API Documentation Job Portal with Smart Resume–Job Matching (Secure Version)

API Overview:
This API manages candidates, recruiters, job postings, and AI-powered resume-job matching. Built with FastAPI and PostgreSQL, it follows REST principles and uses JSON requests/responses. Security best practices are implemented to prevent common attacks.

Authentication & Security:
- JWT Token-based authentication for all protected endpoints.
- Passwords are securely hashed using bcrypt.
- Input validation to prevent SQL Injection and XSS attacks.
- CSRF protection for sensitive operations.
- File uploads are validated for type and size.

Candidate Endpoints:
1. **POST /api/candidates/signup**
- Description: Register a new candidate.
- Request Body: { "name": "John Doe", "email": "john@example.com", "password": "pass123" }
- Response: { "id": 1, "name": "John Doe", "email": "john@example.com" }

2. **POST /api/candidates/login**
- Description: Candidate login.
- Request Body: { "email": "john@example.com", "password": "pass123" }
- Response: { "token": "JWT_TOKEN_STRING" }

3. **POST /api/candidates/upload_resume**
- Description: Upload candidate resume (PDF/DOC).
- Request Body: Form-data: "file": resume file
- Response: { "message": "Resume uploaded successfully", "resume_id": 101 }

4. **GET /api/candidates/recommended_jobs**
- Description: Get top AI-matched jobs.
- Response: [{ "job_id": 10, "title": "Data Scientist", "similarity_score": 0.92 }, ...]

Recruiter Endpoints:
1. **POST /api/recruiters/signup**
- Description: Register a new recruiter.
- Request Body: { "name": "ABC Corp", "email": "hr@abc.com", "password": "pass123" }
- Response: { "id": 1, "name": "ABC Corp", "email": "hr@abc.com" }

2. **POST /api/recruiters/login**
- Description: Recruiter login.
- Request Body: { "email": "hr@abc.com", "password": "pass123" }
- Response: { "token": "JWT_TOKEN_STRING" }

3. **POST /api/jobs/create**
- Description: Create a new job posting.
- Request Body: { "title": "Software Engineer", "description": "...", "skills": ["Python","React"] }
- Response: { "job_id": 10, "message": "Job created successfully" }

4. **GET /api/jobs/top_candidates/{job_id}**
- Description: Get AI-ranked candidates for a specific job.
- Response: [{ "candidate_id": 5, "name": "John Doe", "similarity_score": 0.89 }, ...]

Admin Endpoints:
1. **GET /api/admin/users**

- Description: Get all users (candidates + recruiters).
- Response: [{ "id": 1, "name": "John Doe", "role": "candidate" }, ...]

2. **GET /api/admin/jobs**
- Description: Get all job postings.
- Response: [{ "job_id": 10, "title": "Software Engineer", "recruiter": "ABC Corp" }, ...]

AI Matching Endpoint:
1. **POST /api/match/resume_job**
- Description: Compute similarity between uploaded resumes and job descriptions.
- Request Body: { "resume_id": 101, "job_id": 10 }
- Response: { "similarity_score": 0.92 }

Error Codes:
- 400: Bad Request
- 401: Unauthorized / Invalid Token
- 403: Forbidden
- 404: Resource Not Found
- 500: Internal Server Error

Security Best Practices Implemented:
- Input validation and sanitization
- Password hashing (bcrypt)
- JWT token-based authentication
- CSRF and XSS protection
- Safe file upload handling
- Parameterized queries to prevent SQL injection