Instructions

Project 7

Create your project

- Use the create-react-app to set up and create your initial directory.

Build your app components

- Use the index.html file and mockups as a general guide while you create the components of this project.
- Use the src/App.js file as your main container component.
- Stateless functional components work well for components that focus on the UI and receive data via props. Some examples of the stateless components you could use for your app are:

    - A Photo component that displays the li and img elements.
    - A Nav component for the apps navigation links.
    - A NotFound component for displaying a user friendly message when the search returns no results

- Stateful class components will be better suited for your search form and photo container where data can be managed with state.

Pro Tip: When building out your components, if you're feeling stuck, it can be helpful to follow along with a course in the unit, pausing the videos as needed so you can build alongside the instructor, but instead of building the course project, try to apply what's in the video to this project.

Get a Flickr API key

- Create *yahoo* account/use *tumblr* account to sign in.
- Apply for a non-commercial API key: https://identity.flickr.com/login?redir=%2Fservices%2Fapps%2Fcreate%2Fapply%2F
- You'll need to set up a config.js file in your project that imports your API key into your application so that you and other users can request data from the Flickr API. This should be imported into src/App.js.

- The config.js file should look something like this:

```
const apiKey = 'YOUR API KEY';
export default apiKey;
```

- Import your API key into your application, preferably into src/app.js, and save it to a variable like you would any other module, and use the variable where applicable. That way, your app's users will only need to enter in an API key once.

Important Note: This config.js file must be listed in the .gitignore file so it won't be committed to your github repository. This will prevent your keys and tokens from getting posted publicly to GitHub. It is very important that you do NOT upload any of your personal API keys / secrets / passwords to Github or other publicly accessible place. When you submit this project for grading, your project reviewer will create their own config.js file and use their own API key to run the project.

Routes

- Install React Router and set up your <Route> and <Link> or <NavLink> elements.
- Include a "Search" link that includes a search field to let users search for photos.
- Clicking a nav link should navigate the user to the correct route, displaying the appropriate info.
- The current route should be reflected in the URL.
- Your app should display at least 3 default topic links that return a list of photos matching some criteria. For example: Sunsets, waterfalls, and rainbows.
- It's okay to request and load the photos for the three default topics when the app first loads. Those default topic pages don't have to re-request and reload new data every time one of those pages are loaded.

Pro Tip: When setting up the routes, if you're feeling stuck, it can be helpful to follow along with a course in the unit that covers routes, pausing the videos as needed so you can build

alongside the instructor, but instead of building the course project, try to apply what's in the video to this project.

Requesting the data

- Fetch the data from the Flickr API using the Fetch API or a tool like Axios.
- Make sure data fetching and state is managed by a higher-level "container" component, like src/App.js.
- It is recommended that you use the following link for help with this part of the project: https://www.flickr.com/services/api/explore/flickr.photos.search
    - Enter a tag to search for, such as "sunsets".
    - You should also limit the number of results to 24 using the per_page argument.
    - Choose JSON as the output, then "Do not sign call."
    - Click "Call Method…" At the bottom of the page, and you'll see an example of the API call you'll need to make. You can click on the URL to see what the response will look like.

Pro Tip: When requesting data from the API, if you're feeling stuck, it can be helpful to follow along with a course in the unit that covers fetching data, pausing the videos as needed so you can build alongside the instructor, but instead of building the course project, try to apply what's in the video to this project.

Search

- Be sure to include a search field feature and a search route to search for new categories of images.

Displaying the data

- Make sure each image gets a unique "key" prop.
- There should be **no console warnings** regarding unique "key" props.
- The title of each page displaying images should be dynamically provided via "props".
- The current route should be reflected in the URL.
- There should be no more than 24 images displayed.

CSS styles

- The mockups are just a general guide for how the elements should be arranged and positioned on the page. But other than general arrangement, spacing, and positioning; you are free to experiment with things like color, background color, font, shadows, transitions, animations, etc.

Add good code comments!

Cross-Browser consistency:

- Google Chrome has become the default development browser for most developers. With such a selection of browsers for users to choose from, it's a good idea to get in the habit of testing your projects in all modern browsers.

Review the "How you'll be graded" section!

Quality Assurance and Project Submission Checklist

- Perform QA testing on your project, checking for bugs, user experience, and edge cases.
- Check off all the items on the ***Student Project Submission Checklist***.

NOTE: Seeking assistance

- **If you're feeling stuck or having trouble with this project**
    - **Reach out to the team on Slack.**

Extra Credit

- Add a loading indicator that displays each time the app fetches new data. Since the data for the three main topic pages can be requested when the page first loads, it's okay if the loading indicator is only present on the search route.
- If no matches are found by the search, display a friendly user message to tell the user there are no matches.
- Include a 404-like error route that displays a friendly 404 error page when a URL does not match an existing route.