News Explorer Criteria

Stage 1. Frontend and NewsAPI

Project Rejected Without Review

- The pull request for review was not sent.
- The markup has not been ported into JSX.
- One or more sections are missing.
- There are errors when building or running the project.
- Interactions with the news API are not working.
- The `/saved-news` route is missing or inaccessible.
- 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

- All the blocks from the brief have been coded. The layout matches the designs for all resolutions from the mockup. Differences in size should not visually contradict the layout or get out of the grid in the layout. `/ 12.8`
- 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. `/ 12.6`
- Navigation between pages and links to external resources are 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 hidden blocks can be displayed. `/ 12.6`
- Classes are named according to BEM specifications. `/ 4.2`

- 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. **/ 4.2**
- 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. **/ 4.2**
- `flex` or `grid` layouts are used to arrange elements whenever applicable. **/ 4.2**
- Infrastructural project files are created using Vite or CRA. **/ 4.2**
- The popup windows have opening and closing functionality implemented. **/ 4.2**
- The markup has been ported into JSX:
  - The markup is included within `()`. **/ 4.2**
  - The markup has been moved to the corresponding components. **/ 4.2**
- Your project contains the following: **/ 4.2**
  - An `images` folder
  - A `components` folder with JS and CSS files for the components
  - A `fonts` folder
- There are no warnings. **/ 4.2**

Best Practices

- Buttons, input fields, and links are implemented in all of the states specified in the design. **/ 2.16**
- The design reuses components wherever possible. **/ 2.14**
- Fonts are connected using `@font-face`. **/ 2.14**
- The icons are exported in SVG format. **/ 2.14**
- Form elements should be highlighted when focused. **/ 2.14**
- The form must have placeholders, and there are required properties for the fields. **/ 2.14**
- The use of `reset.css` is prohibited. **/ 2.14**

Recommendations

- System fonts are connected as alternatives to each of your fonts. `/ 1.67`
- Images have an `alt` attribute containing appropriate values. `/ 1.67`
- Raster and vector images have been optimized. `/ 1.66`

Stage 2. Backend

Project Rejected Without Review

- The API doesn't register new users when valid data is input.
- The application doesn't start upon using the `npm run dev` command once all required dependencies have been installed.
- The backend part of the application is not deployed to the server.
- There must not beare code snippets that have been plagiarized.

Performance Criteria

- The repository contains all the necessary infrastructure files: `/ 3.64`
  - `package.json`;
  - `.editorconfig`;
  - `.eslintrc` is required for extending the configuration of `airbnb-base`
    - `package.json` contains the `devDependencies` needed for the linter to work correctly
    - An exception for `_id` is added
    - The following rules are forbidden: `eslint-disable`, `eslint-disable-line`, and `eslint-disable-next-line`
  - `.gitignore` should contain at least the `node_modules` folder
- There are no lint errors. When running `npx eslint .` they should be absent. `/ 3.64`
- The `scripts` section of the `package.json` file contains the following: `/ 3.64`

    - An `npm run start` command that starts the server on `localhost:3000`.
    - An `npm run dev` command that starts the server on `localhost:3000` with hot reloading.
- The following routes function as described: `/ 21.81`
    - A request to `GET /users/me` returns information about the user (email and name).
    - `POST /signup` creates a user with the data passed inside the request body.
    - `POST /signin` returns a JWT when the correct email and password are passed in the request body.
    - `GET /articles` returns all articles saved by the user.
    - `POST /articles` creates an article with the data passed inside the request body.
    - `DELETE /articles/:articleId` deletes the saved article using `_id`.
- Users can't delete saved articles from other user profiles. `/ 3.64`
- All routes are protected with authorization, except for `/signin` and `/signup`. `/ 3.64`
- User routes and article routes are described in separate files. `/ 3.64`
- API errors are handled: `/ 7.28`
    - Error status codes are used: 400, 401, 403, 404, 409, 500.
    - If something is wrong with the request, the server returns a response with an error message and a corresponding status.
    - The error message matches its type.
    - Asynchronous handlers end with a `catch()` block.
    - The API does not return standard database or Node.js errors.
- Safe password storage has been implemented: `/ 3.64`
    - Passwords are stored in a hashed form.
    - The API does not return a password hash to the client.
- Validation implemented correctly: `/ 7.28`

    - Requests are validated before being passed to the controller. The body and (where applicable) headers and parameters are checked against the corresponding schemas. If a request doesn't match the schema, the processing is not passed to the controller and the client receives a validation error.
    - Data is validated before being added to the database.
- In production mode, the database address is taken from `process.env`. `/ 7.28`
- The server can be accessed via HTTPS using the domain specified in `README.md`. `/ 3.64`
- Storing the secret key for creating a JWT is implemented correctly: `/ 7.28`
    - In production (i.e., when deployed to the server), the secret key should be stored in a `.env` file, and this file should not be added to Git.
    - In development (i.e., when run on localhost), the code should run and function correctly, even though there is no `.env` file present.

Best Practices

- All routes are connected to the `index.js` file, which is located in the `routes` folder, and `app.js` contains one main route handled by `routes`. `/ 2.14`
- Asynchronous operations are implemented using promises or async/await. `/ 2.14`
- Validation is described in a separate module. `/ 2.14`
- Logging is set up: `/ 2.14`
    - All requests and responses are logged to the `request.log` file.
    - All errors are logged to the `error.log` file.
    - Log files aren't added to the git repository.
- Errors are handled by a centralized handler. `/ 2.14`
- Centralized error handling is described inside a separate module. `/ 2.14`
- The application API is located in an `/api` subdirectory or on an `api.` subdomain (not just name.zone): `/ 2.14`

- Correct: domain-name.tk/api or api.domain-name.tk
- Incorrect: domain-name.tk

Recommendations

- For API errors, classes have been created to extend the `Error` constructor. `/ 1`
- The Helmet module is used to set security-related headers. `/ 1`
- Configuration and constants are stored in separate files: `/ 1`
  - The Mongo server address and the private key for the JWT in development mode are stored inside a separate configuration file.
  - Application constants (response and error messages) are stored inside a separate file with constants.
- A rate limiter is set up: the number of requests from a single IP address is limited to a particular value in a given amount of time. `/ 1`
- The rate limiter is configured in a separate file and imported into `app.js`. `/ 1`

Stage 3. Authorization with React

- The pull request was not sent for review.
- Data from the API doesn't load or appear.
- Any of the functionality required for this stage does not work (i.e., the news search API, authorization/registration, saving and/or deleting an article, or accessing an article causes error logs in the console).
- There are errors when building or running the project.
- The application is not deployed to the server.
- The project functionality is fully implemented according to the current stage's requirements: `/ 22.04`
  - **General**
    - All project links and buttons are functioning.

- Both header states function correctly. If the user is not logged in, the header should have the "Sign in" button; and if the user is logged in, there should be no "Sign in" button. The "Saved articles" link, along with a "Log out" button, should appear in its place.
- If the user closes the tab and then returns to the site, data is taken from local storage upon mounting the `App` component.

- **"Sign up" and "Sign in" pages**
  - When clicking on the "Sign up" button in the "Sign up" popup window, a request is sent to the `/signup` route, provided that all input fields have been filled in correctly. If the request is successful, a popup should appear, which informs the user that they are registered and offers to log them in.
  - When clicking on the "Sign in" button, provided that all input fields have been filled in correctly, a request is sent to the `/signin` route. If the request is successful, the popup is closed.
  - Until there is valid data in all the input fields on the "Sign up" and "Sign in" pages, the form submit buttons remain disabled.
  - All forms are validated on the client side.
- **Search for cards**
  - The preloader is visible and spinning during request execution.
  - Once the search form has been submitted successfully, a block with results appears. If nothing was found, the message "Nothing found" appears.
  - 3 cards are displayed in the results block. Clicking on the "Show more" button will render the next 3 cards.
  - When an unauthorized user clicks on the icon to save an article, the registration popup window opens.
  - After the user has been authorized, the "Save" icon becomes active in the card block.

        - When clicking the "Save" icon in the card block, a request is made to the /articles route of the API.
        - When clicking on the active "Save" icon in the card block, a request to delete the card is made. After a successful request, the card is removed from the "Saved articles" page.
    - **"Saved Articles" page**
        - The "Saved articles" page displays the following: username, number of articles saved, and the keywords by which articles were found.
        - Card blocks in the "Saved articles" page contain the keyword by which the card was found and the trash can icon for deleting the article.
        - When clicking on the trash icon of the card in the "Saved articles" page, a request to delete the card is made. After a successful request, the card is removed from the "Saved articles" page.
- Registration and authorization:
    - The `/saved-news` route is protected using the `ProtectedRoute` HOC component. `/ 2.52`
    - When trying to access the protected route, unauthorized users are redirected to `/` with an open authorization popup window. `/ 2.52`
    - After successful authorization, the popup window is closed. `/ 2.52`
    - The `/` route is not protected. `/ 2.52`
    - If the user was logged in and closed the tab, they can return directly to any page of the application by the URL, except for the login and registration pages. `/ 2.52`
    - After a successful `onSignOut()` handler call, the user is redirected to `/`. `/ 2.52`
    - The `useHistory()` hook is used correctly. `/ 2.52`
    - The components `<Switch />`, `<Route />`, and `<Redirect />` are used correctly. `/ 2.52`
- The interaction with the JWT token works correctly:
    - The JWT token is stored in localStorage. `/ 2.52`

    - The JWT token is validated by a request to the server, not just local storage. / 2.52
    - When you log out of the account, JWT is deleted. / 2.52
- A global state variable has been created that stores user data. / 2.52
- Components:
    - Hooks are not used inside conditional statements or loops. / 2.52
    - Hooks are called in a component's main function. / 2.52
    - For class components, effects are described inside the component lifecycle methods. / 2.52
    - For list items, a unique id is used instead of an array index. / 2.52
    - All components that use information from the currentUser object should get that information from the CurrentUserContext instead of passing it in as props. / 2.52
    - The context is embedded in the `App` component via `CurrentUserContext.Provider`. / 2.52
    - A state variable has been created in the root App component that stores user data. This variable is used as a value of the context provider. / 2.52
- Asynchronous API requests: / 5.04
    - 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: / 2.52
    - 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. **/ 2.52**
- The initial state of state variables contains the correct data type. **/ 2.5**
- The API request for information about the user and for the array of cards is made once during mounting. **/ 2.5**
- API requests are described inside the `App` component. **/ 2.5**
- There is no memory leak when hanging handlers. All handlers added with `addEventListener` are removed when the component is unmounted. **/ 2.5**
- API error handling: **/ 2.5**
  - 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. **/ 2.5**
- Components from the `react-router` library are used for internal links in the application. **/ 1.66**
- Semantically correct blocks are used for components. No `<div>` or other unnecessary HTML tags are used for components that consist of single-level blocks. **/ 1.66**
- The code is clean and easy to understand: **/ 1.68**
  - 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.