Custom Frontend Criteria

Stage 1. Markup and JSX + Connecting a Third-Party API

Project Rejected Without Review

- The pull request was not sent for review.
- The markup was not ported into JSX.
- Data from the API doesn't load or appear.
- There are errors when building or running the project.
- The frontend part of the application is not deployed to a remote server. For Stage 1, GitHub Pages is acceptable. For Stage 3, Google Cloud is required.

Performance Criteria

Markup and JSX

- The project is adapted for different screen resolutions and looks correct on all intermediate resolutions. There is no horizontal scrolling on resolutions of `320px` and up. The scrollbar can't be hidden using the `overflow: hidden` declaration. **/ 7.4**
- Elements specified in the UI toolkit are used correctly (e.g., buttons, forms, text elements, etc). **/ 7.26**
- Classes are named according to BEM specifications. **/ 3.63**
- Semantic HTML is used, meaning that semantic tags are used. All elements are used correctly (e.g., a paragraph must be a paragraph, a list must be a list). The DOM tree structure doesn't consist only of `<div>` containers. **/ 3.63**
- The correct approach for positioning elements has been chosen and is described using the correct syntax. For example, when positioning an element absolutely, its parent block is relatively positioned. **/ 3.63**

- `flex` or `grid` layouts are used to arrange elements whenever applicable. **/ 3.63**
- Infrastructural project files are created using Vite or CRA. **/ 3.63**
- The markup has been ported into JSX:
    - The markup is included within `()`. **/ 3.63**
    - The markup has been moved to the corresponding components. **/ 3.63**
- Your project contains the following: **/ 3.63**
    - An `images` folder
    - A `components` folder with JS and CSS files for the components
    - A `fonts` folder
- There are no warnings. **/ 3.63**

React and connecting a third-party API

- Routes:
    - There are two separate routes: the root route (`/`) and one other (e.g., `/profile` or `/about`). **/ 3.63**
    - Navigation between pages and links to external resources is working correctly. There are no broken links or links leading to an empty anchor on the page, and external links open in a new tab. All project pages can be accessed, and there are no hidden blocks. **/ 3.63**
- React components:
    - Hooks are not used inside conditional statements. **/ 3.63**
    - Hooks are called in a component's main function. **/ 3.63**
    - For class components, the effects are described inside the component lifecycle methods. **/ 3.63**
- Asynchronous API requests: **/ 7.26**
    - Requests can be made through the Fetch API or by using XMLHttpRequest. Third-party libraries (such as axios or jQuery) are not used.
    - API requests are contained in a separate file.

    - The chain for processing promises ends with a `catch()` block.
    - The first `then()` handler returns `res.json`.
- The project complies with the following code style requirements: **/ 3.63**
    - camelCase is used for function and variable names.
    - Only nouns are used as variable names.
    - Variable names clearly describe what is stored in them. If the project has several variables with similar data, then those variables have unique but descriptive names.
    - Descriptive names are used for functions, which reflect what they do.
    - Function names start with a verb.
    - JS classes and functional components are named using nouns and start with a capital letter.
    - Names must not include inappropriate or unclear abbreviations.
    - Custom hook names start with `use`.
- No third-party JavaScript libraries are used, unless absolutely necessary. If third-party libraries are connected, then they are used appropriately. **/ 3.63**

Best Practices

Markup and JSX

- The design reuses components wherever possible. **/ 1.25**
- Fonts are connected using `@font-face`. **/ 1.12**
- The icons are exported in SVG format. **/ 1.12**
- Form elements should be highlighted when focused. **/ 1.15**
- The form must have placeholders, and there are required properties for the fields. **/ 1.25**
- The use of `reset.css` is prohibited. **/ 1.12**

React and Connecting a Third-Party API

- The initial state of state variables contains the correct data type. **/ 1.25**
- API requests are described inside the `App` component. **/ 1.25**
- There is no memory leak when hanging handlers. All handlers added with `addEventListener` are removed when the component is unmounted. **/ 1.25**
- API error handling: **/ 1.25**
    - The user receives a message in the case of an error.
- Non-variable values (hard-coded constants) are named in all capital letters and stored in a separate configuration file. **/ 1.25**
- If you are using popups, then correct the opening and closing functionality is implemented. Popups can be closed using the cross button, overlay, or ESC key. **/ 1.25**

Recommendations

Markup and JSX

- System fonts are connected as alternatives to each of your fonts. **/ 0.83**
- Images have an `alt` attribute containing appropriate values. **/ 0.83**
- Raster and vector images have been optimized. **/ 0.83**

React and Connecting a Third-Party API

- For internal links in the application, components from the `react-router` library are used. **/ 0.83**
- Semantically correct blocks are used for components. No `<div>` or other unnecessary HTML tags are used for components that consist of single-level blocks. **/ 0.83**
- The code is clean and easy to understand: **/ 0.85**
    - The code is readable and clearly structured. Some parts of the code are explained with comments if needed.
    - There is no extra code: for example, when a variable is declared but not used, or there is some kind of redundant logic.

    - The code is formatted in the same way, and the indentation hierarchy is respected.