ImageAccessCorp.com
22 Paris Ave. Rockleigh, NJ, 07647
201.342.7878

You are tasked with creating a Single-Page application that showcases a simple gallery of images. It's up to you to choose to use any framework/library (AngularJS, Angular, jQuery, TypeScript etc) or even plain JS. The resulting page should work on Chrome (at least) and you are allowed to use the latest supported JavaScript (e.g. ES6+ and all features). You app must cover the following requirements:

- The page will display a simple grid of the thumbnails. Ideally it should dynamically re-arrange the thumbnail sizes and rows as you resize the browser window. The thumbnails must all have the same size (e.g. 100x100 pixels, configurable)
- create a JavaScript **model** that will handle a random number 'n' of thumbnails (e.g. n=20); each thumbnail object will have a unique UUID as identifier
- present each thumbnail in your html with a placeholder img element; this means that when the page is loaded, all *n* thumbnail placeholders will appear with a static placeholder image (be creative with user feedback).
- use the REST API of https://jsonplaceholder.typicode.com/ to:
    - asynchronously and dynamically fetch n random thumbnails URLs from the **/photos** resource
    - fetch the **image data** from the actual files (URLs) and convert the binary data to **base64**
    - assign the created base64 data to its respective **thumbnail model property**(e.g. store them in memory)
- once a thumbnail model has its image data set up (e.g. the base64 data have been loaded), the html **img** element must also update its associated **src** attribute in order to display the thumbnail. Make sure that if the image does not fit the thumbnail, you should resize accordingly by maintaining the width/height ratio
- keep track of the status of each thumbnail **in your model** (*loading*, *loaded*, *error*) and display that status in some way to the user (e.g. overlay in the placeholder image the status, below, etc)
- handle all possible exceptions and possible timeouts gracefully and inform the user
- Create an emulation mode where some errors happen intentionally to show-case how you handle errors and unexpected situations (e.g. create a scenario where half, random, images do not really exist).
- upon the user clicking on a thumbnail, send a POST request to https://jsonplaceholder.typicode.com/posts with its associated UUID as a property and change the border color of the thumbnail to green if the POST was successful and red if not successful
- Make sure that the page can re-initialize itself completely with a 'reload' button, but without actually refreshing
- Use well-structured console.log messages to display the status of the application (e.g. loading, etc)

Provide documentation on how to setup and run the code. If it must be 'compiled', provide both original and compiled versions.