Campus Job Portal

A university is launching a **Campus Job Portal (UniHire)** to connect **Students** with **Companies** for job opportunities. You are required to develop a distributed RESTful API backend application in **Spring Boot**, secured using **JWT Authentication** and role-based authorization.

1. Database Models & Relationships

The system uses five core entities to manage job postings, required skills, and student applications, creating a structure similar in complexity to the e-commerce model you provided.

Model	Attributes	Relationships
User (UserInfo in DB)	userId (Integer), username (String), password (String), roles (String: "STUDENT" or "COMPANY")	- **One-to-Many** with **Job** (Company posts many jobs).
Skill	skillId (Integer), skillName (String)	- **Many-to-Many** with **Job** (via **JobSkill**).
Job	jobId (Integer), title (String), description (String), salary (Double), postedBy (User)	- **Many-to-Many** with **Skill** (via **JobSkill**).
JobSkill	jsId (Integer), job (Job), skill (Skill)	- **Link Table** for the Job-Skill Many-to-Many relationship.
Application	applicationId (Integer), job (Job), student (User), status (String: "PENDING", "ACCEPTED", "REJECTED")	- **Unique** constraint on (Job, Student) pair.

2. Initial Data

The database is initialized with the following data:

Roles: STUDENT, COMPANY

UserInfo: (Passwords are encrypted versions of 'pass_word')

- 1, 'alice', 'pass_word', 'STUDENT'
- 2, 'bob', 'pass_word', 'STUDENT'
- 3, 'techcorp', 'pass_word', 'COMPANY'
- 4, 'finanz', 'pass_word', 'COMPANY'

Skill:

- 1, 'Java'
- 2, 'SQL'
- 3, 'React'
- 4, 'Python'

Job (Posted by 'techcorp' - userId 3):

- 1, 'Backend Intern', 'Develop REST APIs in Spring Boot.', 40000.0
- 2, 'Data Analyst', 'Analyze user data using Python.', 650000.0

JobSkill (Linking Jobs to Skills):

- Job 1 $\rightarrow$ Java, SQL
- Job 2 $\rightarrow$ Python, SQL

Application (By 'alice' - userId 1):

- 1, Job 1, User 1, 'PENDING'

3. Requirements and Authentication

Authentication and Authorization Rules

- Your job is to create the following APIs, using **JWT authentication with roles** to protect consumer and seller specific endpoints.
- All authentication and authorization processes should be implemented using **JWT Token**.
- The JWT token should be sent as a **Bearer token** in the Authorization request header.
  - Example: Authorization value would be **"Bearer <SPACE> <JWT TOKEN>"**.
- APIs preceding with /api/public are **public APIs** and can be accessed by anyone.
- APIs preceding with /api/auth/consumer are **authenticated consumer APIs**.
- APIs preceding with /api/auth/seller are **authenticated seller APIs**.
- If authenticated endpoints are accessed without JWT, return **401**.
- If a consumer endpoint is accessed with a seller JWT, or vice versa, return **403**.

General Capabilities

- **Consumers** can search, add, update, and delete items in their cart.
- **Sellers** can add, update, and delete products to the database

4. Endpoints

Public Endpoints (/api/public)

#	HTTP Method	Endpoint	Description	Status Code
1	GET	/api/public/job/search	Finds jobs matching 'keyword' in **title**, **description**, or linked **Skill Name**.	200 (or 400)
2	POST	/api/public/login	Authenticates user and returns **JWT token** (including role).	200 (or 401)

Student Endpoints (STUDENT Role - /api/auth/student)

#	HTTP Method	Endpoint	Description	Status Code
3	GET	/api/auth/student/applications	Returns all applications submitted by the student.	200
4	POST	/api/auth/student/apply/{jobId}	Submits a new application. **409** if already applied	201 (or 409)
5	DELETE	/api/auth/student/application/{applicationId}	Retracts/Deletes the student's application (must check ownership).	200 (or 404)

Company Endpoints (COMPANY Role - /api/auth/company)

#	HTTP Method	Endpoint	Description	Status Code
6	POST	/api/auth/company/job	Creates a new job posting, linking to specified skills.	201
7	GET	/api/auth/company/jobs	Returns all jobs posted by the authenticated company.	200
8	PUT	/api/auth/company/job	Updates an existing job (and its linked skills).	200 (or 404)
9	DELETE	/api/auth/company/job/{jobId}	Deletes a job posting, cascading to delete related **Applications** and **JobSkills**.	200 (or 404)
10	GET	/api/auth/company/job/{jobId}/applications	Returns all applications for a job posted by the company.	200 (or 404)
11	PUT	/api/auth/company/application/{applicationId}	Updates the status of a specific application (e.g., to "ACCEPTED").	200 (or 404)

Instructions

Install: mvn install

test: mvn test