

XYZ Company

~ Employee Records

Contents:

1. Introduction
2. Technical Stack
3. Build and Deployment
4. Authentication
5. Validation
6. Images(Screenshots)

Developer : M Venu Gopal

Mail ID : venumalladi18@gmail.com

Github Repo : [Django-React-CRUD](#)

Introduction:

Django-React-CRUD App is used to perform CRUD Operations (Create, Read, Update, Delete) on the Employee Records of XYZ Company.

Requirements :

- List the Employee Records of XYZ Company
 - User should be able to Create (Add new employee), Read (Read the list of employees), Update (Update the details of existing employee) and Delete (Delete an employee from the list)
 - Only Authenticated Users should be able to perform Create, Update and Delete Operations (they must be protected) whereas any user should be able to perform Read Operation and get the list of employees
-

Technical Stack:

1. To keep all the dependencies required by this Project isolated, Python Virtual Environment has been used.
2. Backend API has been built using Python and Django framework
3. Front-end has been built using ReactJS
4. CSS, Bootstrap and Reactstrap have been used to style the front-end application
5. User will be able to perform CRUD operations only after successful login
6. As of now, Signing Up a new User feature shall be added based on requirements.

Build and Deployment:

Setup :

1. Download the zip file and extract the code from the above mentioned Github Repository.
2. Make sure python3 and virtualenv are installed on your system.
3. Else to install virtualenv run the command - **pip3 install virtualenv**
4. After successful installation run - **source xyz_env/bin/activate**
5. Once the virtual environment is activated, the name (**xyz_env**) of your virtual environment will appear on the left side of your terminal. This will let you know that the virtual environment is currently active. (As shown in Fig-1 below)
6. Now run the command : **pip3 install -r requirements.txt**
7. This will download all python packages required for the project

Backend :

8. Now, change directory, **cd django_react_proj**
9. To start and run the Backend server, use the command:
python manage.py runserver (In **django_react_proj** folder)
(As shown in Fig-2 below)
10. Backend will run on, localhost:8000/
11. Open localhost:8000/api/employees in browser. This allows you to check your current status of db.
(As shown in Fig-3 below)

Frontend :

12. To start running front-end make sure NodeJS is installed on your system. Now, open Node.js command prompt.
13. Change your current directory from root (Root user) to **front-end folder** and run the command : **npm install**
14. This will install all node-packages required based on package.json file in the **front-end folder**
15. After successful installation, run command: **npm start**

This will run our front-end application on localhost:3000/
(As shown in Fig-4 below)

16. A login form will be loaded and on successful login :

Username : venu

Password : user1234

On incorrect credentials, error will be shown.

You will be redirected to localhost:3000/employees-list/ which will let you perform all required CRUD Operations on the backend.

(As shown in Fig-5a&5b&6 below)

Authentication:

For now, Sign Up feature hasn't been added and form validation is done using brute force, but this can be further developed into database of User accounts in the backend API

Validations:

Adding a new Employee to DataBase:

- EMP ID : Employee ID will be Unique ID assigned by the company to that particular Employee (Cannot be Empty)
- EMP Name : Name of the Employee (Cannot be Empty)
- EMAIL : Mail ID of the Employee (Cannot be Empty and should be valid mail id (Validated based on Input type))
- Phone : Contact number of the Employee (Cannot be Empty and Input should be a valid phone number(Validated based on RegEx - Regular Expression)))
- As shown in Fig-7a & 7b & 7c & 7d & 8 below

Editing an existing Employee from the DataBase:

- Initially, existing data from the database will be loaded on the form, but can be edited as per requirements
- EMP ID : Employee ID will be Unique ID assigned by the company to that particular Employee (Cannot be Empty)
- EMP Name : Name of the Employee (Cannot be Empty)
- EMAIL : Mail ID of the Employee (Cannot be Empty and should be valid mail id (Validated based on Input type))
- Phone : Contact number of the Employee (Cannot be Empty and Input should be a valid phone number(Validated based on RegEx - Regular Expression)))
- As shown in Fig-9&10 below

Delete an Employee from the DataBase:

- On clicking Delete button, alert will pop up asking for deletion confirmation
- On further confirmation, employee will be deleted from the database
- On selecting Cancel, no changes will be made to the database
- As shown in Fig-11&12 below

Screenshots:

Fig-1 :

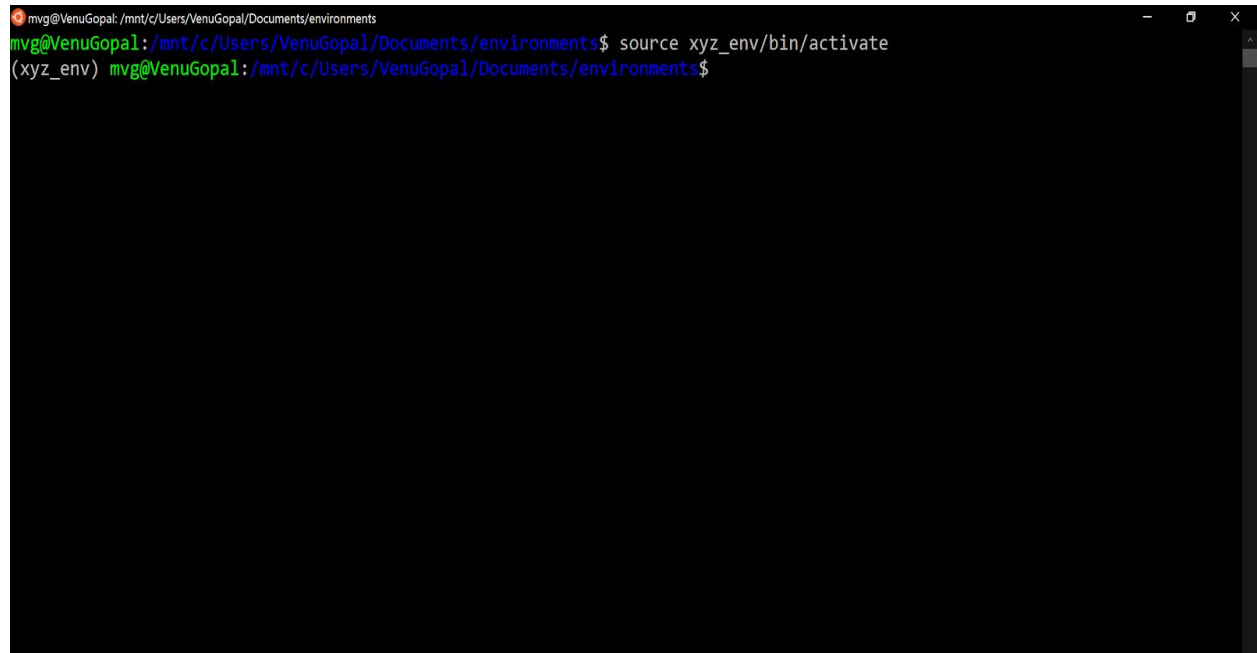
A terminal window with a dark background. The title bar shows the user 'mvg' and the path '/mnt/c/Users/VenuGopal/Documents/environments'. The prompt is 'mvg@VenuGopal:/mnt/c/Users/VenuGopal/Documents/environments\$'. The user enters 'source xyz_env/bin/activate'. The prompt changes to '(xyz_env) mvg@VenuGopal:/mnt/c/Users/VenuGopal/Documents/environments\$'.

Fig-2 :

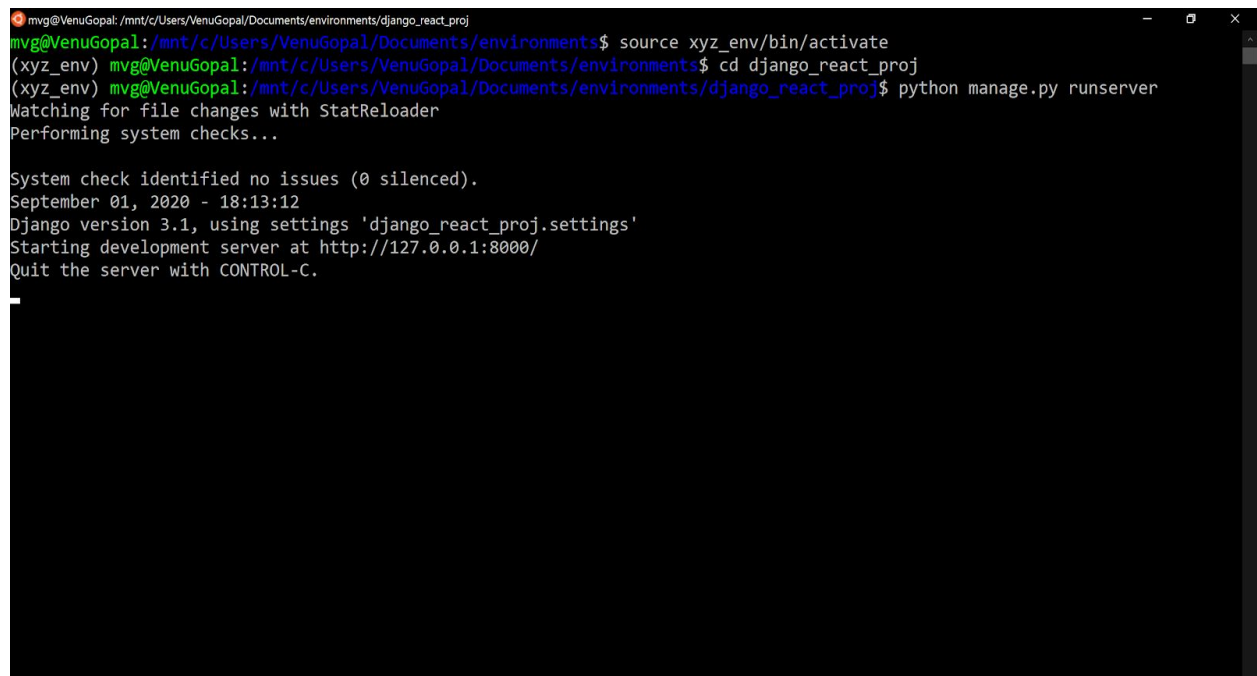
A terminal window with a dark background. The title bar shows the user 'mvg' and the path '/mnt/c/Users/VenuGopal/Documents/environments/django_react_proj'. The prompt is 'mvg@VenuGopal:/mnt/c/Users/VenuGopal/Documents/environments\$'. The user enters 'source xyz_env/bin/activate'. The prompt changes to '(xyz_env) mvg@VenuGopal:/mnt/c/Users/VenuGopal/Documents/environments\$'. The user enters 'cd django_react_proj'. The prompt changes to '(xyz_env) mvg@VenuGopal:/mnt/c/Users/VenuGopal/Documents/environments/django_react_proj\$'. The user enters 'python manage.py runserver'. The output shows 'Watching for file changes with StatReloader', 'Performing system checks...', 'System check identified no issues (0 silenced).', 'September 01, 2020 - 18:13:12', 'Django version 3.1, using settings 'django_react_proj.settings'', 'Starting development server at http://127.0.0.1:8000/', and 'Quit the server with CONTROL-C.'.

Fig-3 :

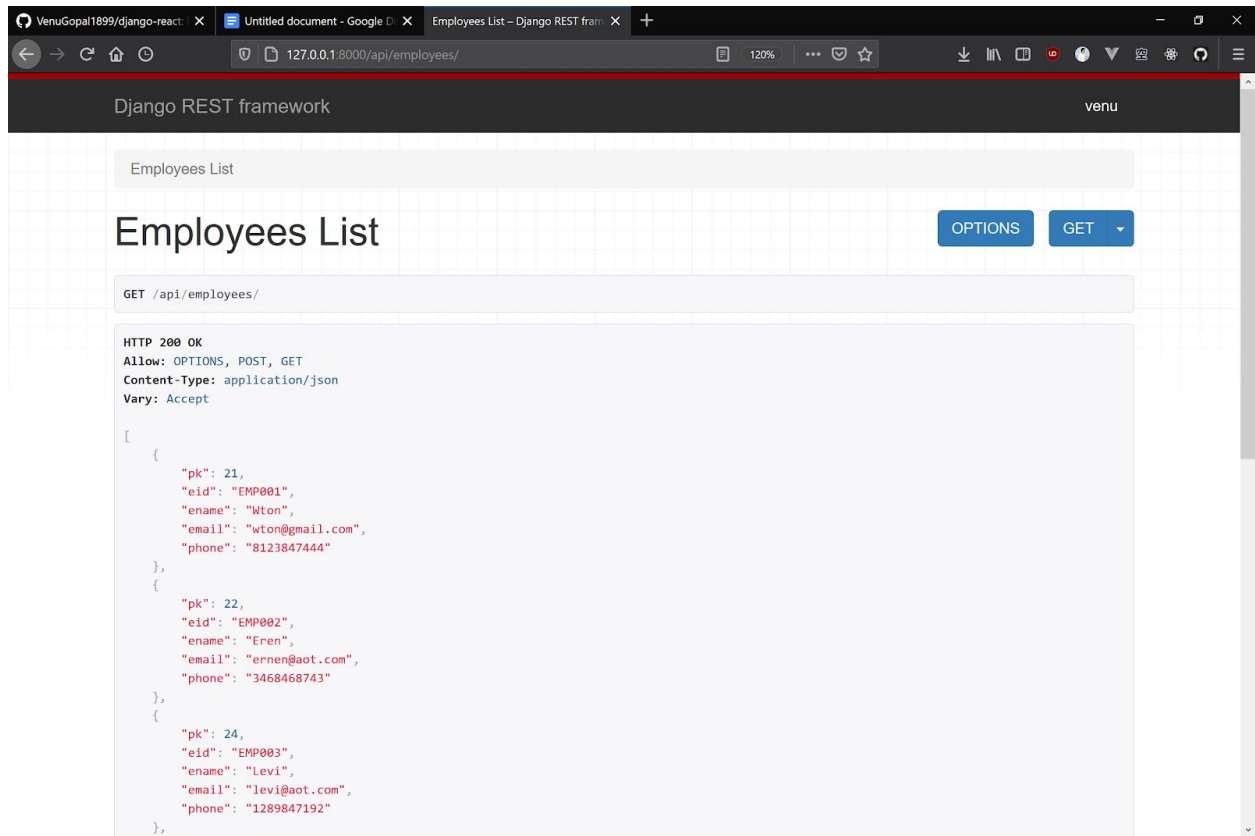


Fig-4 :

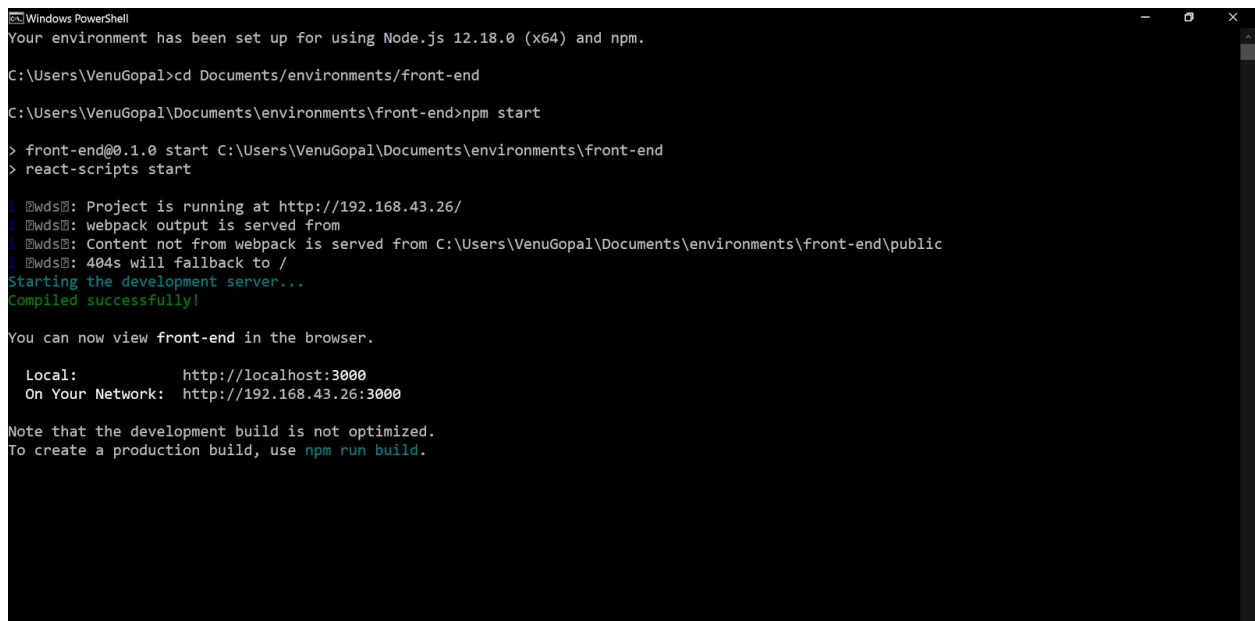
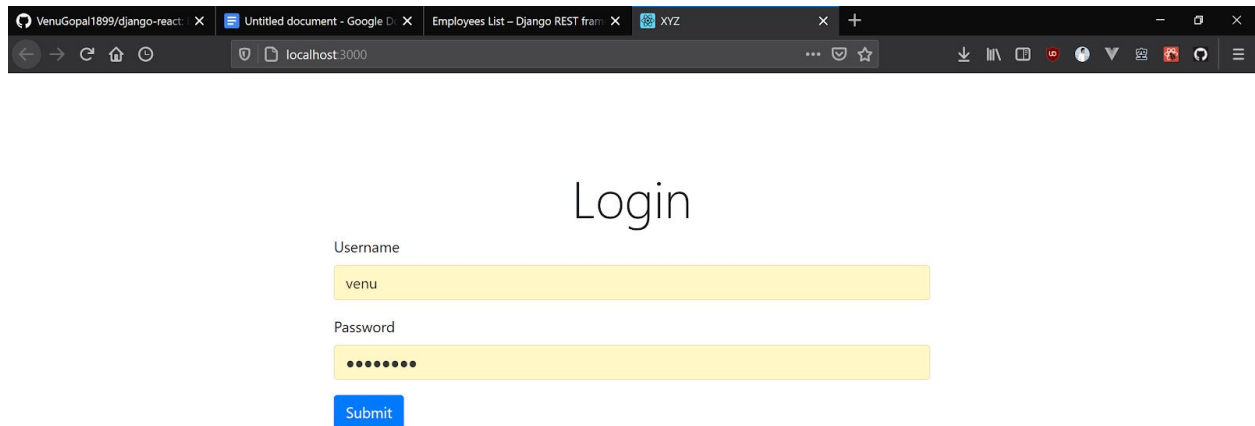


Fig-5a :



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page has a dark header with several tabs: 'VenuGopal1899/django-react', 'Untitled document - Google', 'Employees List - Django REST fram', and 'XYZ'. The main content area is white and features the word 'Login' in a large, black, sans-serif font. Below the title, there are two input fields: 'Username' with the text 'venu' and 'Password' with masked characters '.....'. A blue 'Submit' button is positioned below the password field.

Username

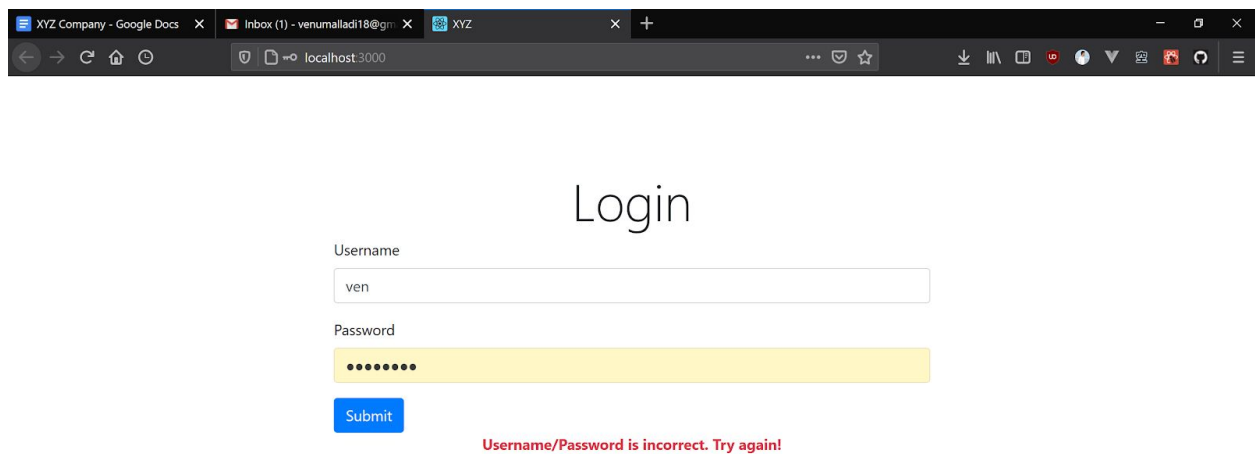
venu

Password

.....

Submit

Fig-5b :



The screenshot shows the same web browser window as Fig-5a, but with a different state. The 'Username' field now contains 'ven' and the 'Password' field contains masked characters '.....'. The blue 'Submit' button is still present. Below the button, a red error message is displayed: 'Username/Password is incorrect. Try again!'. The browser tabs and address bar remain the same.

Username

ven

Password

.....

Submit

Username/Password is incorrect. Try again!

Fig-6 :

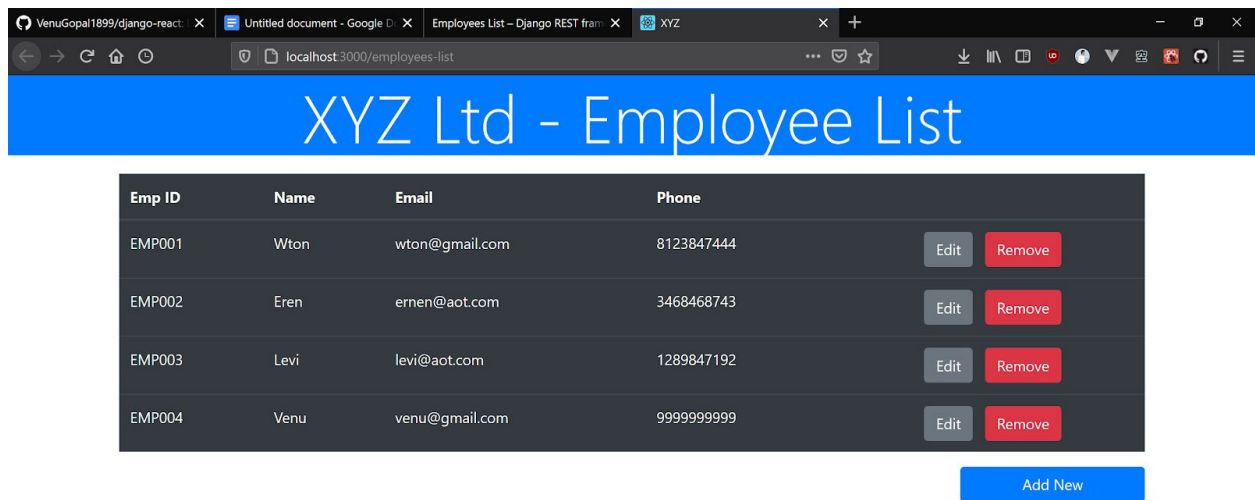


Fig-7a :

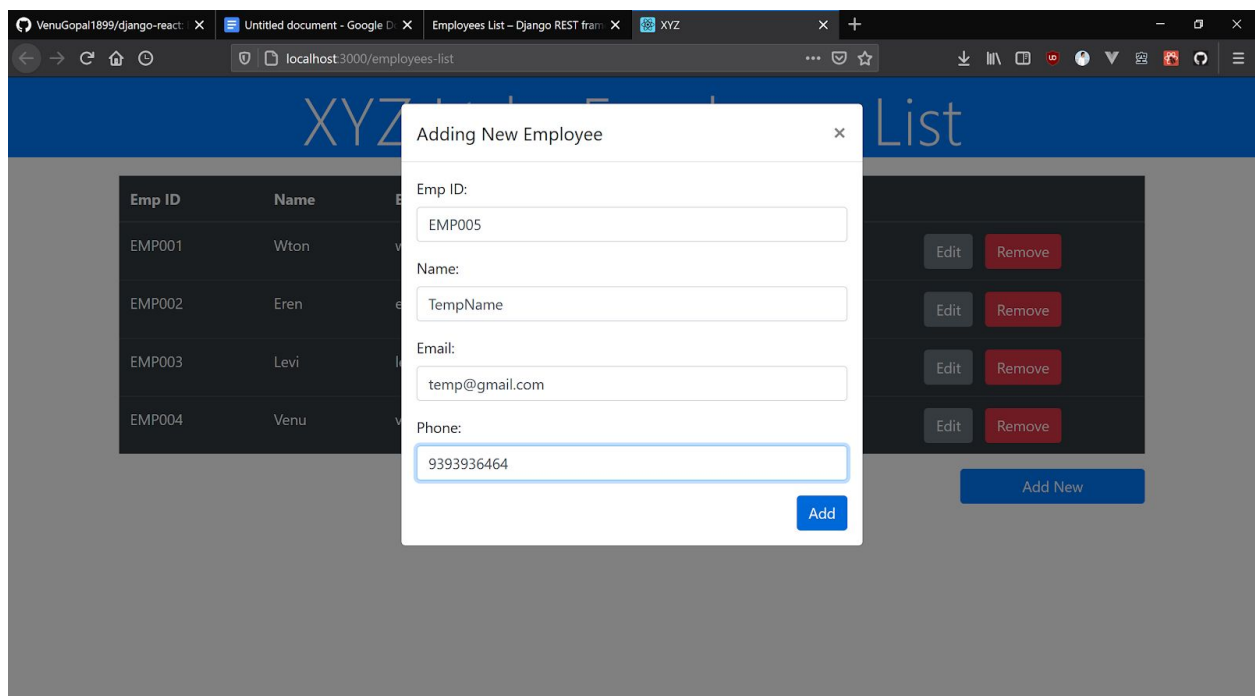


Fig-7b : (EMPTY FORM SUBMISSION)

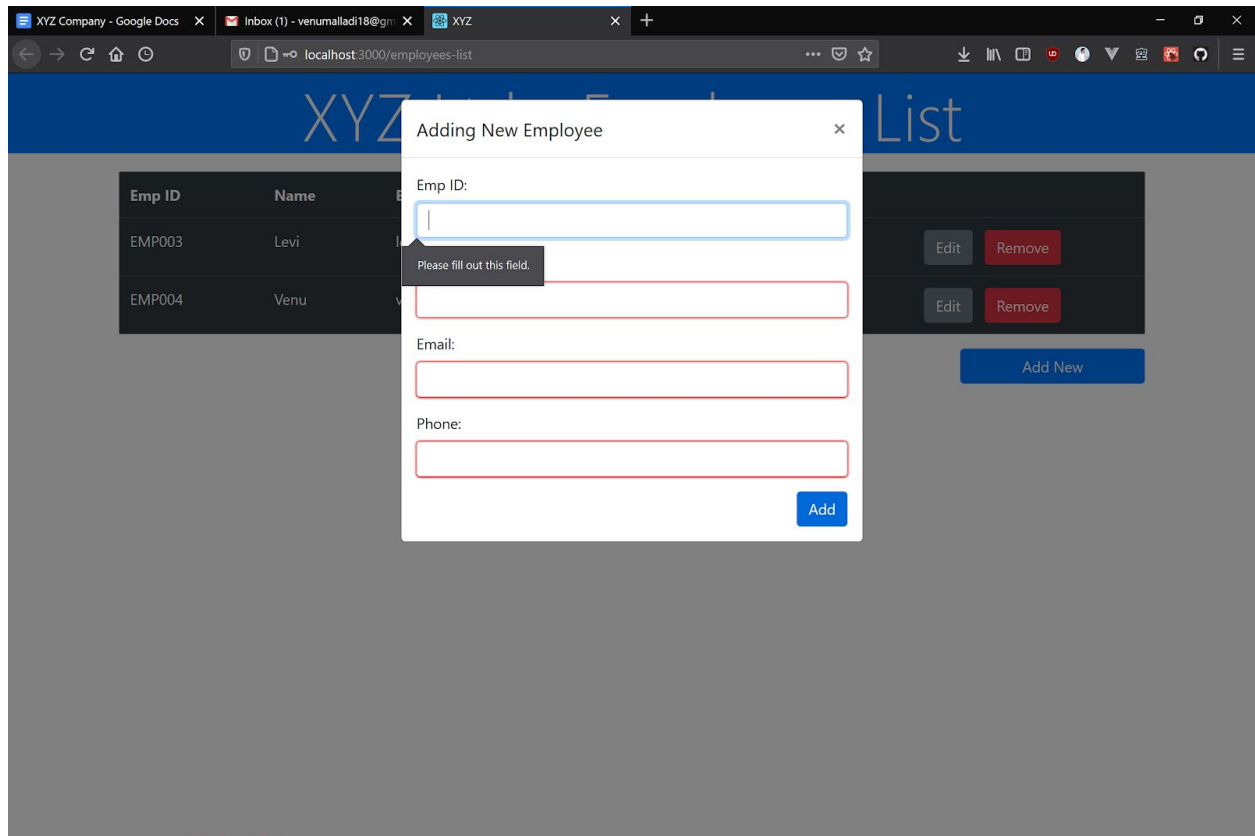


Fig-7c : (INCORRECT MAIL ID FORMAT)

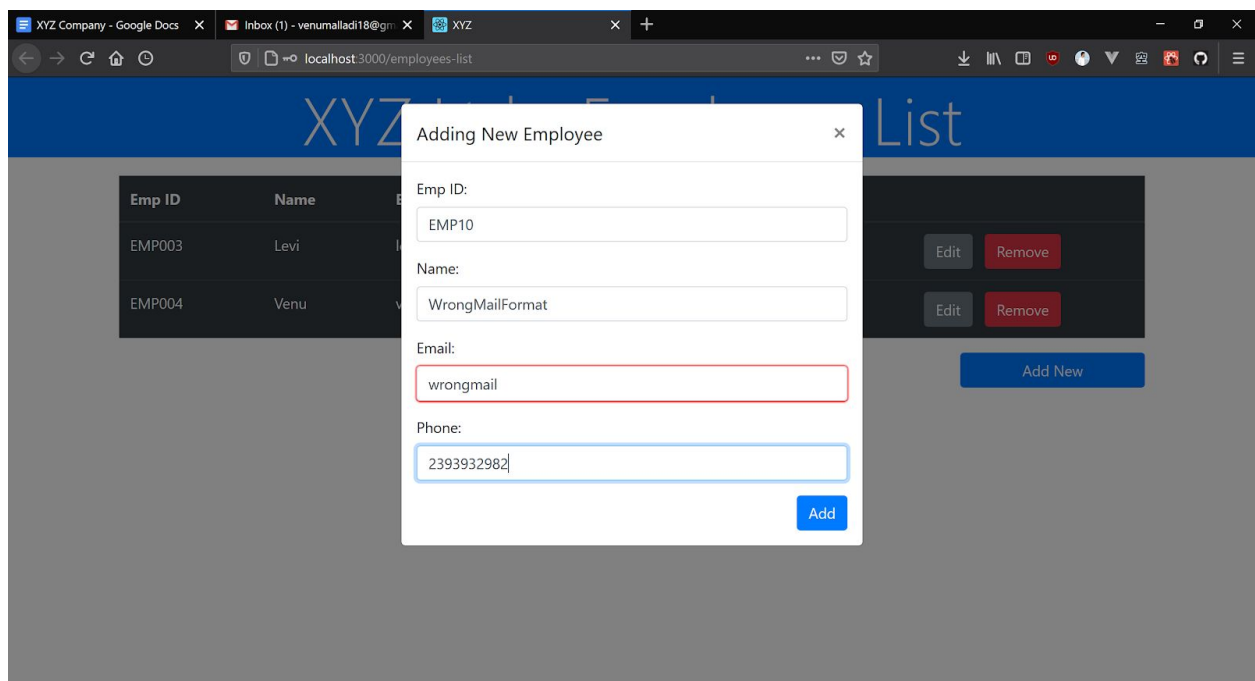


Fig-7d : (INCORRECT PHONE NUMBER FORMAT)

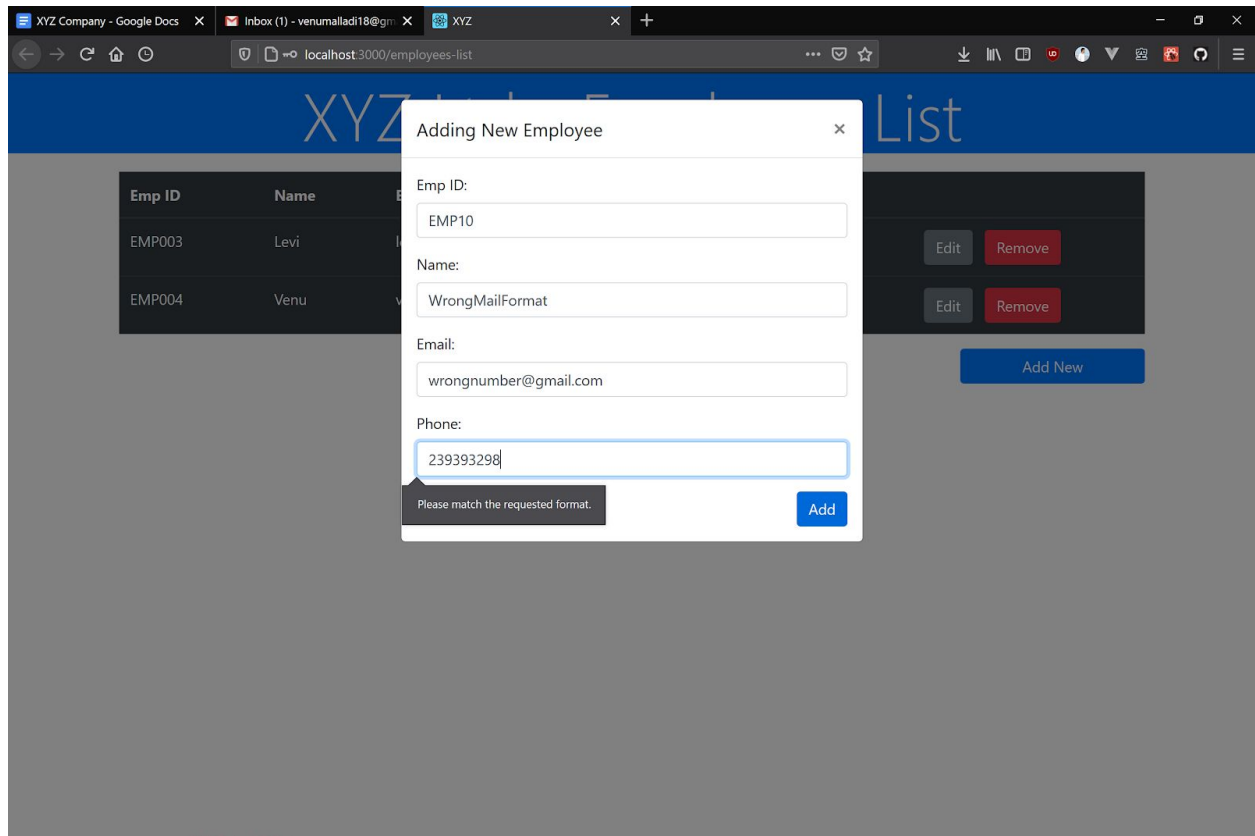


Fig-8 : (ON SUCCESSFUL VALIDATION)

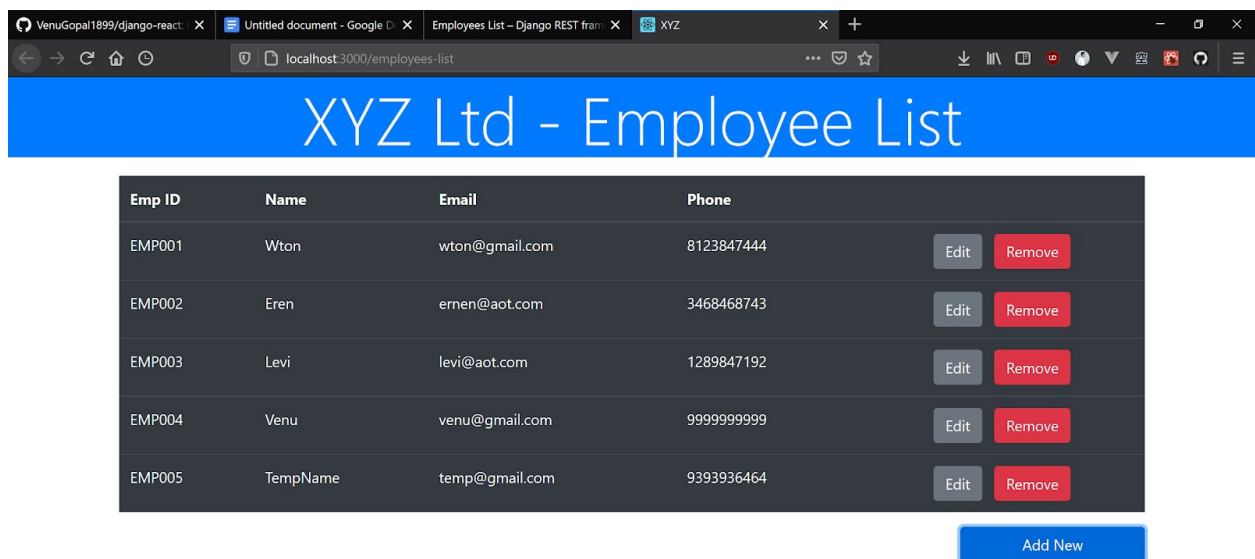


Fig-9 :

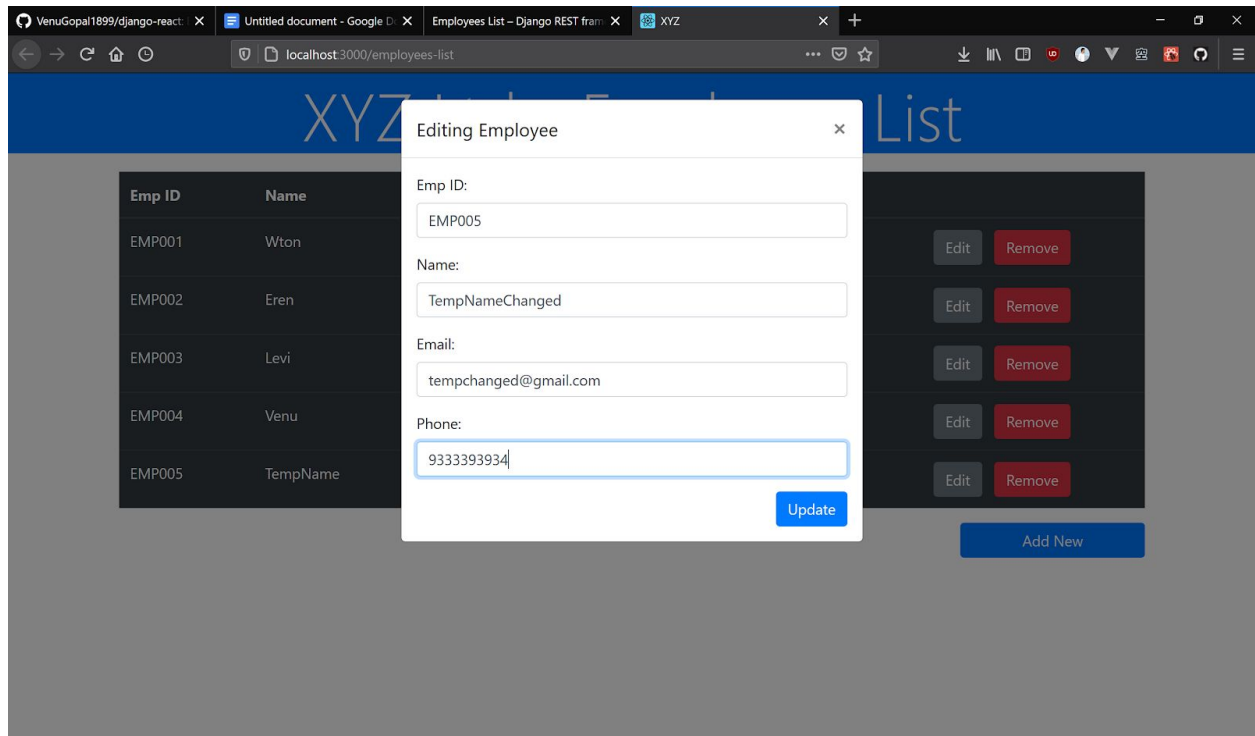


Fig-10 : (ON SUCCESSFUL VALIDATION)

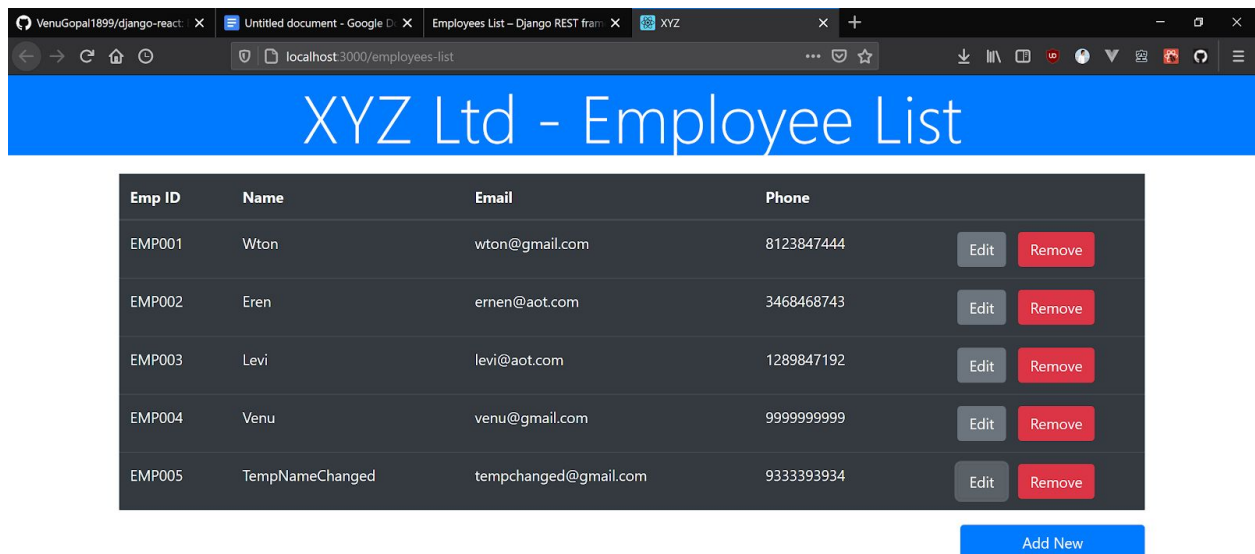


Fig-11 :

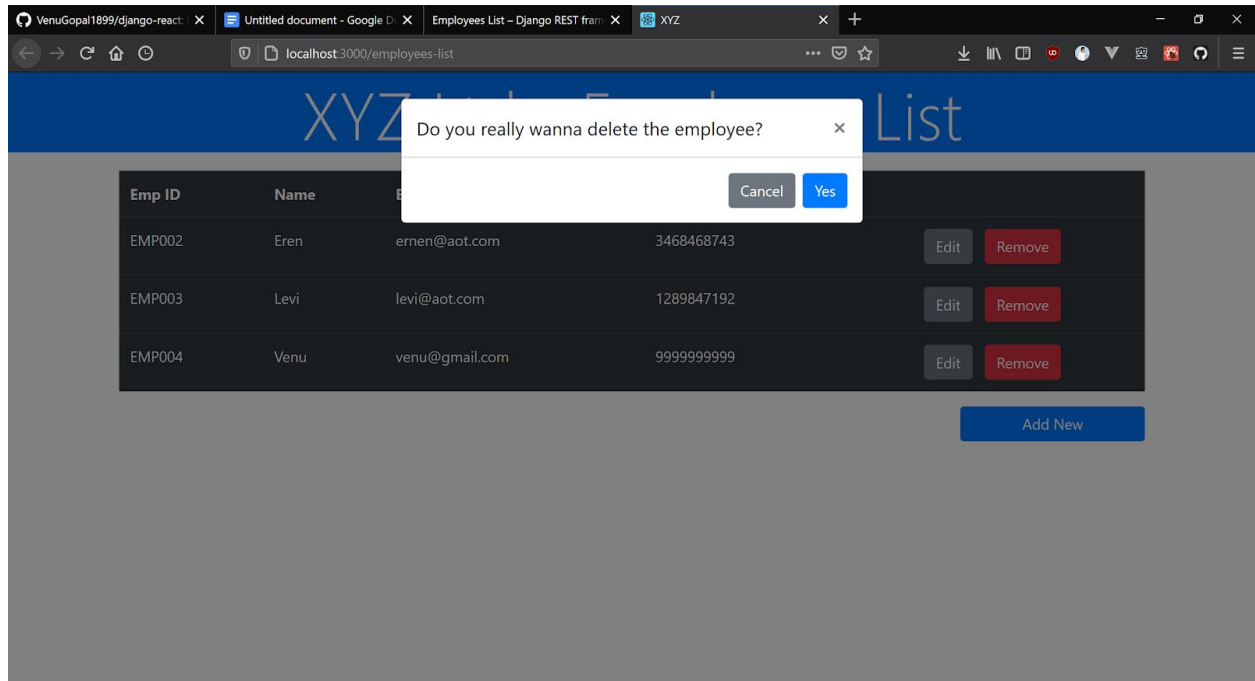


Fig-12 : (ON SUCCESSFUL DELETION)

