TheKiranAcademy

Campus Requirement API Documentation

About the Project

The Campus Requirement API is a Spring Boot REST API project designed to facilitate the management of campus job postings and student applications. It allows institutions to post job opportunities and enables students to apply for these jobs. The primary entities in the system are Students, Users, Job Postings, Job Application and Interviews. The application aims to streamline the hiring process, improve communication between students and employers, and maintain an organized database of job opportunities.

Objective

✓ Automate the job application process through a REST API to enhance integration with existing systems.

✓ Ensure data accuracy with automated validation and storage of job postings and applications.

✓ Facilitate seamless communication between students and employers to improve hiring workflows.

Workflow

1. **Student Registration**:
   - A student can register using the `Student` entity and can log in to the system.
2. **Viewing Job Postings**:
   - After logging in, the student can view all available `JobPosting`s.
3. **Applying for a Job**:
   - When a student applies for a job, a new `JobApplication` is created with the status `APPLIED`. The `JobPosting` is linked to the `JobApplication`.
4. **Approval of Job Application**:
   - The job posting owner (admin/company) reviews the application. If approved, the `JobApplication` status is updated to `APPROVED`.
5. **Scheduling an Interview**:
   - After approval, the student can schedule an interview by creating an `Interview` entity, linked to the `JobApplication`.

Scope of the Project

The scope of the Campus Requirement project involves developing a REST API to manage job postings and student applications, including functionalities for handling Users, Students, Job Postings, and Job Applications.

Project Structure

Overview of Project Architecture: The project follows a layered architecture pattern, with distinct layers for handling different aspects of functionality and data management.

Layers Description

1. Controller Layer: Responsible for handling incoming HTTP requests, processing them, and returning appropriate responses.

2. Service Layer: Acts as an intermediary between the controller and the data access layer (DAO).

3. DAO (Data Access Object) Layer: Responsible for interacting with the database and performing CRUD operations.

Responsibilities of each layer

• Controller Layer:

➢ Handle incoming HTTP requests.

➢ Validate request parameters and payload.

➢ Invoke appropriate methods in the service layer.

➢ Format and return HTTP responses.

• Service Layer:

➢ Implement business logic and rules.

➢ Coordinate interactions between controllers and DAOs.

➢ Perform data manipulation and transformation.

➢ Manage transactions and ensure data integrity.

• DAO Layer:

➢ Provide CRUD operations for accessing and modifying data.

➢ Translate database queries and commands into appropriate SQL statements.

➢ Handle database transactions and connections.

➢ Abstract away database-specific details from the service layer.

Used Technology Stack

✓ Backend Framework: Spring Boot 2

✓ Database: MySQL

✓ Database Framework: Hibernate 5

✓ Java: JDK 1.8

✓ Build Tool: Maven

✓ Development Environment: Eclipse

✓ API Testing Tool: Postman

Modules in Project

1. User Module
2. Student Module
3. Job Posting Module
4. Job Application Module

5. Interview Module

In Detail About Modules and Its APIs

User Module

Description: This module manages operations related to user authentication and management.

API Endpoints:

POST /auth/login: Authenticates a user based on the provided credentials.

POST /user/createUser: Creates a new user record.

DELETE /user/deleteUser/{username}: Deletes a user by username.

GET /user/allUsers: Retrieves a list of all registered users.

GET /user/getUser/{username}: Fetches a user by username.

PUT /user/updateUser: Updates user details.

Student Module

Description: This module manages operations related to student registration and management.

API Endpoints:

POST /students/register: Registers a new student.

GET /students/allStudents: Retrieves a list of all registered students.

DELETE /students/deleteStudent/{id}: Deletes a student by ID.

PUT /students/updateStudent: Updates student details.

Job Posting Module

Description: This module handles job postings and enables institutions to manage job opportunities.

API Endpoints:

POST /job-postings/createJob: Creates a new job posting.

GET /job-postings/allJobs: Retrieves a list of all job postings.

DELETE /job-postings/deleteJob/{id}: Deletes a job posting by ID.

PUT /job-postings/updateJob: Updates job posting details.

Job Application Module

Description: This module allows students to apply for job postings.

API Endpoints:

POST /jobApplication/apply: Submits a job application.

PUT /jobApplication/updateStatus/{id}/{status}: Updates the status of a job application.

Interview Module

Description: This module manages the interview process for job applications, allowing students and employers to schedule and track interviews.

API Endpoints:

POST /interview/schedule

GET /interview/{id}

PUT /interview/updateStatus/{id}/{status

Sample Interview Questions