TRNG- 1876 \_ REVHIRE

Application Overview

The RevHire is a console based application that acts as a job portal focused on connecting job seekers and employers. This application enables the registered employers to post their job requirements and also let the registered job seekers find suitable jobs for their profile.

Job seekers can create profiles, textual and formatted resumes, and apply for jobs. Employers can manage jobs, view applicant details, and shortlist/reject applications.

Core Functional Requirements

As a Job Seeker, I should be able to:

1. Register myself and create an account.
2. Login to my account.
3. Create and manage standard textual resume.
4. Search for jobs using filters like job role, location, experience in years, and company name.
5. Apply for interesting jobs (No Limit).
6. View Applications and their status.
7. Withdraw the application (from the interested jobs) in case of a change of interest.

As an Employer, I should be able to:

1. Create a profile and register my company.
2. Login using the registered account.
3. List and manage job postings.
4. View the details of applicants for each job posting.
5. Shortlist or reject the application for the selected job posting.

Standard Functional Scope

Apart from standard registration and login, the user should be able to change their password and request for a forgotten password (The registered password will be sent to the registered email id).

Definition of Done

* Working application demonstration.
* Sharing the associates’ code repo for technical review with:
* ERD Diagram
* Application Architecture Diagram

Competency wise scoping

|  |  |  |
| --- | --- | --- |
| **Competency** | **Application Type** | **Expectations** |
| Java SQL REST / Java SQL REST Unix / Java SQL REST Gradle / C# SQL REST / Python SQL / JavaScript SQL / JavaScript NoSQL | Console Based Application | **User Inputs:**   1. Ability to accept the user inputs from console 2. Providing recommended format in which the user to key in the inputs 3. Validate the user given inputs for format and convert to appropriate type for application usage.     **System outputs:**   1. Use formatted outputs for better readability and understanding. E.g., currency and date values should be formatted well. 2. Display the reports in the appropriate format such as tables etc.     **User Navigation:**   1. Provide a number-based menu items for the user to navigate for different use cases 2. Handling user selections and providing appropriate screen / feature to the user     **Validation and Error Handling:**   1. Validate the user inputs for its types and format. 2. Display functional related user messages (either for input/error/output) - no system error codes or SQL error codes. 3. Handle the exceptions and errors gracefully.     **Logging:**   1. Ensure the application is using proper logging framework and methods. 2. Ensure the application’s log level is configured using configuration files so that it can be changed without changing the code. 3. Also ensure that the application logging is configured to output to the mentioned log file.     **Testing**:   1. Ensure sufficient test cases are written using appropriate testing frameworks. 2. Ensure the code coverage closed to be 80%     **Security**:   1. Ensure the SQL injection threat is taken care.     **Coding Standard:**   1. Use the industry coding standards and conventions. 2. Modular based code development for better reusability. 3. Ensure proper usage of resource objects such as database connectivity objects to avoid resource leakages. 4. Ensure proper usage of design patterns and application layering (such as Business Service, DAO Layer etc.) wherever applicable. |