used by developers. Once a request is sent through this API, it returns a picture related to astronomy and it provides a brief explanation regarding the picture in the response. The request needs to be sent with the API KEY that was generated using the NASA web site.

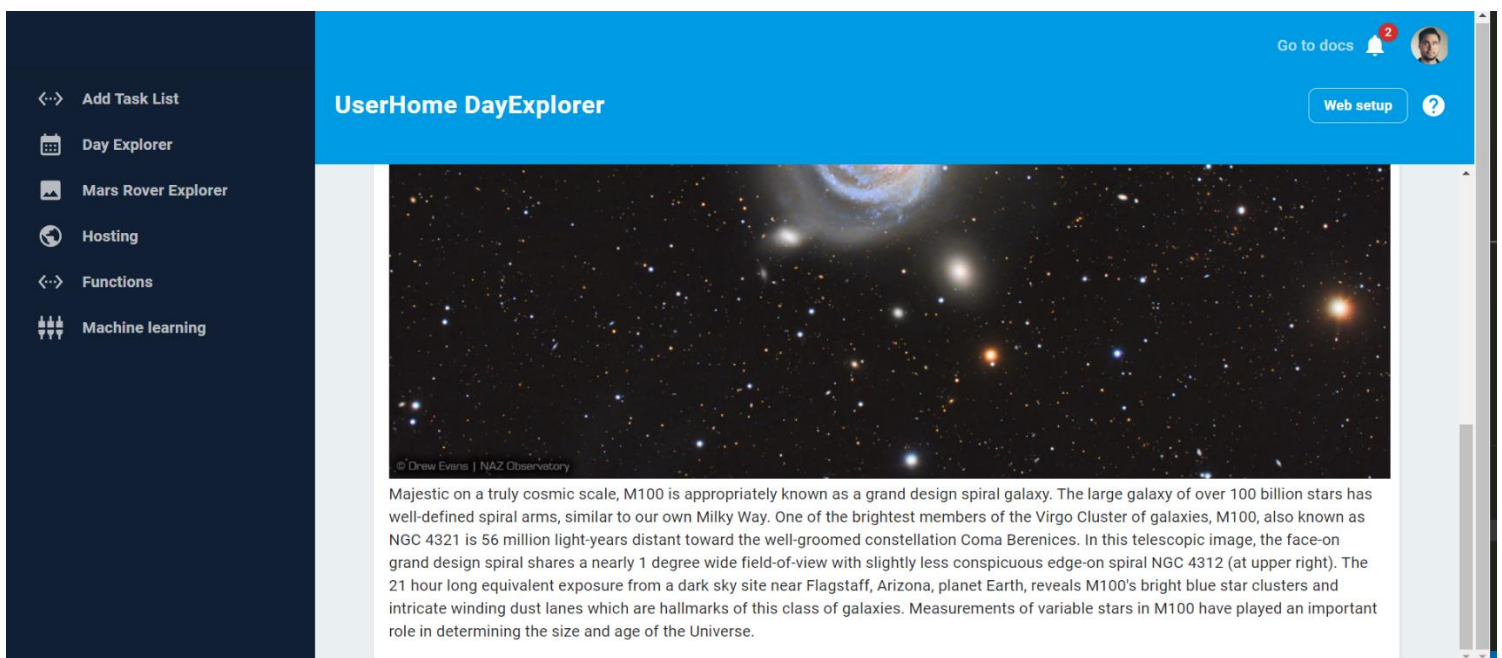
```
const fetchNASADData = async (date) => {
  try {
    const NASA_KEY = process.env.REACT_APP_NASA_API_KEY;
    const formattedDate = dayjs(date).format('YYYY-MM-DD'); // Format the date
    const response = await
    fetch(`https://api.nasa.gov/planetary/apod?api_key=${NASA_KEY}&date=${formattedDate}`);
    const data = await response.json();
    setNasaData(data);
  } catch (error) {
    console.error('Error fetching NASA data:', error);
  }
};
```

### *Fetch Request to get the APOD Response*

The above Code implementation done for this APOD API. In there I used `api_key` parameter inside of my global frontend `.env` file so that I can easily call it when it needed. When inserting data through the API end point query I use to convert the user Input Date to a formatted date type



***User Interface 1 for the APOD Component***



***User Interface 2 for the APOD Component***

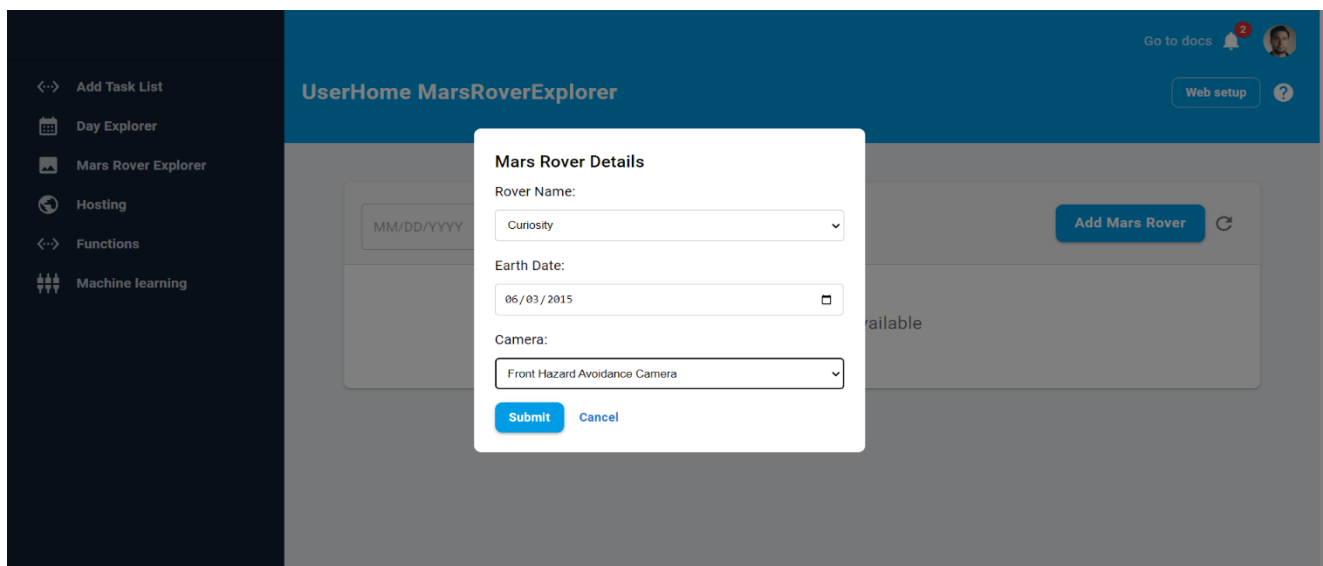
## Mars Rover Photos

The Mars Rover Photos API was designed to collect images gathered by the Rovers which Curiosity, Opportunity, and Spirit rovers on the Martian planet. This API query is different than the APOD API endpoint. This has 5 attributes that can be changed. There are rover's name, The date type (Earth or sol) camera type, pages and api\_key. Some time one request can have bunch of images so it splits to pages which one can hold 25 images with there details

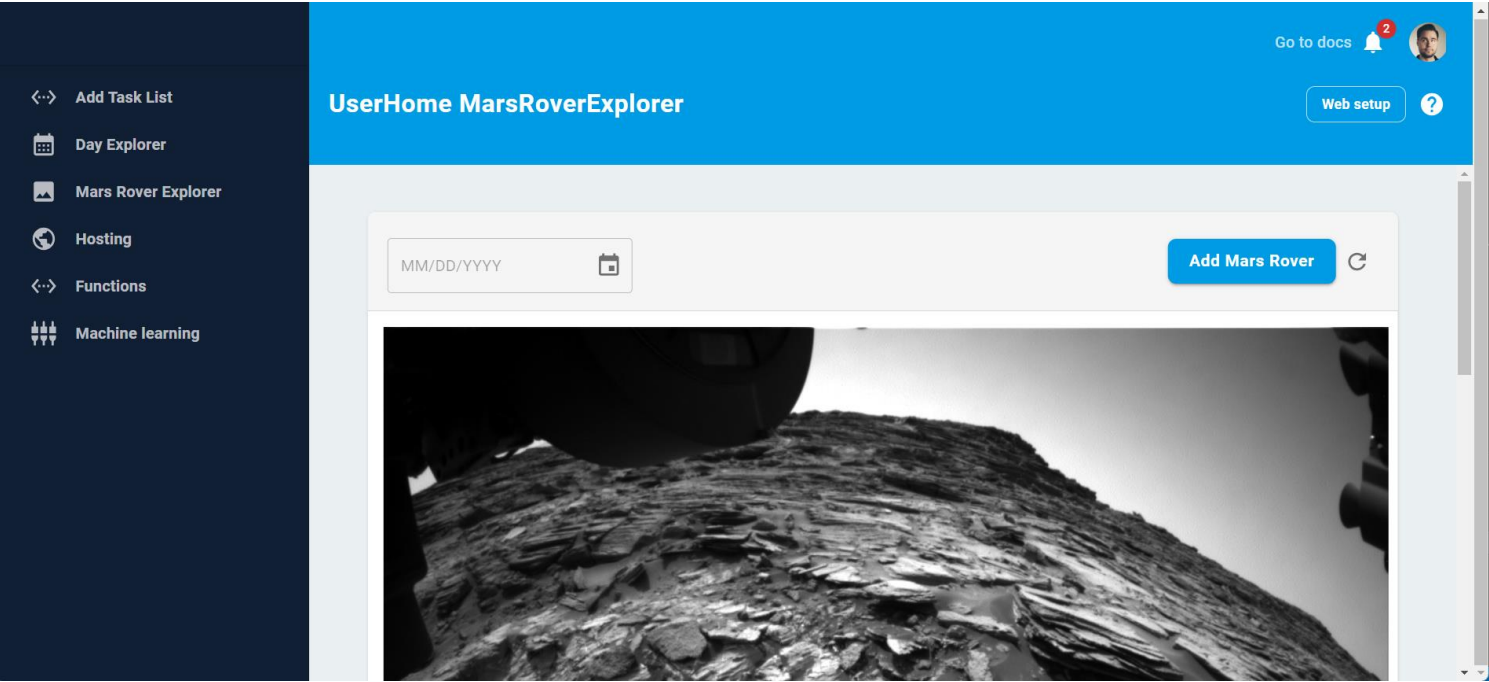
```
const fetchMarsRoverData = async (roverName, earthDate, camera) => {
  try {
    const NASA_KEY = process.env.REACT_APP_NASA_API_KEY;
    const formattedDate = dayjs(earthDate).format('YYYY-MM-DD');
    console.log(roverName, earthDate, camera);
    const response = await fetch(`https://api.nasa.gov/mars-photos/api/v1/rovers/${roverName}/photos?earth_date=${formattedDate}&camera=${camera}&api_key=${NASA_KEY}`);
    const data = await response.json();
    setMarsRoverData(data);
  } catch (error) {
    console.error('Error fetching Mars Rover data:', error);
  }
};
```

### *Fetch Request to get the Mars Rover Response*

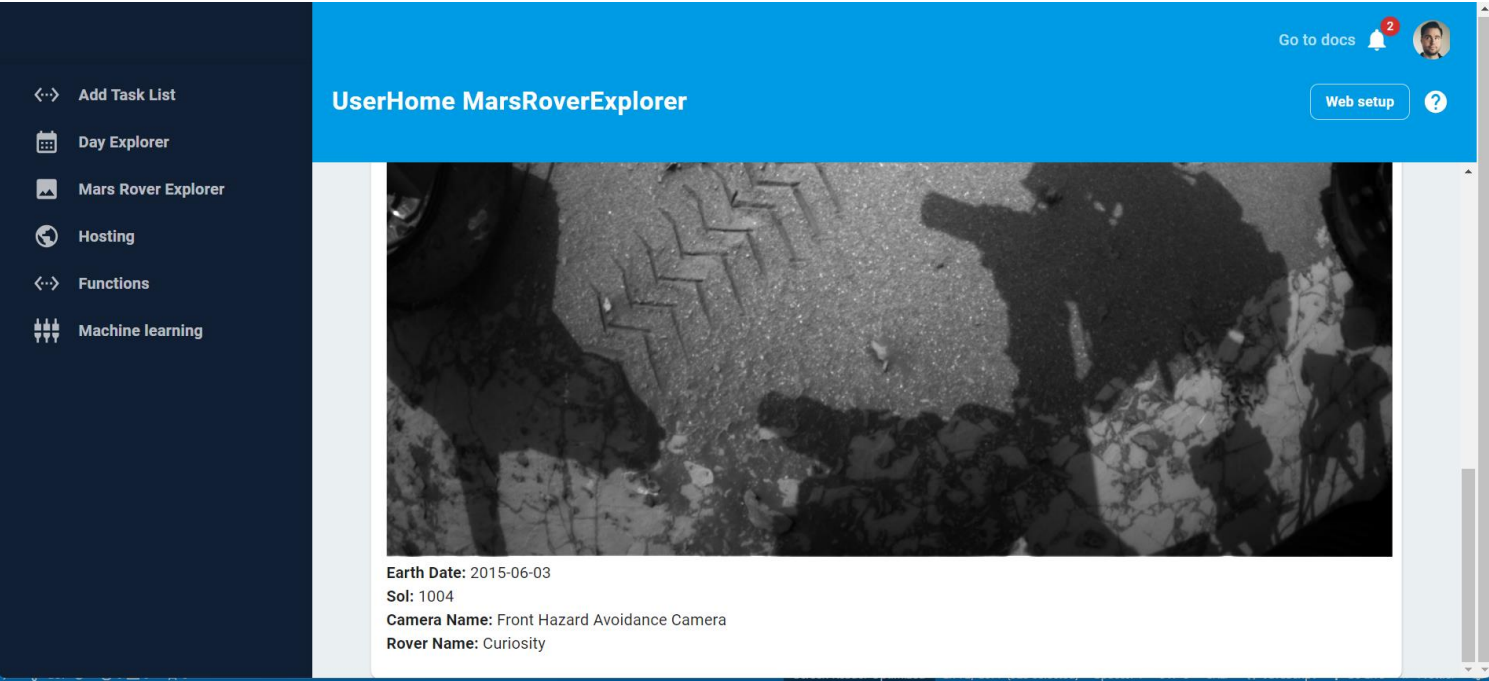
I use this API in a different way. I use to give the control to the user about the API endpoint query variables such as rover name, camera, date, So that one can search their desired intends. In there also I convert the the date format to a standard one before passing it to the query.



*User Interface of the Mars Rover Details modal*



*User Interface of the Mars Rover Details Image*



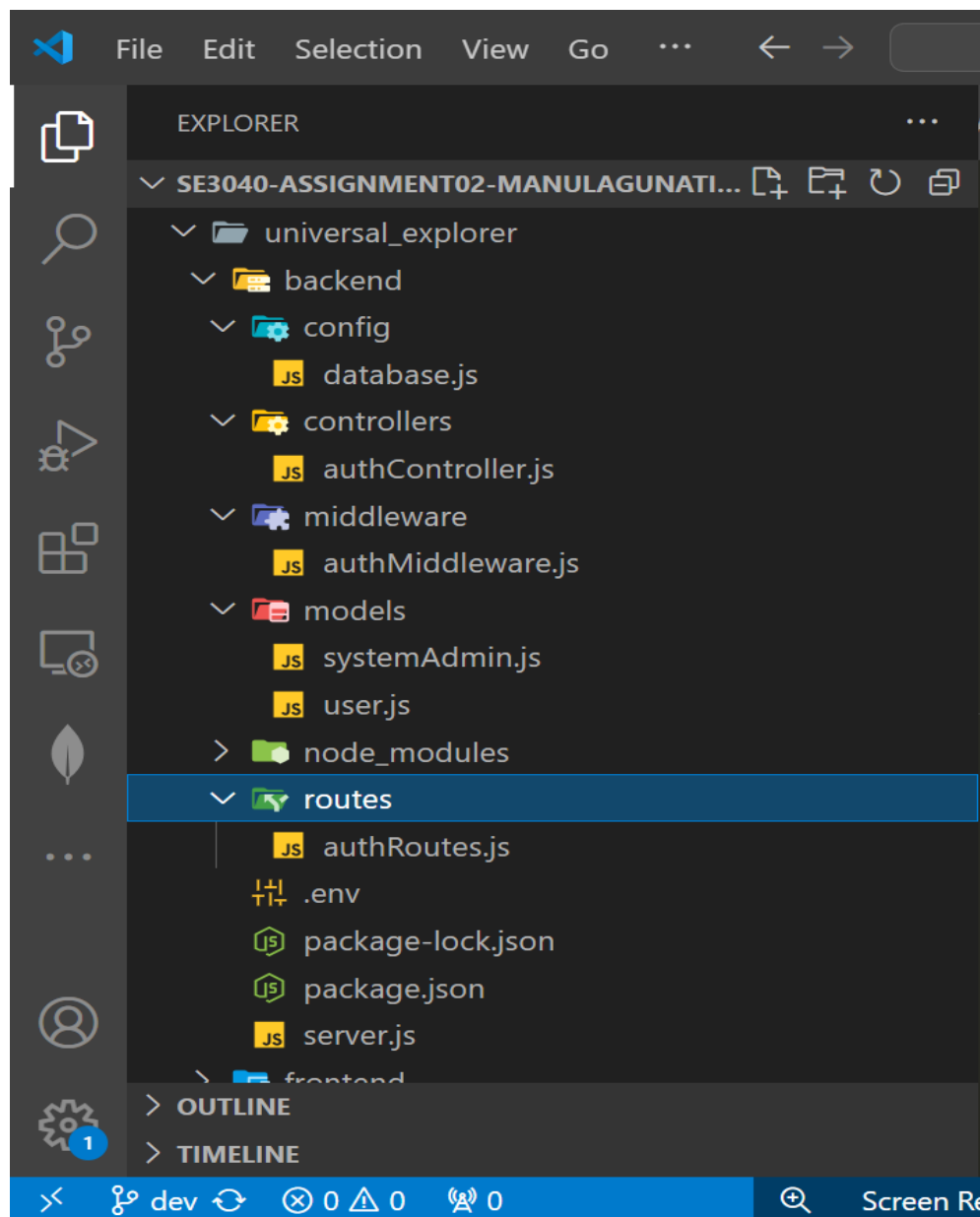
*User Interface of the Mars Rover Details*



## 2. Implementation of the system

In the Development of the web site I manage 2 separate folders, one for backend and one for the frontend. This **Universal Explorer** uses Material UI as the CSS framework, React Library for the project implementation, MongoDB as NoSQL Database and NodeJS as the backend framework.

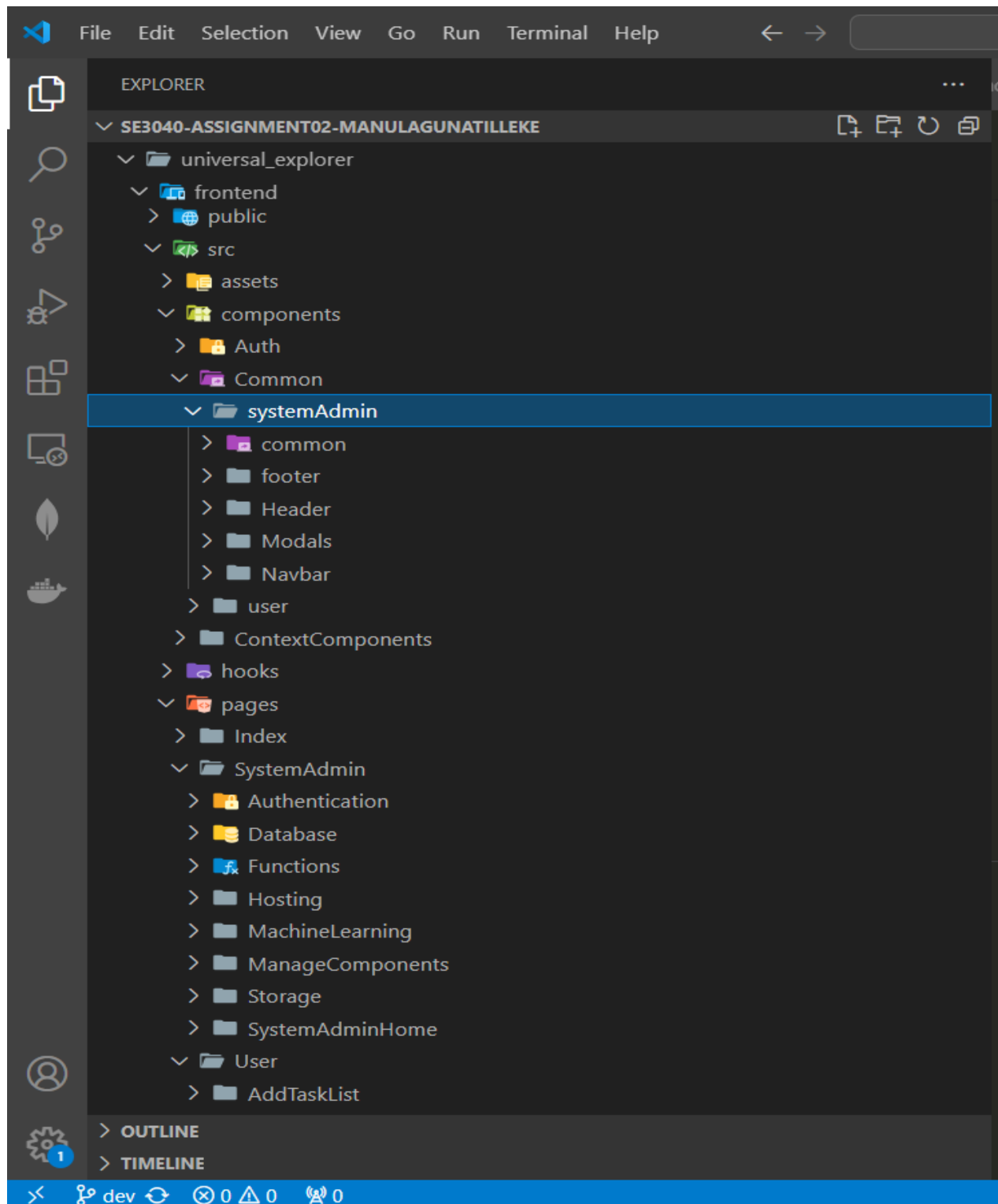
The backend development was done mostly for the user management system. I developed a collection of APIs that allowed the users to register themselves and login to the system. Authentication is also needed to login to the system and access the other components for the user. This was done by using honeytoken node package, which gives a random JSON web token every time the user is logged in, so the user must be logged in to access the components with a valid JSON web token for authentication. The testing of the APIs was done using the Postman API Testing tool and In frontend I use some unit testing as well.



*File Structure of the backend system*



The frontend development of Universal Explorer was done using the React Library. I used the function-based components for the development of the components over class-based components. I also used Material UI as a CSS framework which was used to get components such as Box, Modals, and Icons.



## Frontend Folder Structure

### 3. User Interfaces

Fill out inputs and hit 'submit' button.

Task ID \*  
123456

Priority \*  
High

Description \*  
About 24th April of this year curiosity

Submit Cancel

Search by task ID

Add Task

#### Add Tasks as Reminders

Fill out inputs and hit 'submit' button.

Task ID \*  
25678  
Task ID must be at least 6 characters

Priority \*  
Low

Description \*  
Error checking

Submit Cancel

Search by task ID

Task ID: 123456  
Priority: High  
Description: About 24th

Add Task

#### Form Validations

# Universal Explorer



## Nasa API Services

*"You can explore the Univers with one Click"*

LOGIN

Home Page

# Universal Explorer

LOGIN



### Login

UserType:

Select Type

Email:

it213214361@my.sliit.lk

Password:

\*\*\*\*\*

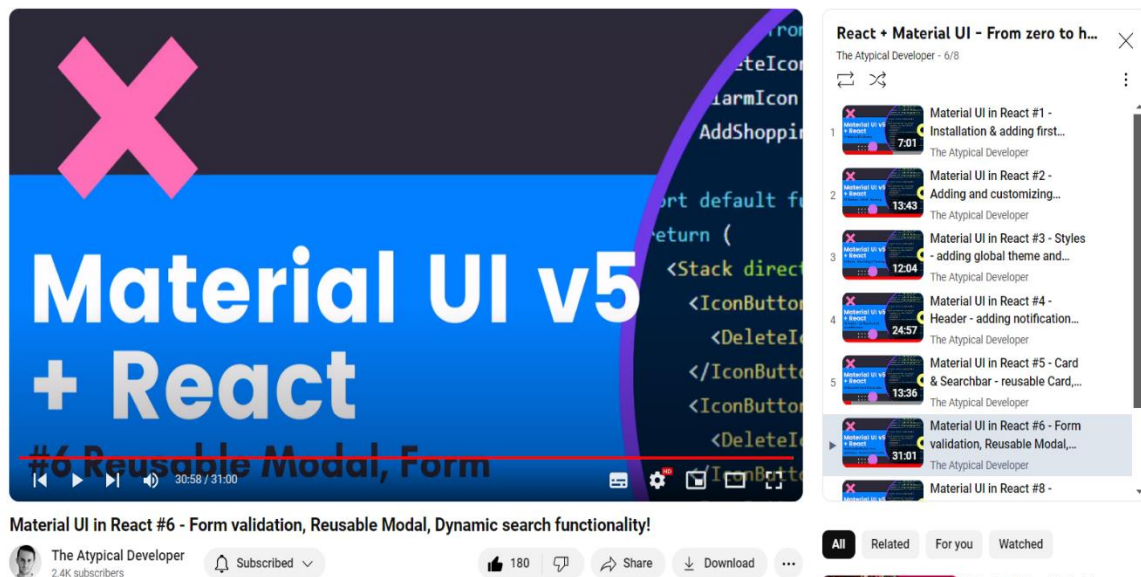
LOGIN

Login Page

## 4. Challenges Faced

### Usage of NASA Open APIs

The main challenge that was faced by me was to create the Responsive User Interfaces with MUI which is New Framework for me to learn and It has lots features like global theme editing and how easy to pass style attributes as props to the components which can use again and again. It also gives readability for the code, easy to handle the responsiveness of the web site, etc... I followed A crash course to learn about this material UI and “The Atypical Developer - Material UI in React” gives me lot help for that.



### Material UI in React Tutorial that was followed for the understanding of the API

After following this tutorial, I was able to know how to use the MUI and how to make a Responsive Web site for my Assignment.

### Responsive User Interface Creation

The next challenge I faced was got to know how to use the APIs that are available in the NASA Open API Library. I went through the documentation provided by NASA regarding the APIs. Initially I did not get a full understanding regarding these APIs. I followed lots of YouTube tutorials that were uploaded by the YouTubers “Benjamin Siegel”, which gave me a proper understanding for the first time.