30.0 hours
Project Status Passed

Project 7

React Gallery App

Exceeds Expectations

Your project was reviewed and looks great! Take a look over your grade details to see how you can improve.

Keep Going

- Getting Started
- Instructions
- How You'll Be Graded
- Your Grade

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question on Slack or in the Treehouse Community.

13 steps

- ## Instructions

  Markdown to describe steps the student must complete to finish the project with a "Meets Expectations" grade.

- **Create your project**

  - Use the create-react-app to set up and create your initial project 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 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 components are well suited for your search form and photo container where data can be managed with state.

  The difference between stateful and stateless components can be confusing but in simple terms, one has state and one doesn't. Stateful components have state and can track changing data. Stateless components do not have state and simply use the data that is passed to them via props.
  NOTE: When you navigate the app with the nav links or the browser's forward and back buttons, the URL in the address bar should show the correct route, and the correct data, which you'll request in the next step, should display on the page.

- **Get a Flickr API key**

  - Create/use a Yahoo or Tumblr account to sign in.

- Apply for a non-commercial API key.
- 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 an API key once.

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.

NOTE: When your routing is setup correctly, App.js will have a `Switch` element, and nested inside of that, you will have separate `Route` tags for each of your three main topics.
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 are 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.

NOTE: When you're correctly fetching your data from a container component and passing it down to the display component, and your routing is correctly setup, the URL in the address bar will always match what is displayed on the page regardless of whether you use the nav buttons, the browser's forward and back buttons, or paste a URL directly into the address bar.

- **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 of 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.
    - Review material in the unit.
    - Practice your Google skills by finding different ways to ask the questions you have, paying close attention to the sort of results you get back depending on how your questions are worded.

NOTE: What you submit is what will get reviewed.

- When you submit your project, a snapshot is taken of your repository, and that is what the reviewer will see. Consequently, any changes you make to your repo after you submit will not be seen by the reviewer. So before you submit, it's a smart idea to do a final check to make sure everything in your repo is exactly what you want to submit.

Extra Credit

To get an "exceeds" rating, complete all of the steps below:

4 steps

- **Browser navigation works for the search route**

    - The app should keep track of browser history and change the page's data based on the current URL. Clicking the browser's forward and back buttons should navigate the user through all search history, keeping the URL and fetched data in sync.

- **404 Error**

    - Include a 404-like error route that displays a friendly 404 error page when a URL does not match an existing route.

- **Loading Indicator**
    - 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.
- **No Matches Message**
    - If no matches are found by the search, display a friendly user message to tell the user there are no matches.

- **NOTE: Getting an "Exceed Expectations" grade.**
    - See the rubric in the "**How You'll Be Graded**" tab above for details on what you need to receive an "Exceed Expectations" grade.
    - Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a “Meets Expectations” grade, you won’t get a second chance. Exceptions can be made for items that have been misgraded in review.
    - Always mention in the comments of your submission or any resubmission, what grade you are going for. Some students want their project to be rejected if they do not meet all Exceeds Expectations Requirements, others will try for all the "exceeds" requirement but do not mind if they pass with a Meets Expectations grade. Leaving a comment in your submission will help the reviewer understand which grade you are specifically going for

Download files

Zip file

Project Resources

Workshop

Create Apps with No Configuration (create-react-app)

External Link

React Router

Workshop

Data Fetching in React

External Link

flickr.photos.search – Endpoint

External Link

flickr.photos.search – API Explorer

External Link

Photo Source URLs – Flickr API

https://www.npmjs.com/package/prop-types

prop-types npm package

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help