**Hello!**

Congratulations on moving forward to the next stage in the process with NanoString! We'd like you to write a sample REST-based Web API solution to demonstrate your development and creative thinking skills.

## **Summary**

We would like you to build a REST-based web API and associated datastore that will perform a subset of CRUD operations based on NASA’s Astronomy Picture of the Day (<https://github.com/nasa/apod-api>). We are aware that the API design outlined here is not optimal. This is done intentionally. Rarely are we able to design and develop software in the most optimal way. Often, we must make the best implementation choices with what we are given and in the timeframe with which we need to operate in. You are welcome to give feedback as to how to improve the API design.

## **API Requirements**

1. **GET //api/nasa-astronomy-picture/{dateString}**

RETURNS the NasaAstronomyPictureModel metadata object for the requested *valid* DateString

Requirements:

* The code shall validate the dateString input parameter. If the dateString is not valid, the API shall return a 400 Bad Request. See the Validation Rules section at the end of this document.
* The code shall check to see if there is an *active* record in the datastore that corresponds to the date
  + If an *active* record is found, the code shall return the object in a 200 OK response
  + If an *active* record is NOT found
    - The code shall make an http request to NASA’s API to fetch the Astronomy Picture of the Day metadata object. (documentation can be found at: <https://github.com/nasa/apod-api>).
    - If the Picture of the Day is an image (it could be a YouTube video), the code shall then make subsequent requests to fetch the standard and hd versions of the image and cache those images locally. You may choose to store those images in the database or local filesystem or in-memory datastore.
    - After the successful request to NASA’s API, the code shall persist the Picture of the Day metadata object in the local datastore, assigning a unique ID integer to the object.
      * Note that if the Astronomy Picture of the Day is an image and you’ve cached the images, you’ll need to update the Sd and HdUrl parameters so that the client can request the cached image. See the Image endpoint outlined later in this document.
    - After persistence of the metadata object and images, the code shall return the newly created metadata object in a 200 OK
* The code shall return a 500 Internal Server Error if any Exceptions are thrown

1. **GET //api/nasa-astronomy-picture/**

RETURN a collection of NasaAstronomyPictureModel objects found in the datastore.

Requirements:

* The code shall return a collection of *active* NasaAstronomyPictureModel metadata objects which have been previously persisted in the datastore in a 200 OK response. No need to go and fetch all the raw objects from the NASA API
* The code shall return a 500 Internal Server Error if any Exceptions are thrown

1. **GET //api/images/{imageName}**

RETURNS the image for the requested *valid* imageName

Requirements:

* The code shall validate the imageName input parameter. If the imageName is not valid, the API shall return a 400 Bad Request. See the Validation Rules section at the end of this document.
* The code shall check to see if an image with the imageName exisits
  + If the does not image exist, a 400 Bad Request shall be returned
  + If the does exist, the code shall return the image to the client with the appropriate mime type set in a 200 OK
* The code shall return a 500 Internal Server Error if any Exceptions are thrown

## **Additional Features**

Once you have the requirements for Astronomy Picture of the Day working, implement at least one of the additional items from the list below. Pick the one that you feel most comfortable completing. Doing all of them is not required. You may do as many as you like.

* Add Authentication
* Add Logging
* Add better error/exception feedback for the client besides an HTTP status code
* Add UPDATE (PUT) functionality to update the fields of the APOD data structure. It is not necessary or desirable to also update the image
* Add DELETE (mark as inactive, not physically delete) functionality to the NasaAstronomyPictureModel object in the datastore given an image’s int id
* Add Swagger support
* Add Automated Tests. Feel free to add any testing frameworks or helper libraries you wish. You do not have to use XUnit if there is another framework you feel more comfortable with

## **Evaluation**

When reviewing your submission, our engineers will evaluate it along the following categories:

* Meets Requirements
* C# Code
  + Is it clear?
  + Does it communicate intent?
  + Are comments provided where appropriate?
* Server-side Architecture
  + Separation of concerns
  + Thoughtful directory structure and file names
* Code Quality
  + Documentation
  + Consistent style

## **Submission Instructions**

* Your code challenge is due **one day before your team interview**
* Do not post your response publicly
* Please upload your completed project to:
  + <https://nanostring.app.box.com/f/be96cadd327b41cebfd055380a5641f0>
  + **To make your zip folder as small as possible, please do not include the /bin, /obj, or /packages folders in your zip**

Good luck with your challenge! We are looking forward to seeing the results.

## **Validation Rules**

Input Parameter DateString shall be considered valid if:

* It conforms to a yyyy-MM-dd format
* Is within a date range of today’s date minus two months in the past

Input Parameter ImageName shall be considered valid if:

* It is not null, is not all whitespace, and is length > 4

NasaAstronomyPictureModel object fields

* Copyright – none
* Date – Required. Must be in yyyy-MM-dd format
* HdUrl – Required. Must be a valid, absolute Uri
* Id – Required (except on POST requests)
* MediaType – Required. The two valid strings are “image” and “video”
* SdUrl – Required. Must be a valid, absolute Uri
* Title – Required