retailo

Hiring Task

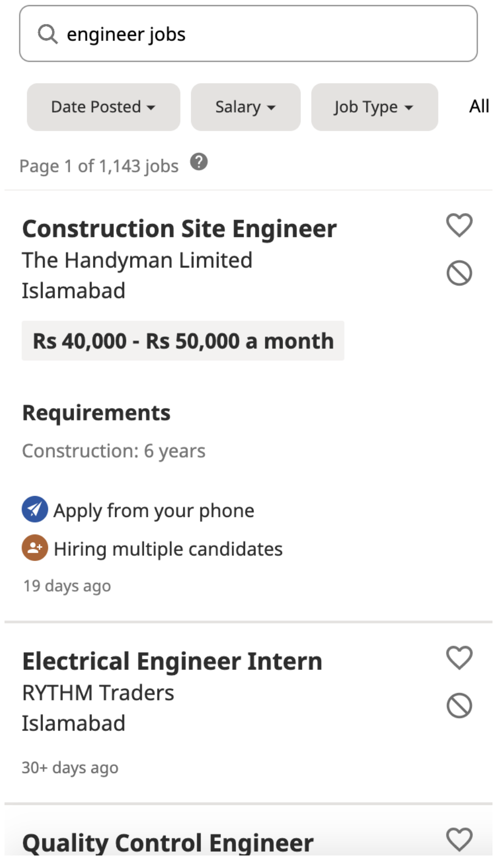
Time To Complete	2-3 days	**Submission via**	GitHub
Position	Entry-level Engineer	**Task #**	BEF-1

Objective

To get insights into your knowledge of backend systems, aptitude and learning flexibility for newer technologies.

Task Description

We would like to build a search engine for jobs where a user can input a job title into the search-field and get matching job listings. This task will revolve around building a backend system for such a job-site app.



Page 1 of 1,143 jobs

Construction Site Engineer
The Handyman Limited
Islamabad

Rs 40,000 - Rs 50,000 a month

Requirements
Construction: 6 years

Apply from your phone
Hiring multiple candidates
19 days ago

Electrical Engineer Intern
RYTHM Traders
Islamabad
30+ days ago

Quality Control Engineer

Job-listing entity is defined with the following attributes:

```
{
  "id": 1, (unique identifier)
  "job_title": "Mechanical Systems Engineer",
  "company": "Plambee",
  "location": "New York",
  "post_date": "9/29/2020", (mm/dd/yyyy)
  "apply_email": "lcotton0@alibaba.com",
  "leave_type": "Monthly" / "Weekly" / "Seldom"
}
```

Write a Node.js app that exposes RESTful APIs as following:

- `jobs/search?query=engineer,mechanic`
  This will be a `GET` request which will return results against the input query parameter.
- `jobs/create`
  This will be a `POST` request which can create a new job-post on our portal and will save the data in our database.
- `jobs/delete`
  This will be a `DELETE` request which will take `job_id` as a unique identifier and will delete the corresponding job from our database.

Data Source

Use an online data generator such as mockaroo and integrate their API in your system to generate data in real time.

Task Achievements

❏ Generate fake data for job listings according to the entity specification given above and return it in the API response. Don't worry about saving this data in the local database.

❏ Enable the user to create a job-listing via the `POST` endpoint. Think about what kind of database should be used for this use-case. Moreover, ensure that all such user created results are always returned in the search against matching job title, along with generated fake results.

❏ The user should be able to delete the job-listing by providing `job_id`. Ensure that your app doesn't crash when an invalid `job_id` is provided.

❏ The `query` parameter should support multiple search keywords, separated by comma.

- ❏ Implement a mechanism to optimize search APIs performance such that it does not return thousands of results all at once. There should be a way that those results are returned in chunks.
- ❏ Implement sorting and filtering on search API. Think about what reasonable filters and sorting values would make sense and implement those. Make sure those are documented in the API docs in README.
- ❏ Ensure your code has no coding mistakes and errors by integrating ESLint and setting up this lint configuration. Fix all errors raised by this configuration and your codebase should have zero warnings/errors.
- ❏ Project repository (submission on Github) should include a README file that lists instructions on how to setup and run the app.

Bonus Tasks

- ❏ Set up an automated mechanism of verifying that all API endpoints (searching, creating, deleting) behave correctly when valid/invalid data is supplied.
- ❏ Dockerize the application installation and server setup.