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 which will be 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 React 18 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 version 16.
4. Npm gets installed along with node.
5. Installed React version 18.
6. @mui/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. Class component named Thumbnail created.
2. Initial data loading process invoked in ComponentDidMount lifecycle hooks.
3. Once data received filtered the response data and modified link url to get thumbnail url.
4. Using map to iterate the modified image array and image tag used to show thumbnail images.
5. Css display grid used to show thumbnail images in grid pattern

6. bottomDescription css class used to show thumbnail description at the bottom of thumbnail image.
7. Select dropdown used to provide section, sort, window filters to the user.
8. Image Interface created as per the noted model from documentation.
9. Thumbnail Service created to separate the methods to call api endpoints.
10. http-common.ts file created for making axios api call with api base end point and updating header as Authorization with client-Id.
11. Once the response received from api, filtered the response for images only then modified the image url to make thumbnail url as per documentation.
12. Pushed local repository to GitHub remote repository

<u>Testing:</u>

1. Inbuilt test.tsx file used to write Test use cases for unit testing the component behaviour as per the desired logic.
2. Executed test cases and resolved the failed test cases.

<u>Deployment:</u>

1. Updated local repository to GitHub.
2. GitHub ghpages activated and path given as docs folder to publishing application.
3. Npm package gh-pages added for deploying in gh-pages
4. Updated package.json with key homepage and value as Git Repository url.
5. Updated package.json scripts array for deploying to gh-pages.
6. Pushed the project to GitHub Repository master.
7. Deployed to gh-pages using cli command npm run deploy which builds and pushes the build files to gh-pages branch in respository and host the page.