Authentication System Documentation

Overview

This Django-based Authentication System provides a secure, scalable, and flexible user authentication mechanism for modern web applications. It leverages Django REST Framework (DRF), JWT tokens, and additional security layers like email verification, 2FA, audit logging, and rate limiting.

Features

1. User Registration and Login with JWT Authentication
2. Email Verification for Account Activation
3. Password Reset via Email
4. Role-Based Access Control (RBAC)
5. Session Management with Auto Logout
6. Audit Logging of User Activities
7. Rate Limiting to Prevent Brute-Force Attacks
8. Two-Factor Authentication (2FA) Support
9. Refresh Token Blacklisting
10. Login Notification Emails
11. Trusted Devices for Reduced 2FA Prompts
12. Account Lockout on Suspicious Activity
13. OAuth2 Social Login (Google, Facebook)
14. User Activity Dashboard
15. Account Deactivation
16. Adaptive Authentication

Project Structure

The project is organized into directories for user accounts, audit logging, and the core Django project configuration, along with dependencies listed in a requirements file.

Prerequisites

You need Python (version 3.9 or higher), pip, a virtual environment tool, and an email service configured for sending verification and notification emails. Optionally, Redis or another caching backend can be used for enhanced session management and rate limiting.

Installation & Setup

Begin by extracting the project files and opening the project folder in your development environment such as VS Code. Set up a Python virtual environment and install all the required Python packages. Configure your environment variables for secrets and email service credentials. After that, run Django migrations to prepare the database and create a superuser to access the Django admin interface. Finally, start the development server to begin testing your authentication system.

Core Components

User Registration & JWT Authentication

Users can register with their details and will receive an email with an activation link. Once activated, they can log in to obtain JWT access and refresh tokens, which are used to authenticate future API requests securely.

Email Verification

The system sends an activation email to new users. The activation link is time-limited and securely generated to prevent tampering.

Password Reset

Users can request a password reset email that includes a secure, time-sensitive link to change their password safely.

Role-Based Access Control (RBAC)

User permissions are managed through groups and roles. This controls access to specific views and API endpoints based on the user’s role, for example, distinguishing between normal users and administrators.

Session Management & Auto Logout

The system supports automatic logout based on token expiration and inactivity. Tokens are revoked on logout to prevent reuse, improving security.

Audit Logging

User actions such as login, logout, and API usage are logged for auditing purposes. This enables monitoring of user activity and helps detect suspicious behavior.

Rate Limiting

To protect against brute-force attacks, the system limits failed login attempts and can lock accounts temporarily, sending notification emails on suspicious activity.

Two-Factor Authentication (2FA)

An additional authentication step using time-based one-time passwords (TOTP) via authenticator apps can be enabled. Users can also mark trusted devices to reduce 2FA prompts on familiar devices.

Advanced Features

- Users receive login notification emails when new devices or IPs are detected.
- Trusted device management allows skipping 2FA on familiar devices.
- Account lockout policies enhance security by temporarily disabling accounts after suspicious attempts.
- Social login integrations allow authentication via Google or Facebook using OAuth2.
- Users can deactivate their accounts through the system's API.
- Adaptive authentication triggers extra verification only on high-risk login attempts.
- Dashboards provide insights into user activity and security events for admins and users.

Security Considerations

Always deploy over HTTPS to protect credentials and tokens in transit. Keep secret keys and sensitive configurations in environment variables outside the source code. Maintain strong password policies and keep dependencies updated. Use audit logs to monitor unusual activity. Consider adding HTTP security headers and enforcing Content Security Policies to prevent attacks.

Extending the System

The system can be extended to integrate with front-end applications via REST APIs, add SMS-based 2FA, support CAPTCHA on login and registration forms, add finer-grained permissions, or connect with third-party monitoring tools.