Project Report: Django Chart Application

Author: Jawher Sakka

Date: 10/11/2024

---

1. Project Overview

This Django project is designed to demonstrate both web and REST API functionality by displaying a D3.js chart. Users can view the chart in the web app after logging in and access the chart data via a secure REST API with JWT-based authentication.

This project was developed as a coding challenge to assess technical skills in web development, REST API implementation, and data visualization.

---

2. Objectives

- **Create a secure web application** using Django with login-based access to visual content.
- **Develop a REST API** endpoint to serve chart data in JSON format, accessible only via authenticated requests.
- **Implement data visualization** on the web interface using D3.js, allowing dynamic chart display.

---

3. Project Structure

The project is organized into the following major components:

1. **Django Project (`my_chart_project`)**: The main Django project containing settings and configuration.
2. **Django App (`chart_app`)**: The custom app handling chart data and visualization.
3. **Templates**: HTML template for rendering the D3.js chart on the web page.
4. **Static Files**: JavaScript files for D3.js and custom chart-related scripts.

---

4. Technical Details

4.1 Technologies Used

- **Backend**: Django, Django REST Framework, and Simple JWT for authentication.
- **Frontend**: D3.js for dynamic chart visualization.
- **Authentication**: JWT (JSON Web Token) for secure API access.
- **Environment Management**: Pyenv and Virtualenv for Python environment, and nvm for Node.js management.

4.2 Key Libraries and Dependencies

- **Django REST Framework**: To create and manage RESTful API endpoints.
- **Simple JWT**: To secure API access with JSON Web Token (JWT) authentication.
- **D3.js**: For data-driven visualizations, specifically a line chart in this project.

---

5. Functionalities

5.1 Web Interface

- **User Login**: Users must log in to access the web interface.
- **Chart Display**: Once logged in, users can view a D3.js line chart rendered using NVD3.
- **Data Source**: Sample data is hardcoded for demonstration, but the setup allows easy expansion for real data sources.

5.2 REST API

- **Secure Access**: JWT authentication is required to access the chart data via API.
- **Data Endpoint**: `/api/chart-data/` returns JSON data, including x and y values for chart plotting.

---

6. Installation and Setup

6.1 Environment Setup

1. Install Python dependencies using **pip**.
2. Use **nvm** to install Node.js and manage versions.
3. Initialize Django’s database and create a superuser for login.

6.2 Running the Project

- Start the server with `python manage.py runserver` and access the application in a web browser.
- Log in to view the chart at `/chart/`.
- Access the API at `/api/chart-data/` with a valid JWT token.

---

7. Challenges and Solutions

- **JWT Authentication Integration**: Configuring Django REST Framework with JWT for seamless authentication.
- **Data Visualization**: Setting up D3.js within Django templates required ensuring JavaScript assets and libraries were accessible.

---

8. Future Improvements

1. **Database-Driven Chart Data**: Connect to a database to dynamically fetch data for the chart.
2. **Enhanced API Features**: Add pagination, filtering, and data transformation options.
3. **UI Enhancements**: Improve the front-end interface for a more user-friendly experience.

---

9. Conclusion

This project successfully demonstrates a Django-based web application with both a D3.js chart on the web interface and a secure REST API for data access. The use of Django, Django REST Framework, and D3.js enables a modern approach to building and securing web applications with data visualization capabilities.