**Security Best Practices**

## 1. Overview

This document details the security architecture implemented within the CV Processing Portal project. Our primary focus has been on establishing robust authentication and authorization mechanisms to safeguard both the system and sensitive user data. We've built a framework designed to ensure that only verified users—specifically Recruiters and Administrators—can access designated features, with all sensitive information securely stored and communication consistently encrypted.

## 2. Authentication

For user authentication, we leverage JSON Web Tokens (JWT). Here's how it works:

Upon successful login with valid credentials, a unique JWT is generated.

This token is then sent to the client and contains essential information such as the user's identity, their assigned role, and an expiration timestamp.

The client is subsequently required to include this token in the Authorization header of every request made to protected endpoints.

To ensure the utmost security for user credentials, all passwords are not stored directly but are instead secured in the database using BCrypt hashing. This one-way encryption prevents direct access to passwords even if the database were compromised.

## 3. Authorization (Role-Based Access Control)

Our authorization strategy employs Role-Based Access Control (RBAC), which strictly limits access to system functionalities based on a user's assigned role:

ROLE\_ADMIN: Users with this role are granted full access to the system, including critical functions like user management and the ability to view detailed audit logs.

ROLE\_RECRUITER: Users in this role have specific permissions enabling them to upload and view CVs, as well as manage job listings.

Access control is meticulously enforced through:

Annotations: We utilize annotations such as @PreAuthorize("hasRole('ADMIN')") directly within our code to define access rules for specific methods or endpoints.

JWT Claims: The roles and permissions embedded within the JWT claims are rigorously checked by security filters on the backend for every incoming request.

## 4. Token Validation

Every request targeting a protected API endpoint must include the JWT token. This is done by including it in the Authorization header in the following format:

Authorization: Bearer <token>

The backend performs a series of crucial validations for each incoming token:

Token Signature: Verifies the integrity of the token to ensure it hasn't been tampered with.

Expiration Time: Checks if the token is still valid and has not expired.

User Role and Permissions: Confirms that the user's role and associated permissions align with the requested resource.

Should the token be invalid, expired, or lack the necessary permissions, access will be promptly denied, and an appropriate HTTP status code (401 Unauthorized or 403 Forbidden) will be returned.

## 5. Secure Storage

CV files, which often contain sensitive personal information, are stored securely in AWS S3 buckets. These buckets are configured with limited public access, meaning:

Only authorized users are permitted to access these files.

To facilitate temporary and secure access, we generate pre-signed URLs. These URLs provide time-limited access to specific files without exposing the underlying storage credentials.

## 6. HTTPS Enforcement

To safeguard against data interception and ensure the confidentiality and integrity of all communications, HTTPS is strictly enforced across all environments (development, staging, and production). This encryption protocol protects all data exchanged between the frontend and backend.

## 7. Audit Logging

We maintain a comprehensive audit\_logs table to track all security-sensitive events within the system. This practice is vital for monitoring system activity and investigating any potential security incidents. Events logged include:

Successful and failed login attempts

Changes to user roles

Any attempts at unauthorized access

## 8. Rate Limiting

To fortify our defenses against brute-force attacks and other malicious automated attempts, we have implemented rate limiting on login attempts. Specifically:

The number of login attempts within a given timeframe is restricted.

After a predefined number of failed attempts, the user account will be temporarily locked out, preventing further attempts for a specified duration.

This comprehensive security architecture is designed to provide robust protection for the CV Processing Portal while ensuring scalability and ease of maintenance as the project evolves.