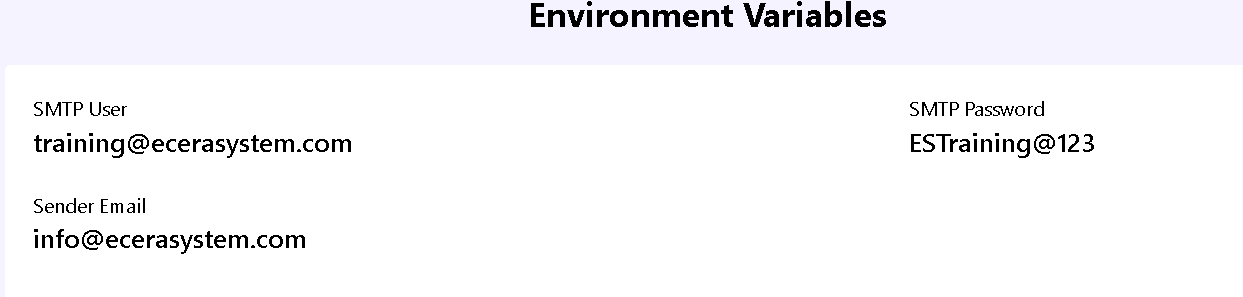
**Env Variable** [**https://citria.org/admin/env-variables**](https://citria.org/admin/env-variables)

This is a React component that manages different views and operations related to environment variables configuration. Here is what it does:

1. Defines state:
   * currentPage - tracks which view is active
   * envVar - stores fetched env var data
   * loading - for api request loading
2. fetchEnvVar - makes API call to get existing env var and sets state
3. Renders conditionally based on currentPage value:
   * ShowEnvVariables: Displays env vars
   * StoreEnvVariables: Form to store new vars
   * EditEnvVariables: Form to edit existing vars
4. showInfoFun - callback to refetch vars and update state after storing/editing
5. showAnotherScreen - utility to change active view by changing currentPage state
6. Calls fetchEnvVar on initial render to get existing env vars if any

So in summary, it is handling all the state management to conditionally render different env var views and reusable components. The components each handle a single task while this oversees the high level control flow.

**Show Env Variable**

****

Import Statements:

javascript

Copy code

**import React from 'react';**

**import PageTitle from '../../Pages/Shared/PageTitle';**

These statements import the necessary dependencies. React is imported because the code is written in React, and PageTitle is imported from a file located at ../../Pages/Shared/PageTitle. It's a custom component for rendering a page title.

Functional Component:

javascript

Copy code

function ShowEnvVariables({ envVar, showEdit }) {

This defines a functional component named ShowEnvVariables. It takes two props, envVar and showEdit.

Function inside Component:

javascript

Copy code

const editFormOpener = () => {

showEdit('edit');

};

This is a function editFormOpener that calls the showEdit prop with the argument 'edit' when invoked. It seems to be related to opening an edit form.

JSX (React Markup):

javascript

Copy code

return (

<div className='h-full flex column flex-wrap justify-center align-middle'>

<PageTitle title='Env variables' />

<h1 className='text-2xl font-bold p-5'>Environment Variables</h1>

<div className='w-full flex flex-wrap justify-between bg-white p-5 rounded relative'>

<button

onClick={editFormOpener}

className='editBtn absolute right-5 top-5 bg-blue-500 text-white px-2 py-1 rounded'

>

Edit

</button>

{/\* Three sections displaying environment variables \*/}

{/\* SMTP User \*/}

<div className='labelInfo w-full md:w-[35%] mb-5'>

<h3 className='text-sm'>SMTP User</h3>

<h2 className='text-lg font-semibold'>

{`${envVar.smtpUser}`}

</h2>

</div>

{/\* SMTP Password \*/}

<div className='labelInfo w-full md:w-[35%] mb-5'>

<h3 className='text-sm'>SMTP Password</h3>

<h2 className='text-lg font-semibold'>{envVar.smtpPassword}</h2>

</div>

{/\* Sender Email \*/}

<div className='labelInfo w-full md:w-[35%] mb-5'>

<h3 className='text-sm'>Sender Email</h3>

<h2 className='text-lg font-semibold'>{envVar.senderEmail}</h2>

</div>

</div>

</div>

);

The return statement contains JSX code defining the structure of the component. It includes a title, an "Edit" button, and three sections displaying different environment variables (SMTP User, SMTP Password, and Sender Email). The actual values of these variables are taken from the envVar prop.

Export Statement:

javascript

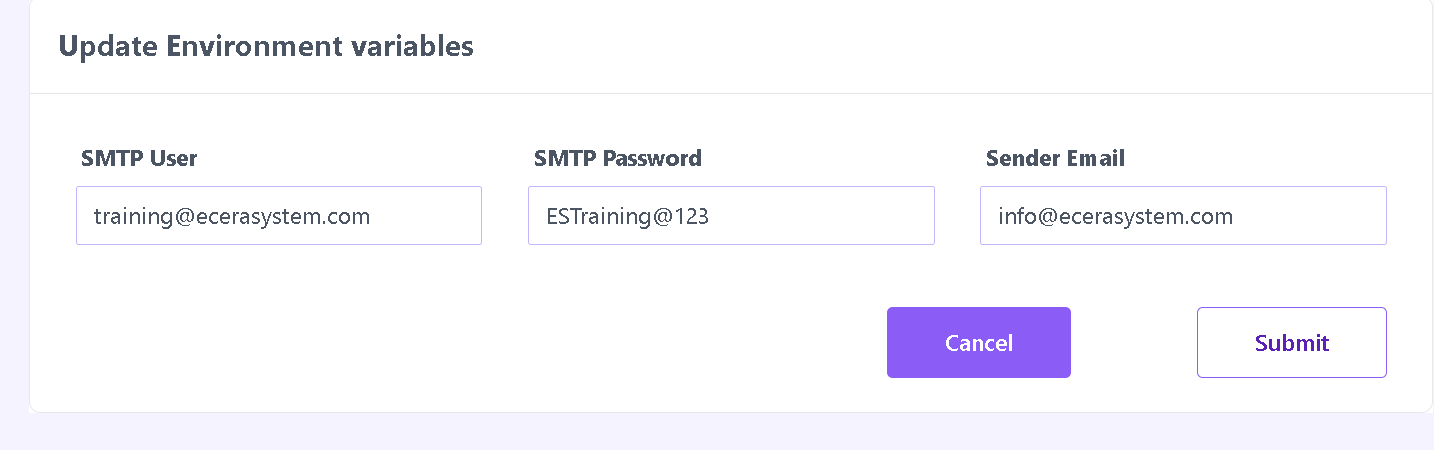
Copy code

export default ShowEnvVariables;

This exports the ShowEnvVariables component, making it available for use in other parts of the application.

In summary, this component is designed to display and potentially edit SMTP-related environment variables. The actual editing functionality seems to be triggered by clicking an "Edit" button, which calls the editFormOpener function passed as a prop.

**Edit Env variables**



The EditEnvVariables file is a React component that serves as a form for updating environment variables related to SMTP configuration. Here's a summary of its key functionalities:

State Management:

Utilizes React hooks (useState) to manage state variables such as formData, loading, and formErrors.

Context Usage:

Utilizes the React context API (useContext) to access the showToast function from a context provider.

Data Fetching:

Uses the useEffect hook to fetch environment variables from the server on component mount.

Axios is used to make an HTTP GET request to the server endpoint for fetching environment variables.

Form Handling:

Updates the form state (formData) on user input through the handleChange function.

Implements form validation through the validateForm function to ensure essential fields like SMTP User, SMTP Password, and Sender Email are not empty.

Sends an HTTP PUT request to the server to update environment variables on form submission.

Displays success or error messages based on the server response using the showToast function.

If validation fails, displays relevant error messages and prevents form submission.

UI Rendering:

Renders a form with input fields for SMTP User, SMTP Password, and Sender Email.

Utilizes React Icons (BsExclamationCircleFill) for displaying error icons.

Displays a loading spinner (Spinner component) during data fetching and form submission.

Navigation:

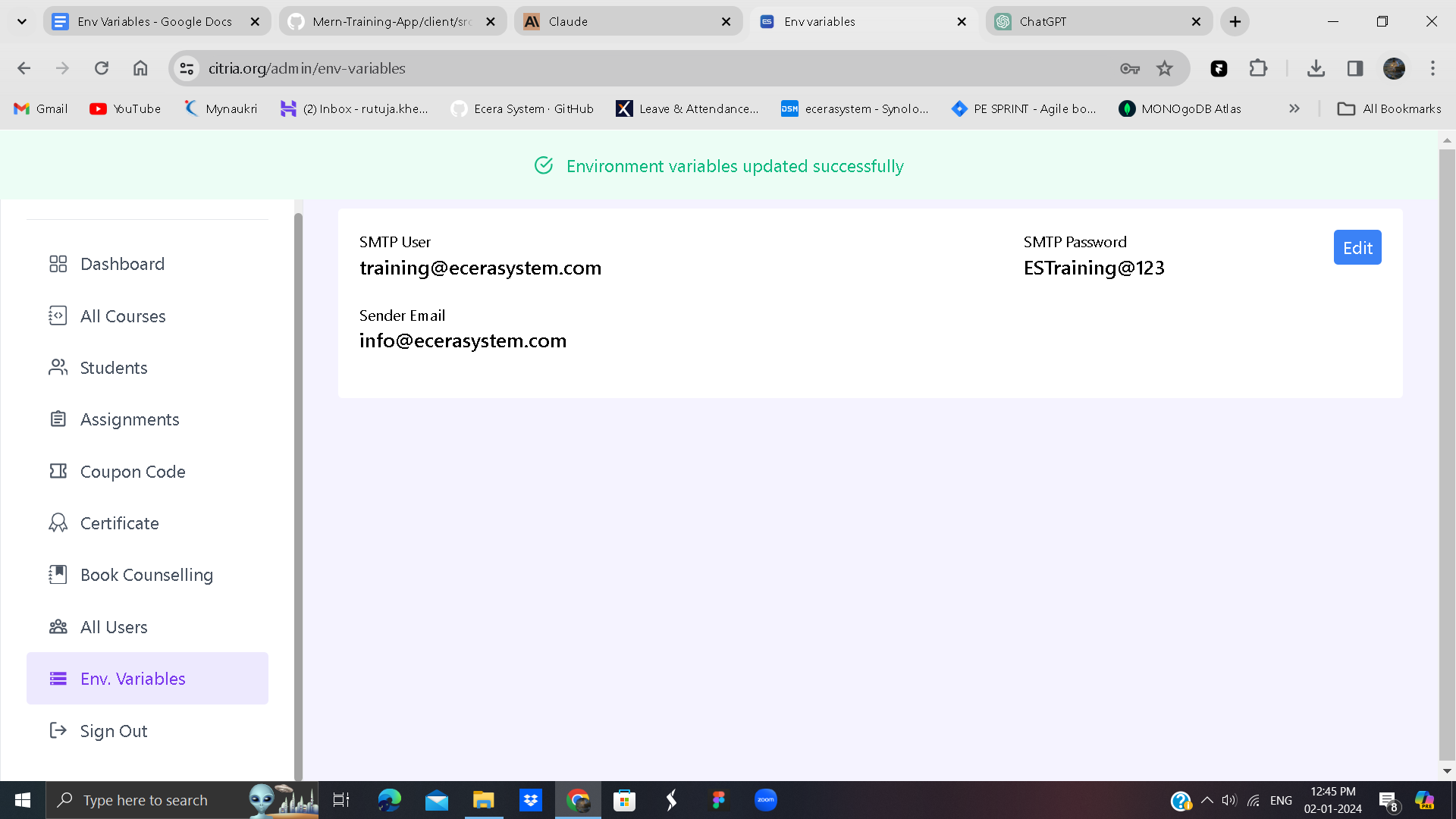
Uses the useNavigate hook from React Router to navigate back to the previous page (showInfo("fetch")) on cancel button click.

Styling:

Applies responsive styling using Tailwind CSS classes for a clean and user-friendly interface.

Overall, this component provides a structured and user-friendly interface for updating and managing environment variables within a React application.

**StoreEnv Variables**

****

The StoreEnvVariables file is a React component that represents a form for storing environment variables related to SMTP configuration. Here's a summary of its key features:

State Management:

Utilizes React hooks (useState) to manage state variables such as loading, formData, and formErrors.

Context Usage:

Uses the React context API (useContext) to access the showToast function from a context provider.

Form Handling:

Updates the form state (formData) on user input through the handleChange function.

Implements form validation through the validateForm function to ensure essential fields like SMTP User, SMTP Password, and Sender Email are not empty.

Sends an HTTP POST request to the server to store environment variables on form submission.

Displays success or error messages based on the server response using the showToast function.

If validation fails, displays relevant error messages and prevents form submission.

UI Rendering:

Renders a form with input fields for SMTP User, SMTP Password, and Sender Email.

Utilizes React Icons (BsExclamationCircleFill) for displaying error icons.

Displays a loading spinner (Spinner component) during form submission.

Navigation:

Calls the showInfo(true) function to indicate a successful form submission.

Styling:

Applies responsive styling using Tailwind CSS classes for a clean and user-friendly interface.

Overall, this component provides a user-friendly interface for users to input and store SMTP-related environment variables within a React application.