Energy Consumption and Generation Visualization Web Application Documentation

This project aims to develop a web application that visualizes energy consumption and generation data, providing users with insights into renewable energy usage and trends. Users can register, log in, and view personalized visualizations of energy data.

Objective

The objective of this project is to create a user-friendly web application that allows users to:

- Register and log in.
- View personalized insights into energy consumption and generation trends.
- Interact with dynamic charts and graphs to visualize energy data.
- Filter data by different energy sources and time frames.

Credentials to login for Testing the Application:

To directly login, use these credentials (which I already added to the user table):

- Url: http://d19mahtt26gbp.cloudfront.net (Deployed onto AWS CloudFront)
- username: saikumar
- password: Qwert@1234
- Backend server is running at: http://3.12.74.202:8000

Implementation:

1. **User Authentication**:
    - Implemented user registration and login functionality.
    - Secured user passwords using proper hashing techniques, using bcrypt.
    - I used JWT (JSON Web Tokens with HS256 encoding algorithm) to handle authenticated sessions after login and logging out to the user as token expiry is set to 1 day.
2. **Front-End**:
    - Developed the user interface with React using Preline UI and Tailwind CSS, ensuring device responsiveness.
    - Directed users to the dashboard after successful login.
    - Forms for user registration and login are included.
    - Implemented dynamic charts and graphs for visualizing energy data, with filters for different energy sources and time frames.
3. **Back-End**:
    - Used Python with Django and SQLite3 to create a RESTful API.
    - Managed user authentication, energy data retrieval, and filtering by date range and energy source.
    - Stored user information and energy data securely in a database.
4. **Data Visualization**:

- Integrated a JavaScript library, HighCharts, Chart.js with React for creating interactive charts and graphs.
- Different Axes charts help in visualization and offer insights into consumption vs. generation trends, highlighting renewable energy contributions.

5. **AWS Deployment:**

   - Deployed the application and server to AWS using services like EC2, S3, and CloudFront.
   - Frontend code is deployed to S3 bucket and used CloudFront to serve the FrontEnd of the Application.
   - Backend code is deployed onto EC2 instance and created a custom VPC with one private subnet and kept this EC2 in it.
   - Followed below mentioned reference links and youtube links to deploy both frontend and backend of the application onto AWS.

Installation and Setup

Prerequisites

- Node.js and npm installed globally
- Python, Django installed
- AWS account (for AWS deployment)

Local Setup

1. **Download the zip file:**

   - Extract the zip file.
   - frontend folder having react frontend code.
   - enerygy_visualization folder having python django code.

2. **Install front-end dependencies**

   - cd frontend
   - remove the node_modules folder if it exists.
   - run: npm install

3. **Install back-end dependencies**

   - Navigate to the energy_visualization Folder.
   - Run the command: pip install -r requirements.txt

4. **4.1 Start Frontend Server**

   - To start the React development server, navigate to the frontend folder
   - Now, run: npm start

   4.2 Start Backend Server

   - To start the backend server, navigate to the energy_visualization folder and follow the below commands:
     - python manage.py makemigrations
     - python manage.py migrate
     - python manage.py runserver
   - To directly login, use these credentials (which I already added to the user table):

        - username: saikumar
        - password: Qwert@1234
    - If you want to run your instance from scratch:
        - To create a DB(like Users db in my case), insert necessary SQL commands and run:
            - python manage.py createsuperuser (This creates all required tables in db.sqlite3 file).
            - Now, to insert a record user record, you can manually send a username, email, and password in the body of POST API (URL: localhost:/register)
            - Using an app like Postman API to mimic frontend registration (Because the password will be hashed and stored in a table).
            - Note: Don't use spaces in between while giving a username.
        - To add dummy data, add data into energyData.csv, which has the following Columns:
            - timestamp (In UTC format)
            - username (unique and not null)
            - energy_source (Can be Solar, Wind, Hydro, Other).
            - consumption (Units in KWh)
            - generation (Units in KWh)
        - Once dummy data is added, run the below command to insert data into the energy_data table:
            - python manage.py populate_db (Inserts csv data into energy_data table).
        - Now navigate back to energy_visualization folder and run python manage.py runserver to start the Backend Server.

5. **AWS Deployment**

    - I have deployed the backend (Python-Django) onto the AWS EC2.
    - The front end was deployed onto the AWS S3 Bucket and the AWS CloudFront.
    - Now, both frontend and backend are deployed onto AWS.
    - I added multiple resources as I approached deploying on AWS.

Note: In localhost, The applications should run at two different ports (or a different port if specified, so please add the backend endpoint into the Utils file in the frontend's src folder).

User Interface

The React-based user interface provides a responsive and user-friendly experience. Users can register, log in, and access the interactive dashboard to explore energy consumption and generation data through visually appealing charts and graphs. These visualizations offer insights into renewable energy usage trends, helping users make informed decisions regarding their energy consumption.

API and Data Storage

The backend leverages Django (a Python web framework) to create a RESTful API responsible for user authentication, data retrieval, and filtering. This API uses a

secure database to store user information and energy data.

Resources

- React: https://react.dev/
- Tailwind CSS: https://tailwindcss.com/docs/installation
- Preline CSS: https://preline.co/
- Django: https://www.djangoproject.com/
- AWS: https://aws.amazon.com/
- Chart.js: https://www.chartjs.org/docs/latest/getting-started/
- Highcharts: https://www.highcharts.com/
- https://www.youtube.com/watch?v=7djMZ5OTG_E
- https://www.youtube.com/watch?v=XOTsSJ8EOoY&t=886s
- https://www.youtube.com/watch?v=vpUwv8_w9GQ
- https://www.youtube.com/watch?v=N-gzLmYLRRk