Security Event & Alert Backend (Django)

Generated on: 25-12-2025

Author: Satendra Kumar

Project Overview

The Security Event & Alert Backend is a Django-based web application designed to efficiently manage security events and alerts. It provides secure and scalable REST APIs with JWT authentication, role-based access control, and real-time alert notifications using WebSockets. The system is optimized for fast querying, pagination, and can be easily switched from SQLite to Postgres for production environments.

Features

The backend system provides the following key features:

1 JWT-based authentication for secure access

2 Role-based access control (Admin / Analyst)

3 Event ingestion API for recording security events

4 Automatic alert generation based on events

5 Alert listing and filtering with optimized queries

6 Alert status update (Admin only)

7 Pagination for large datasets

8 SQLite database for development (Postgres-ready)

Tech Stack

The following technologies were used to build this backend system:

1 Python 3.12 → Core programming language with async support for WebSockets and fast development

2 Django → Web framework providing ORM, authentication, admin panel, and security features

3 Django REST Framework (DRF) → For building REST APIs, serialization, authentication, and throttling

4 SimpleJWT → JWT-based stateless authentication for APIs

5 SQLite → Lightweight database for development; easily switchable to Postgres

6 Swagger (drf-yasg) → Auto-generated API documentation and testing

7 Throttling → Rate limiting to prevent abuse or brute-force attacks

8 Channels → Async support for WebSockets and real-time communication

9 Daphne → ASGI server to run Django with WebSockets support

Project Structure

The project is organized as follows:

security_backend/ ■ ■■■ core/ ■ ■■■ models.py ■ ■■■ serializers.py ■ ■■■ views.py ■ ■■■ permissions.py ■ ■■■ urls.py ■ ■■■ consumers.py ■ ■■■ routing.py ■ ■■■ security_backend/ ■ ■■■ settings.py ■ ■■■ urls.py ■ ■■■ manage.py ■■■ README.md

Setup Instructions

To set up the project locally from scratch, follow these instructions:

1 Install Python 3 full version and venv module: sudo apt install python3-full python3-venv

2 Create virtual environment: python3 -m venv venv

3 Activate virtual environment: source venv/bin/activate

4 Install required dependencies: pip install -r requirements.txt

5 Run migrations: python manage.py migrate

6 Start the development server: python manage.py runserver

User Roles Setup

To manage users and assign roles:

1 Create superuser: python manage.py createsuperuser

2 Access admin panel: http://127.0.0.1:8000/admin/

3 Create users and assign password

4 Assign user group: Admin or Analyst

API Endpoints

The backend provides the following API endpoints:

1 Auth:

2 POST /api/token/ → Get JWT token

3 POST /api/token/refresh/ → Refresh JWT token

4 Core APIs:

5 POST /api/events/ → Create event

6 GET /api/alerts/ → List alerts

7 PATCH /api/alerts// → Update alert status (Admin only)

8 Docs:

9 Swagger → http://127.0.0.1:8000/swagger/

10 ReDoc → http://127.0.0.1:8000/redoc/