Project Documentation

Framework Used

Django REST Framework

The backend API for this project is built using Django REST Framework, a powerful and flexible toolkit for building Web APIs in Django.

Vanilla JavaScript

The frontend interactions are implemented using vanilla JavaScript, which directly manipulates the DOM and handles asynchronous operations via Fetch API.

Project Documentation

Database Used

PostgreSQL

The project uses PostgreSQL as its database management system. PostgreSQL is a powerful, open-source object-relational database system known for its reliability, feature robustness, and performance.

Project Documentation

Core Functionalities

Data Fetching

- API Endpoint: The JavaScript code fetches user data from the API endpoint http://127.0.0.1:8000/api/list-users/.

- Data Handling: The fetched user data is logged to the console for verification and further processing.

Dynamic Table Rendering

- User Data Display: The application dynamically renders user data in a table format on the webpage. Each row displays the following user details:

  - Full Name

  - Organization Name

  - Email

  - Organization ID

- Action Buttons: Each row includes 'Edit' and 'Delete' buttons to allow users to modify or remove their data.

User Profile Management

- Form Handling: The application provides forms for users to enter their details, including:

  - Full Name

  - Organization Name

  - Organization ID

  - Email

  - Password (and Password Confirmation for account creation)

- Data Pre-fill for Editing: When editing user details, the form fields are pre-filled with the existing data for ease of modification.

- Form Submission: The application handles form submission to update user details in the backend.

User Data Editing

- Edit Functionality: Users can switch to edit mode to update their profile information.

- Dynamic UI Update: The interface updates dynamically to show the edit form with pre-filled data when the 'Edit' button is clicked.

- Data Update: Upon form submission, the updated user details are sent to the API to be saved.

User Data Deletion

- Delete Functionality: Users can delete their profile information using the 'Delete' button.

- API Interaction: The delete function sends a DELETE request to the API endpoint to remove the user's data from the system.

State Management

- Interface States: The application manages different states, such as:

  - Login view

  - Edit view

  - Report view

- Dynamic Content Display: Based on the current state, the interface updates to show or hide relevant sections to provide a smooth user experience.

Project Documentation

Conclusion

This project leverages Django REST Framework and vanilla JavaScript to create a dynamic and interactive user management system. The backend API handles data fetching, updating, and deleting user information, while the frontend dynamically renders user data, manages form inputs, and provides a responsive interface for editing and deleting user details. PostgreSQL ensures robust and efficient data management. The clear separation of concerns between the backend and frontend ensures a robust and maintainable codebase, facilitating future enhancements and scalability.