Lab 5

- Due Friday by 11:59pm
- Points 100

CS-554 Lab 4

React SpaceX API

For this lab, you will create a SpaceX API Single Page Application using React and Vite. **For this Lab you should use Function Components with the useState (https://react.dev/reference/react/useState) and useEffect (https://react.dev/reference/react/useEffect) Hooks along with React Router (https://reactrouter.com/en/main)**

You will be creating a Single Page Application using React and React Router Dom that implements the following routes.

You will be using the **SpaceX API (https://github.com/r-spacex/SpaceX-API/blob/master/docs/README.md)** . You do not need an API key because we are doing just GET requests to the API. The API only requires authentication if you try to update or delete a resource in the API. You will use the following end-points for your React components: **Launches (https://api.spacexdata.com/v4/launches)** , **Payloads (https://api.spacexdata.com/v4/payloads)** , **Rockets (https://api.spacexdata.com/v4/rockets)** , **Ships (https://api.spacexdata.com/v4/ships)** , **Launch Pad (https://api.spacexdata.com/v4/launchpads)** , and **Cores (https://api.spacexdata.com/v4/cores)** listings. Please look at the data returned so you know the schema of the data and the objects it returns. These endpoints give you the lists, you can get an individual item using the same base URL's and adding a /:id parameter. For example: **https://api.spacexdata.com/v4/launches (https://api.spacexdata.com/v4/launches)** gives your the list of launches, **https://api.spacexdata.com/v4/launches/5eb87cd9ffd86e000604b32a (https://api.spacexdata.com/v4/launches/5eb87cd9ffd86e000604b32a)** Gives you the data for a single launch with an ID of: 5eb87cd9ffd86e000604b32a. Same applies to payloads, rockets, ships and launchpads.

It's important to recognize the interconnected nature of the data we're dealing with. Take launches, for instance; they're not standalone events. Each launch is tied to an array of ship IDs, a Rocket ID, a Launchpad ID, and an array of payload IDs, a core ID. So, when presenting the details of a launch,

it's important to provide links to the related components for each ID referenced in the data. Similarly, launch pads are linked to arrays of launch and rocket IDs and there are various links in the other end points referencing id's in the other endpoints, forming an intricate web of connections. Whenever an ID appears in the data being rendered for any of the endpoints utilized in the assignment, it should be linked to its corresponding details route/component.

Do NOT just dump raw JSON to the page, if you do, MAJOR points will be deducted. Render all the data in proper JSX elements that make sense for the data you're displaying!

Pages

/

The root directory of your application will be a simple page explaining the purpose of your site (to talk about SpaceX, the API etc.. the /history endpoint of the API gives you some nice SpaceX history, the /company endpoint of the API gives you some company info).

This page will have a `<Link>` to the Launches Listing (`/launches/page/0`), The Payloads Listing (`/payloads/page/0`), The Cores Listing (`/cores/page/0`), The Rockets Listing (`/rockets/page/0`), The Ships Listing (`/ships/page/0`), and the Launch Pads Listing (`/launchpads/page/0`)

/launches/page/:page

This route will render a paginated list of launches. You will display 10 launches per page. It will use the :page param to determine what page to request from the API. If you are on page 0, you will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more Launches in the list, the SPA will display to a 404 "page".** For this list you can just display the `name` and `flight_number` fields at least and perhaps one of the images from the data.

/launches/:id

This route will show details about a single launch. **If the launch does not exist, the SPA will display to a 404 "page".** You should show as much data as possible from the API. There is a launch image, a YouTube webcast of the launch and many other details about the launch. . DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of

the page, any lists should be displayed as UL/OL.). You should display as much data about the launch as possible. Any ID's related to your other endpoints should link to your details component for that endpoint. For example, in the launch with an ID of 5eb87d30ffd86e000604b378 there is a rocket in the data with an ID of 5e9d0d95eda69973a809d1ec, that should link to your /rockets/:id route/component. You should link ALL related data, there is an array of ship ids, array of payload ids, and launchpad ID. (**This goes for all your other :id routes/components below as well**)

/payloads/page/:page

This route will render a paginated list of payloads. You will display 10 payloads per page. It will use the :page param to determine what page to request from the API. If you are on page 0, you will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more payloads in the list, the SPA will display to a 404 "page".** For this list you can just display the `name` field.

/payloads/:id

This route will show details about a single payload. **If the payload does not exist, the SPA will display to a 404 "page".** DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of the page, any lists should be displayed as UL/OL.). You should display as much data about the payload as possible.

/cores/page/:page

This route will render a paginated list of cores. You will display 10 cores per page. It will use the :page param to determine what page to request from the API. If you are on page 0, you will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more cores in the list, the SPA will display to a 404 "page".** For this list you can just display the `serial` field at least and perhaps one of the images in the data.

/cores/:id

This route will show details about a single core. **If the core does not exist, the SPA will display to a 404 "page".** DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of the page, any lists should be displayed as UL/OL.). You should display as much data about the core as possible.

/rockets/page/:page

This route will render a paginated list of rockets. You will display 10 rockets per page(There are less than 10 for this endpoint, but you should still implement pagination in case the data in the API ever grows and there are more than 10 rockets). It will use the :page param to determine what page to request from the API. If you are on page 0, you will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more rockets in the list, the SPA will display to a 404 "page". for example, since there are only 4 rockets currently, going to /rockets/page/10 should display a 404.** For this list you can just display the `name` field at least and perhaps one of the images in the data.

/rockets/:id

This route will show details about a single rocket. **If the rocket does not exist, the SPA will display to a 404 "page".** DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of the page, any lists should be displayed as UL/OL.). You should display as much data about the rocket as possible.

/ships/page/:page

This route will render a paginated list of ships. You will display 10 ships per page. It will use the :page param to determine what page to request from the API. If you are on page 0, you will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more ships in the list, the SPA will display to a 404 "page".** For this list you can just display the `name` field.

/ships/:id

This route will show details about a single ship. **If the launch does not exist, the SPA will display to a 404 "page".** DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of the page, any lists should be displayed as UL/OL.). You should display as much data about the ship as possible.

/launchpads/page/:page

This route will render a paginated list of launch pads. You will display 10 launch pads per page (There are less than 10 for this endpoint, but you should still implement pagination in case the data in the API ever grows and there are more than 10 launch pads). . It will use the :page param to determine what page to request from the API. If you are on page 0, yyou will show a button to go to the *next* page and not show the button to go to the previous page. If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page. If you are on the last page, you will only show the previous button. **If the Page does not contain any more Launch Pads in the list, the SPA will display to a 404 "page".** For this list you can just display the `name` field at least and perhaps one of the images in the data.

/launchpads/:id

This route will show details about a single launch pad. **If the launch pad does not exist, the SPA will display to a 404 "page".** DO NOT just dump raw JSON to the page, you must render all the data in an element that makes sense for the data. (image in img tags, it would be really cool to embed the YouTube video of the launch as the center focus of the page, any lists should be displayed as UL/OL.). You should display as much data about the launch as possible.

Pagination

DO NOT hardcode page numbers! You will lose major points if you do!!!

The minimum you must provide for a pagination UI:

- **If you are on page 0, you will show a button to go to the *next* page. DO NOT show the previous button**
- **If you are on page 1+, you will show a *next* button and a *previous* button, that will take you to the previous page.**
- **If you are on the last page, you will show the previous button to go to the previous page.**

DO NOT show the next button!

There are a couple of ways you can handle pagination. You can just grab all the data from the endpoints and then paginate through it on the client-side. The API supports pagination as well, but it requires sending query parameters in the request body of the request. Please see the API's **documentation on API queries (https://github.com/r-spacex/SpaceX-API/blob/master/docs/queries.md)** for more information on pagination if you choose to use that method. **With queries, you can even populate the related data (Like launches having a payload id, so you can query the API to get the launch, and perhaps the name of the payload names along with the launch instead of just the IDs. Checkout the "populate" option in the documentation linked about queries)**

HTML, Look, and Content

Besides the specified settings, as long as your HTML is valid, the general look and feel is up to you. Feel free to have fun with it and you should have fun with it! You are free to use any package and design this in any way you like as long as it fulfills all the requirements stated. As far as the components, you should try to reuse as many components as you can. The lists for example, you could make one generic list component that just renders differently based on the different data being passed to it as a prop. You should try to reuse components as much as you can!

Extra Credit

There are two chances to get extra credit:
+5 points if you go all out on the design and make a really beautiful application that wows the TA grading your assignment!
+5 points if you utilize the query options in the API for displaying related data. For example, on the launch details page, The payload is an array of payload ID's, you can use the populate option in a query option to get the payload names with the launch details instead of the ID's. To get the extra credit, you must do this for ALL related data in the components.
+5 points if you add search capability to search launches, payloads and cores by their `name` (for launches and payloads) or `serial` (for cores) fields.

General Requirements

1. Remember to submit your `package.json` file but **not** your `node_modules` folder.
2. Remember to run HTML Validator!

3. Remember to have fun with the content.
4. Remember to practice usage of async / await!