APPROACH FOR IMGUR GALLERY

Requirement Analysis:

From the coding challenge instructions the requirement was analysed as follows

1. To Implement web app for browsing images which will be loaded by consuming the rest api provided by https://api.imgur.com and documentation available in https://apidocs.imgur.com
2. After studying the documentation of imgur.com following things are noted:
   a) API ENDPOINT for gallery images are noted down.
   b) To access the Rest API, Authorization header with client-Id is required and to get client-id, our web application needs to be registered in imgur.com.
   c) To get thumbnail images of various sizes different characters listed as per documentation needs to be included in the actual image url received in the response.
   d) For user to select section, sort, window of images the valid values are noted and needs to be passed as url params.
   e) Data models noted for the type of response will be received in case of success or error.
3. Registered the web application in imgur.com and noted the generated client-Id.
4. Tested the api endpoints using postman with client-id passed as authorization header and analysed the response received.

Implementation:

Environment Setup:

1. Choosen Angular 14 to design and develop the single page application by consuming the Rest API from api.imgur.com
2. Choosen VS CODE for IDE.
3. Installed node latest version 16.
4. Npm gets installed along with node.
5. Installed angular latest version 14.
6. @angular/material Npm package added to project for ui development.
7. Git initiated in the root of local project.
8. Repository created in GitHUB.

Developing Feature:

1. Mat-grid-listing ui component from material used to show thumbnail images in grid pattern
2. Mat-grid-tile-footer used to show description at the bottom of thumbnail image.
3. Mat-select ui component used to provide section, sort, window filters to the user.
4. Image Interface created as per the noted model from documentation.
5. Image Service created to separate the methods to call api endpoints.
6. Environment file created for keeping the api end points and client-Id.

7. Once the response received from api, filtered the response for images only then modified the image url to make thumbnail url as per documentation.
8. Pushed local repository to GitHub remote repository

Testing:

1. Inbuilt spec.ts file used to write Test use cases using Jasmine and karma test compiler for unit testing the component behaviour as per the desired logic.
2. Executed test cases and resolved the failed test cases.

Deployment:

1. Updated local repository to GitHub.
2. GitHub ghpages activated and path given as docs folder to publishing application.
3. Build output directory was updated as docs in angular.json for ghpages to publish content.
4. Cli command ng build used with baseHref flag value with the published link that was created when activated ghpages.
5. Pushed the project to GitHub including the generated docs folder with published files.
6. By accessing the ghpages published link in browser user will be able to access the web application.