**Billing Address**

Implemented the Billing Address feature to enhance the user experience during the course purchasing process. Utilizing the MERN (MongoDB, Express.js, React, Node.js) stack, I created a dedicated navigation section within the user profile for managing billing information. The form includes mandatory fields such as first name, last name, address line 1, contact number, city, state, country, and zip code. Users can conveniently save this information, reducing the time and effort required during future payments.

Upon successful storage of the billing address, users can view their information in the user profile's billing section. An "Edit" button allows users to modify their billing details seamlessly. The use of Tailwind CSS ensured a clean and responsive design, contributing to an intuitive user interface.

The backend, powered by Node.js and Express.js, handles the storage and retrieval of billing information securely. MongoDB serves as the database for persistent data storage. Custom toast messages provide real-time feedback on form submissions, ensuring a smooth user experience.

This Billing Address feature not only streamlines the payment process but also offers users the flexibility to manage their billing details effortlessly.

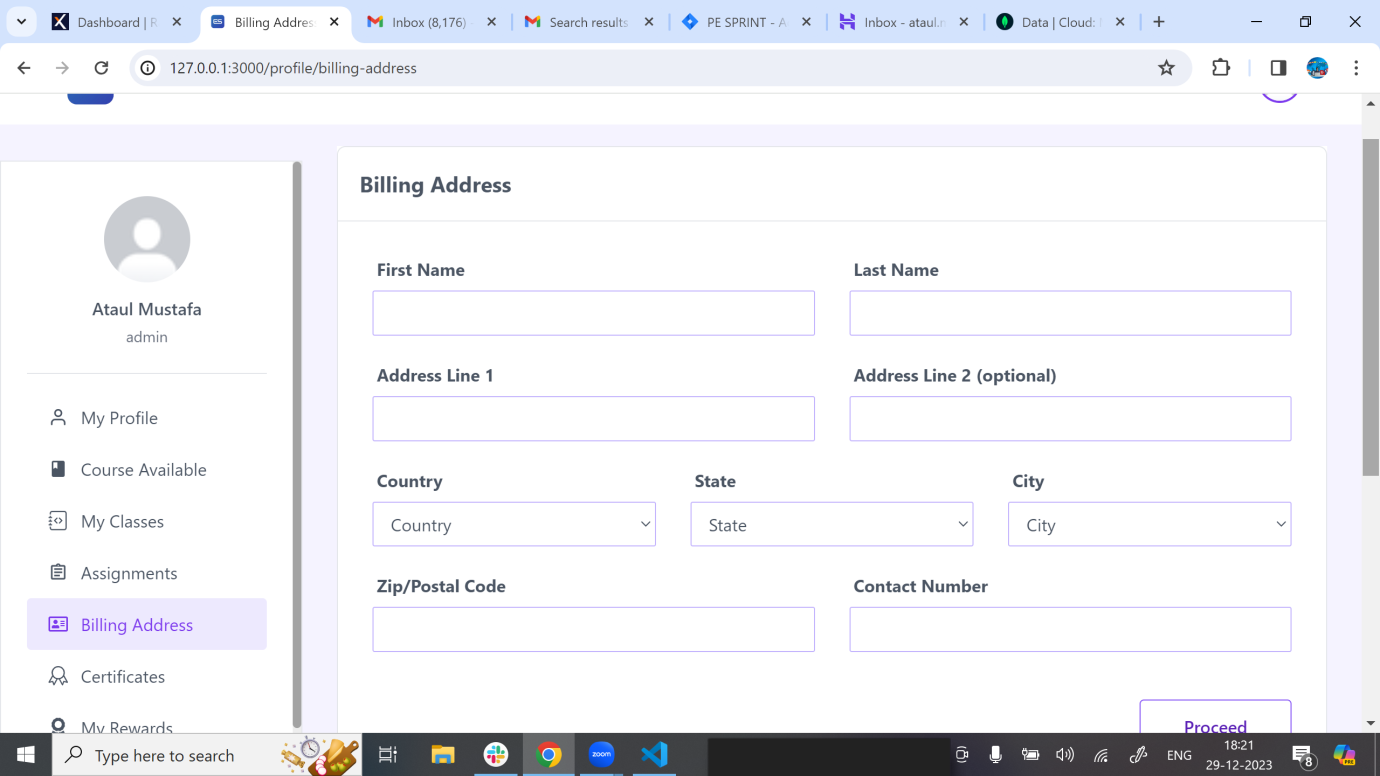
Next, feel free to ask about another feature or sub-feature, and I'll provide a detailed explanation for it.

**Store Billing Address**

The "Store Billing Address" feature facilitates a streamlined course payment process by allowing users to securely save their billing details. Implemented using the MERN stack, the frontend, developed with React and styled with Tailwind CSS, presents users with a form containing mandatory fields such as first name, last name, contact number, and address details.

Upon submission of the form, the backend, powered by Node.js and Express.js, ensures secure data handling and stores the billing information in a MongoDB database. Real-time validation checks guarantee data accuracy, and custom toast notifications provide immediate feedback on successful storage or any encountered issues.

For user convenience, the stored billing address is displayed in the user profile's billing section. An "Edit" button enables users to make modifications if needed. This feature not only enhances the user experience during payment but also contributes to the overall user-friendly design of the platform.

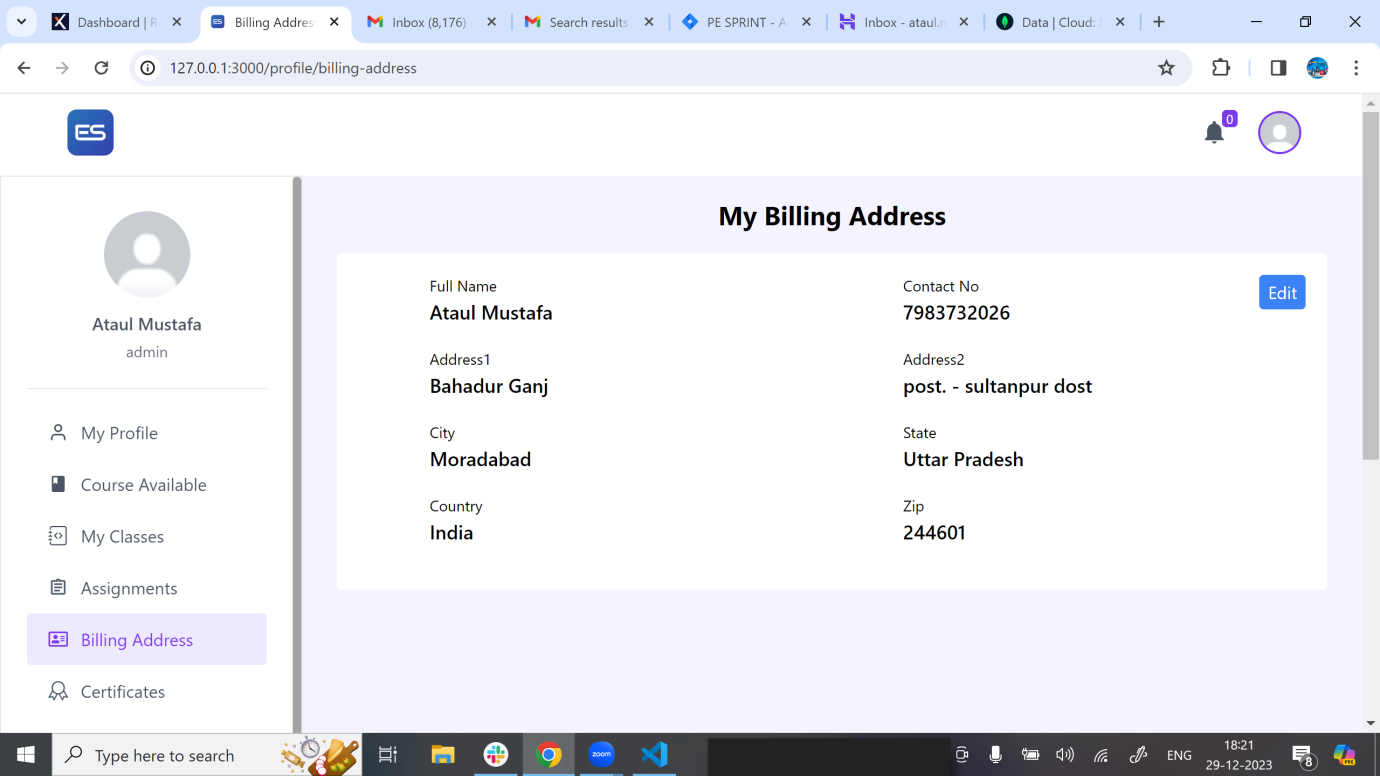


**Fetch Billing Address**

The "Fetch/Get Billing Address" functionality ensures a seamless user experience by allowing users to retrieve their stored billing information. Implemented within the user profile section, this feature utilizes React on the frontend, with Tailwind CSS providing a clean and responsive design.

When users navigate to the billing section, the frontend communicates with the backend, built on Node.js and Express.js, which in turn fetches the billing details from the MongoDB database. The retrieved information is then dynamically displayed in the user interface, offering users visibility into their stored billing address.

Real-time interaction between the frontend and backend ensures that users consistently see the most up-to-date billing information. This feature not only adds convenience for users but also maintains data accuracy, contributing to the overall efficiency of the platform. The use of the MERN stack ensures a robust and responsive implementation of the "Fetch/Get Billing Address" feature.

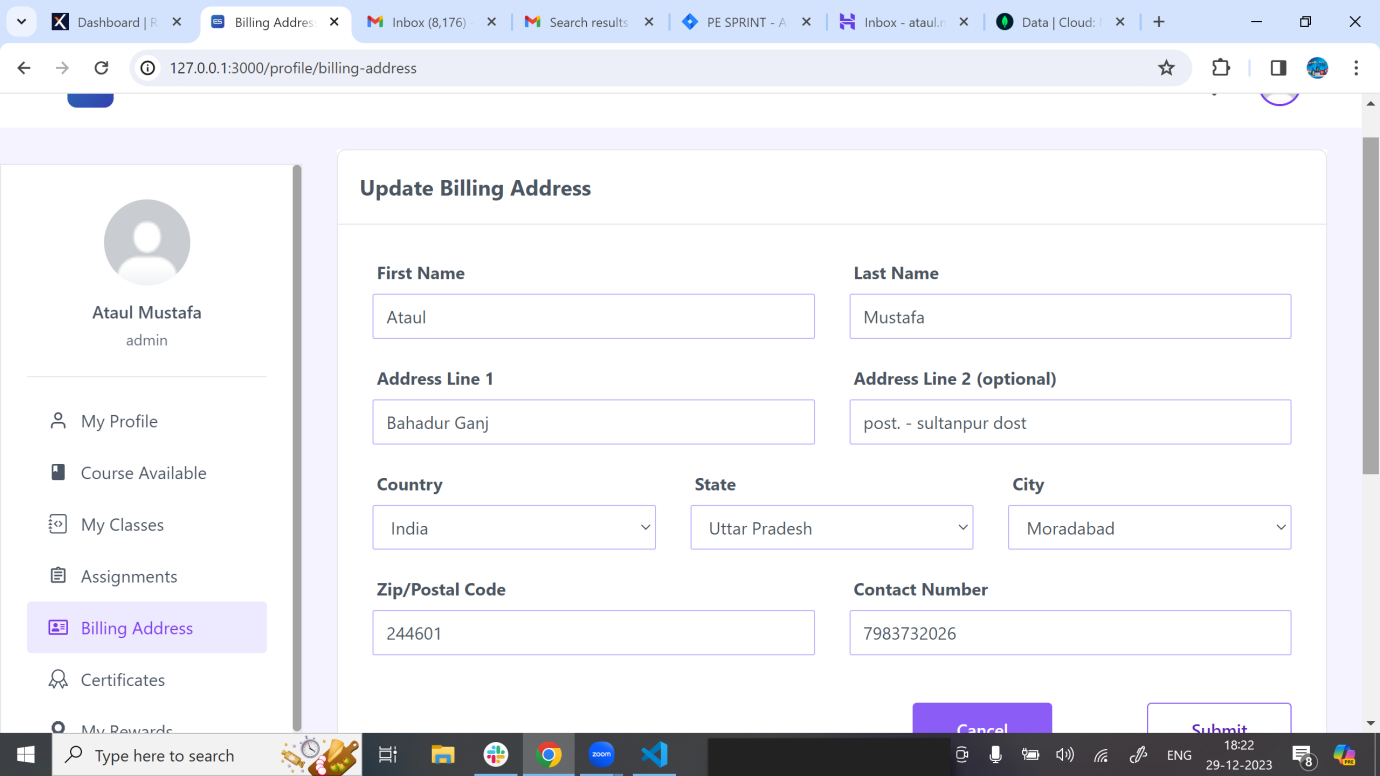


**Update Billing Address**

The "Update Billing Address" functionality was designed to allow users a hassle-free way to update their billing information. Implemented using React and Tailwind CSS for the frontend, users can initiate edits through a dedicated button in the user profile. The backend, powered by Node.js and Express.js, manages secure data handling and interacts with the MongoDB database for persistent storage.

Upon clicking "Edit," users encounter a pre-filled form, maintaining consistency with the initial billing address creation. All fields are editable, except for the primary address line, ensuring data accuracy. Real-time validation checks and custom toast notifications enhance the user experience by providing instant feedback on any modifications.

This feature not only empowers users to keep their billing information up-to-date but also contributes to a responsive and user-friendly platform. The technology stack, including MERN components, plays a crucial role in delivering a seamless editing experience.



**Password Management**

The "Password Management" feature focuses on ensuring the secure handling of user passwords within the system. Implemented with the MERN stack, the frontend, developed with React and styled using Tailwind CSS, provides a user-friendly interface.

Users can change their passwords through a dedicated form within their profile. The form includes fields for the old password, new password, and confirmation of the new password. Real-time validation checks enhance the user experience by providing immediate feedback on the accuracy of the entered information.

On the backend, powered by Node.js and Express.js, secure protocols manage password changes, ensuring data integrity and confidentiality. MongoDB is utilized for the persistent storage of password-related data.

Custom toast notifications provide real-time feedback on the success of password changes, contributing to a smooth user experience. The cohesive integration of the MERN stack ensures a robust and responsive implementation of the "Password Management" feature, emphasizing both security and user convenience.

**Forgot Password**

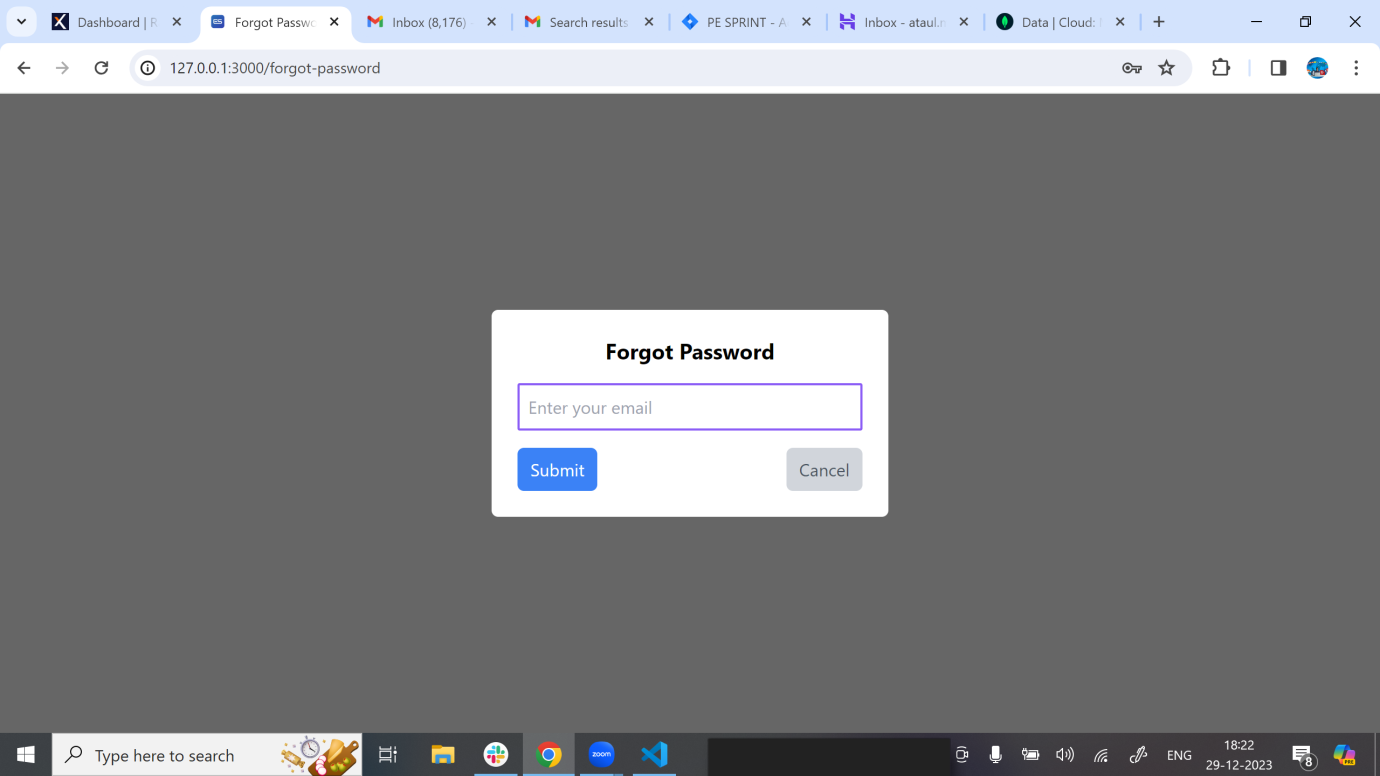
The "Forgot Password" feature provides a secure and user-friendly way for users to recover their passwords in case of forgetfulness.

**Initiating the Process:** Users click on the "Forgot Password" button available on the signup form when they need to recover their password.

**Email Verification:** Users are prompted to enter their registered email address.

Real-time validation ensures password strength and accuracy.

The frontend, developed with React and styled with Tailwind CSS, offers a seamless and responsive interface. On the backend, powered by Node.js and Express.js, secure protocols handle the email verification and password reset processes.



**Reset Password**

The "Reset Password" feature allows users to securely reset their passwords, providing a straightforward and user-friendly process.

**Recovery Email:** Users receive a recovery email containing a unique link after initiating the password recovery process.

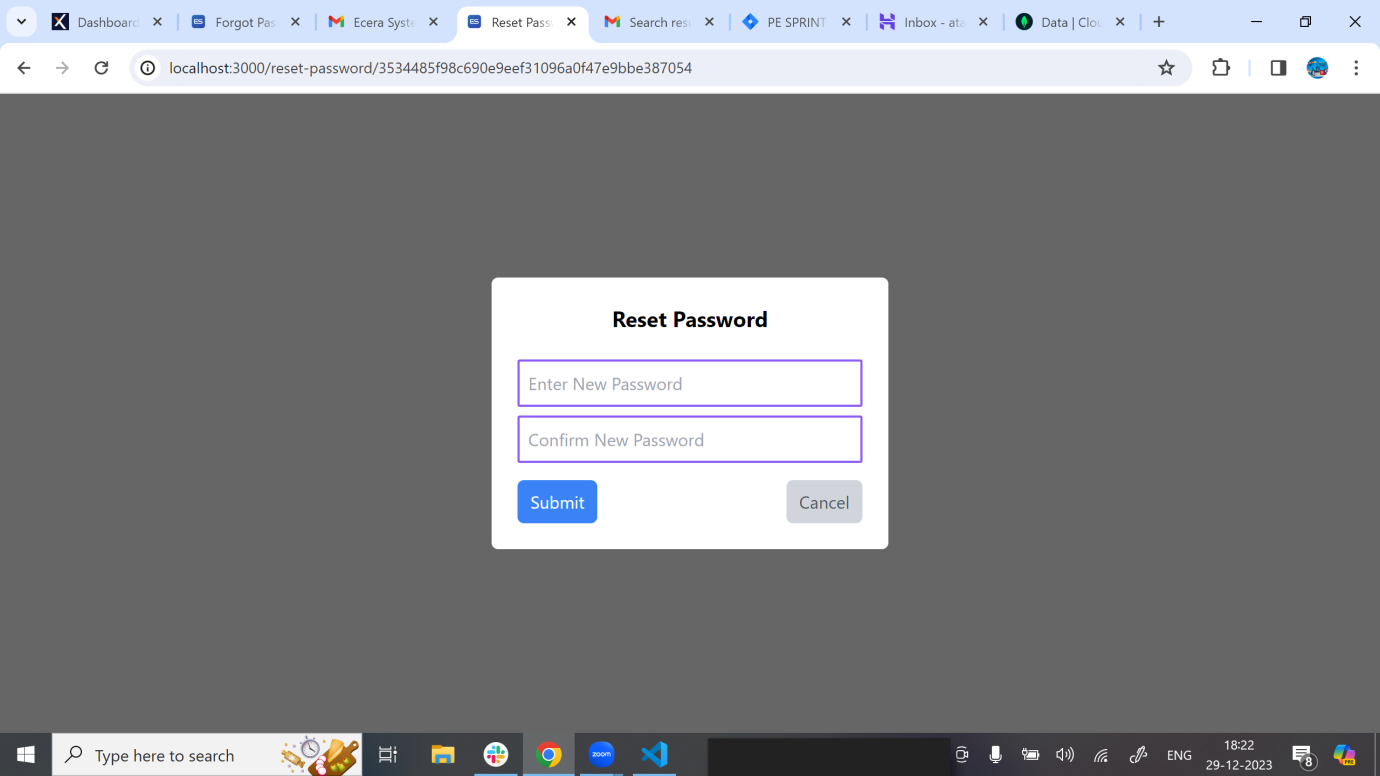
**Secure Link Activation:** Clicking the unique link redirects users to a secure form for resetting their password.

**New Password Entry:** Users enter and confirm their new password on the secure form.

Real-time validation checks ensure password strength and accuracy.

The frontend, developed with React and styled using Tailwind CSS, offers an intuitive and responsive interface. Secure communication between the frontend and the backend, powered by Node.js and Express.js, ensures the integrity of the password reset process.

Custom toast notifications provide immediate feedback on the success of the password reset, guiding users through the process.



**Change Password**

The "Change Password" feature empowers users to update their passwords conveniently within their profile.

**Accessing Change Password Form:** Users navigate to the change password section within their user profile.

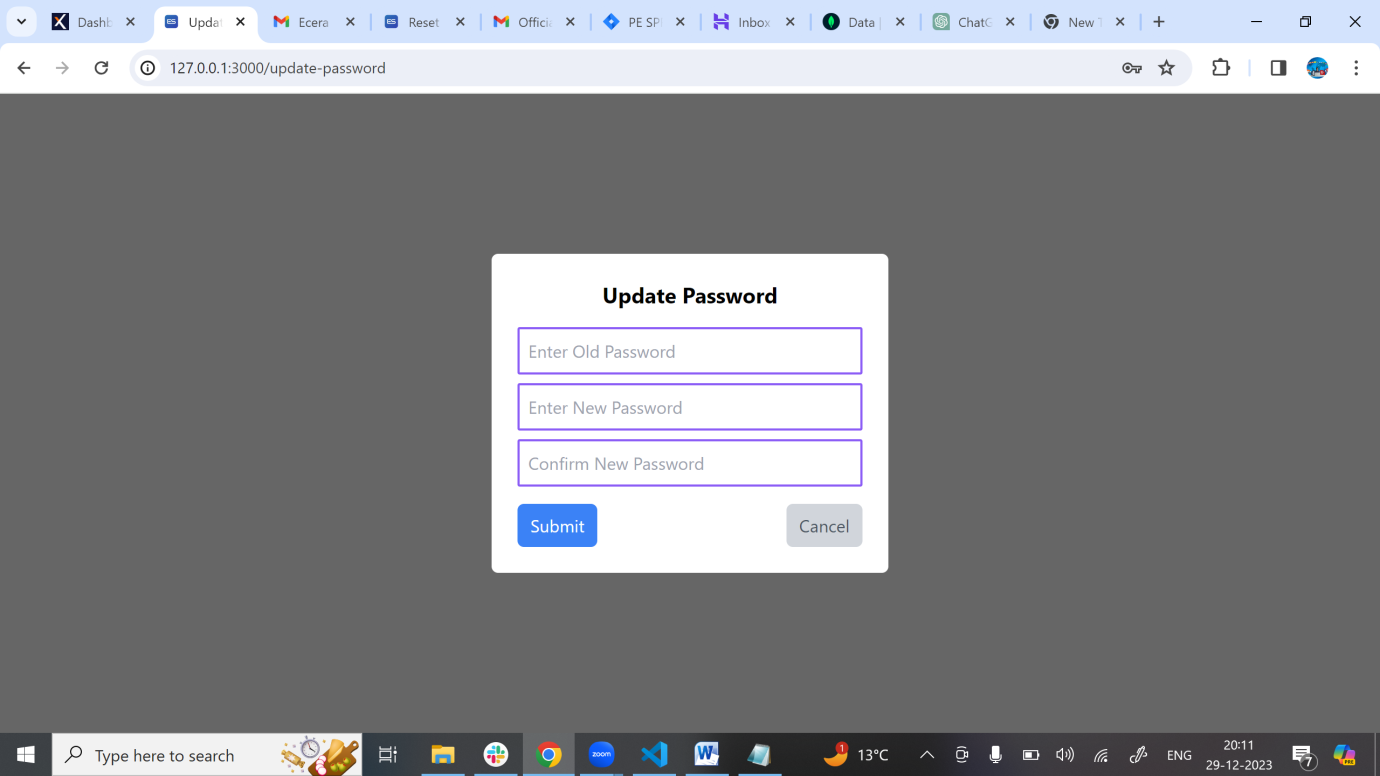
**Entering Details:** Users input their old password, new password, and confirm the new password in the dedicated form.

Real-time validation checks ensure password strength and accuracy.

**Submitting Changes:** Upon submitting the form, the frontend communicates securely with the backend, powered by Node.js and Express.js.

**Password Update:** The backend securely updates the user's password in the MongoDB database.

The frontend, developed using React and styled with Tailwind CSS, provides a responsive and intuitive interface for users. Secure communication between the frontend and backend ensures the confidentiality of the password change process.



**Change Password with Admin**

The "Change Password" feature on the admin side enables administrators to update the passwords of any user within the system.

**Accessing Admin Dashboard:** Admins navigate to the user management section within the admin dashboard.

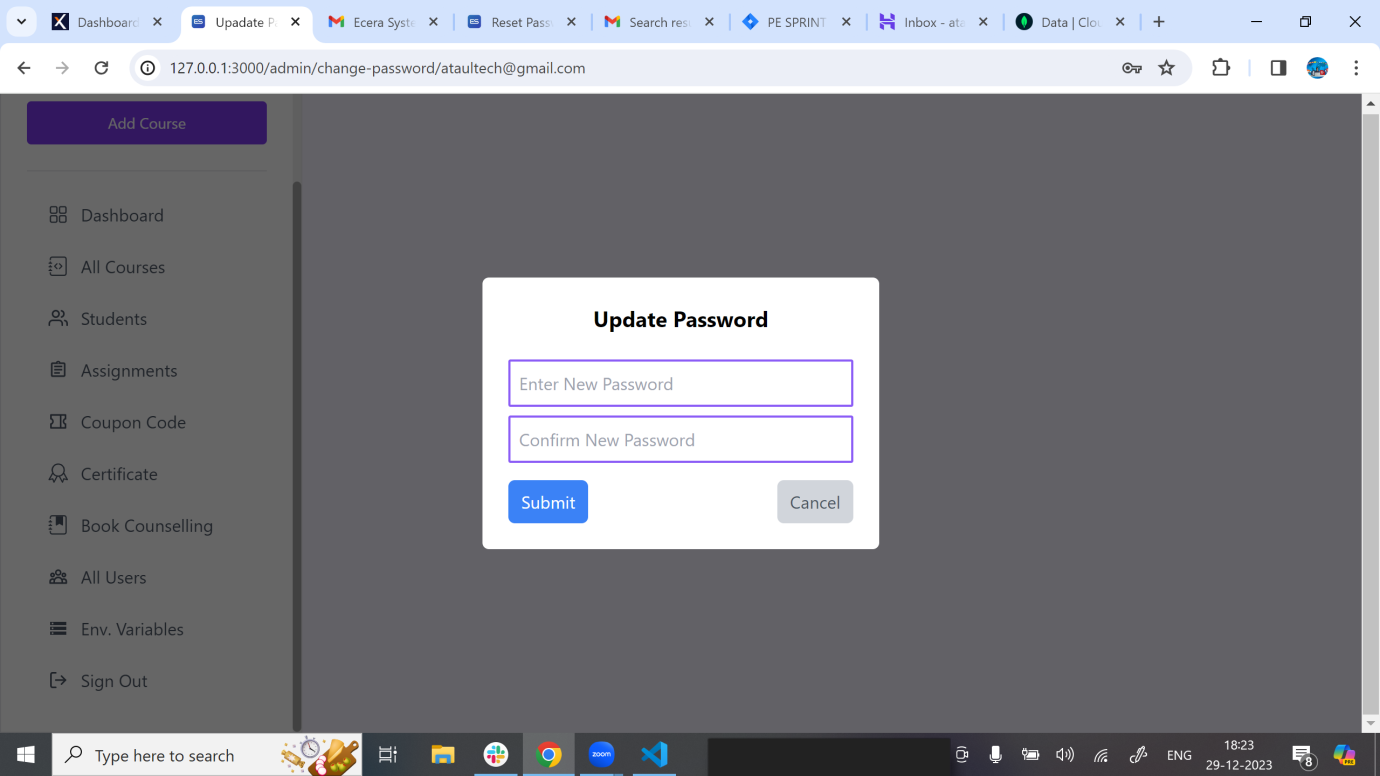
**Selecting User:**  Admins choose the specific user for whom they wish to change the password.

**Entering New Password:** Admins input a new password for the selected user.

**Submitting Changes:** The admin dashboard securely communicates with the backend, powered by Node.js and Express.js.

**Password Update:** The backend securely updates the selected user's password in the MongoDB database.

The frontend of the admin dashboard, developed using React and styled with Tailwind CSS, provides a responsive and efficient interface for administrators. Secure communication between the frontend and backend ensures the confidentiality of the password change process.



**Environment Variables Management**

The "Environment Variables Management" feature provides an efficient way for administrators to manage application settings without directly editing the .env file. Implemented with the MERN stack, this feature allows admins to add, retrieve, and edit environment variables directly from the dashboard.

On the frontend, built with React and styled using Tailwind CSS, administrators can easily navigate to the environment variables section. The form for adding or editing variables includes fields for key-value pairs, ensuring flexibility in configuration. Real-time validation ensures data accuracy during input.

The backend, powered by Node.js and Express.js, manages the secure storage and retrieval of environment variables in the MongoDB database. The dynamic nature of this feature allows administrators to adapt application settings without the need for manual file editing.

Custom toast notifications provide real-time feedback on successful variable addition or edits. This feature not only simplifies configuration management but also enhances the security and accessibility of environment variables within the application. The use of the MERN stack ensures a cohesive and responsive implementation of the "Environment Variables" feature.

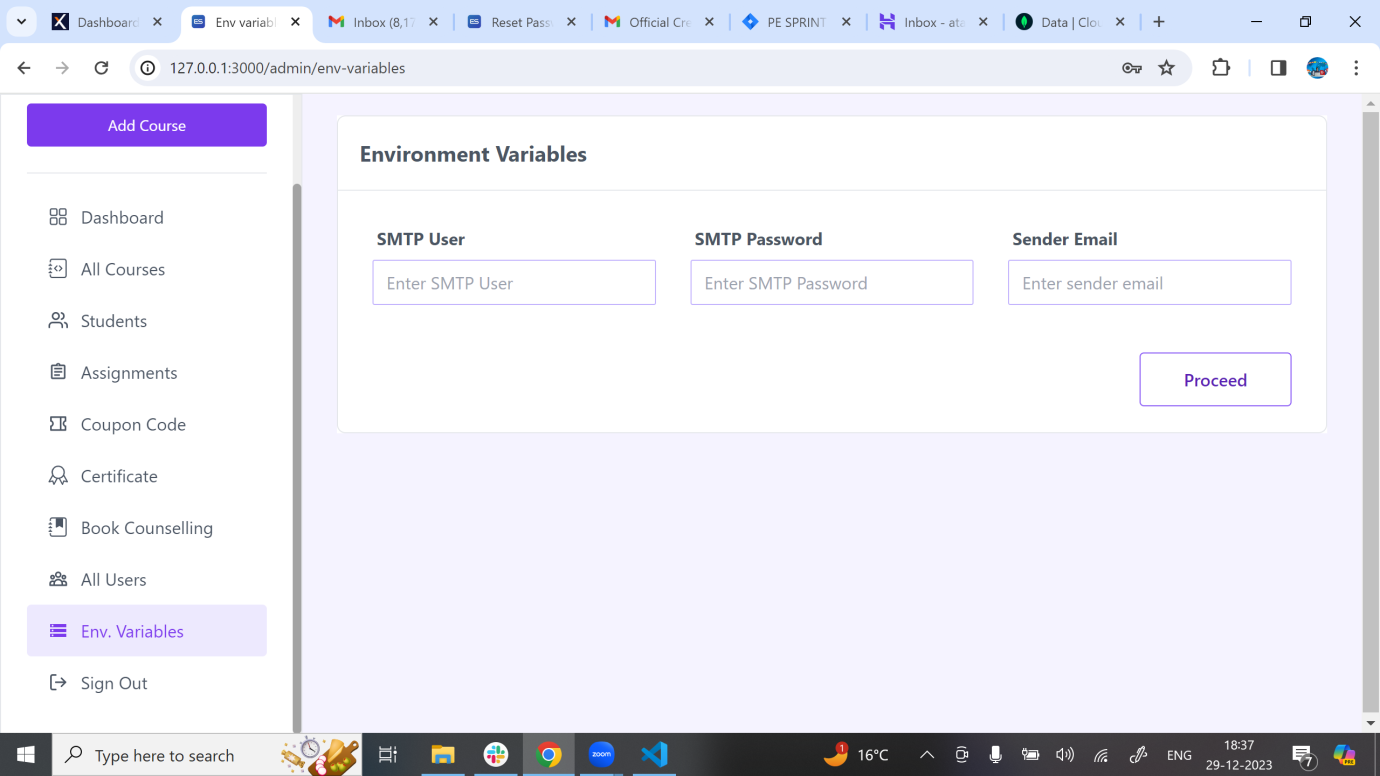
**Store Environment Variable**

The "Store Environment Variables" feature empowers administrators to efficiently manage application settings directly from the dashboard. Leveraging the MERN stack, this feature is implemented with React and Tailwind CSS on the frontend for a user-friendly interface.

Administrators access the environment variables section where they can add or edit variables. The form includes key-value pairs, ensuring flexibility in configuring application settings. Real-time validation checks on the frontend guarantee the accuracy of the entered data before submission.

On the backend, powered by Node.js and Express.js, the application securely stores environment variables in a MongoDB database. This dynamic interaction ensures that administrators can adapt application settings without the need for manual intervention in configuration files.

Custom toast notifications provide immediate feedback on the success of variable storage, contributing to a smooth user experience. This feature not only simplifies the process of managing environment variables but also enhances the overall security and accessibility of critical application settings. The MERN stack's integrated technologies ensure a robust and responsive implementation of the "Store Environment Variables" feature.

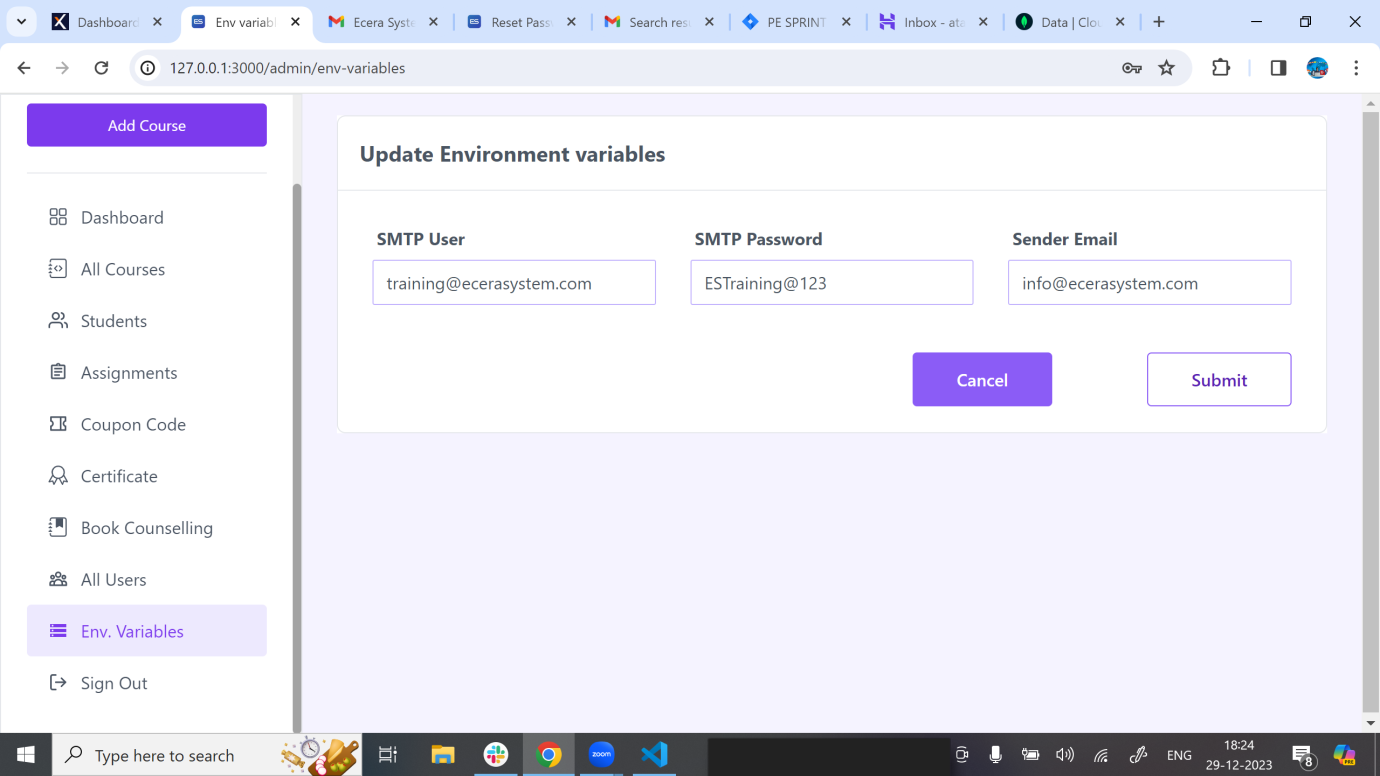


**Update Environment Variables**

The "Edit Environment Variables" feature allows administrators to modify application settings directly from the dashboard, streamlining the configuration management process. Leveraging the MERN stack, the frontend is built using React and styled with Tailwind CSS for a user-friendly interface.

Within the environment variables section, administrators can click on the "Edit" button available on the right upper corner. This action triggers the display of an editable form pre-filled with the existing key-value pair. Real-time validation checks ensure the accuracy of the modified data before submission.

On the backend, powered by Node.js and Express.js, the application securely updates the MongoDB database with the edited environment variable. The dynamic nature of this feature allows administrators to adapt settings without manual file editing, contributing to a more efficient configuration management process.

****

**Fetch/get Environment Variables**

The "Fetch Environment Variables" feature facilitates the retrieval of application settings directly from the dashboard, offering administrators a seamless and user-friendly experience. Implemented with the MERN stack, the frontend is developed using React with Tailwind CSS for a clean and responsive design.

Administrators can easily navigate to the environment variables section where the frontend communicates with the backend, built on Node.js and Express.js. The backend retrieves environment variables securely stored in the MongoDB database and dynamically displays them in the user interface.

Real-time interaction between the frontend and backend ensures that administrators consistently access the most up-to-date application settings. This dynamic feature not only simplifies the process of retrieving environment variables but also enhances the overall efficiency of configuration management.

