Explanation

API Overview:

The Job Finder API allows users to search for job listings based on specific criteria. The API fetches job listings from multiple job platforms like LinkedIn, Indeed, and Google Jobs and then filters these listings based on the user-defined search criteria.

Endpoints:

1. GET /

Description: A simple health check endpoint that returns a welcome message and the current API version.

Response:


```
{
 "message": "Welcome to Job Finder API",
 "version": "1.0.0"
}
```


2. POST /api/jobs/search

- **Description**: This endpoint allows users to search for job listings based on specific criteria. Users can specify details such as job position, experience level, salary, job nature, location, and skills.

Request Body (JobSearchCriteria):


```
{
 "position": "Software Engineer",
 "experience": "3-5 years",
 "salary": "70000",
 "jobNature": "Full-time",
 "location": "New York",
 "skills": "Python, JavaScript"
```

}

Response Body (JobSearchResults):

{

"relevant_jobs": [

{

"job_title": "Software Engineer",

"company": "TechCorp",

"experience": "3-5 years",

"jobNature": "Full-time",

"location": "New York",

"salary": "70000 USD",

"apply_link": "https://company.com/job/12345"

},

]

}

- **Explanation**: The endpoint processes the job search criteria and returns a list of relevant job listings. These listings are filtered based on the search parameters.

Job Matching Process:

1. Job Data Scraping

The API scrapes job listings from various job platforms (LinkedIn, Indeed, and Google Jobs) using the scrape_jobs function. The scraping logic fetches job listings in raw form (e.g., title, company, location, job description, etc.).

2. Formatting the Jobs

The raw job data is then formatted using the format_jobs function. This function converts the raw data into a structured format suitable for the API response. Here, the following actions are performed:

- **Job Nature**: Determines whether the job is remote or onsite based on the is_remote flag.
- **Salary**: The minimum and maximum salary are extracted, and if they are available, they are displayed in a specific format.
- **Experience**: The experience requirement is typically not provided directly in job listings, so it is marked as "not found."

The output of this step is a list of formatted job listings that include essential job details such as job title, company, experience, salary, job nature, location, and an application link.

3. Filtering Jobs

After formatting the jobs, the filtering process begins. The API uses a **HugChat bot** (or an alternative OpenAI model) to determine whether each job is relevant to the user's search criteria.

- **HugChat**: A chatbot is used to evaluate the relevance of each job listing based on the search criteria provided by the user. The bot is trained to assess whether a job matches the user's specifications, such as the job position, experience, salary range, job nature, location, and required skills.

The chatbot processes jobs in batches (2 jobs at a time), and for each job, it responds with either "Yes" or "No" based on relevance.

Steps:

1. The search criteria (e.g., job position, experience) are combined into a user-friendly query.
2. A batch of jobs is sent to the chatbot for evaluation.
3. The bot checks if each job is relevant and returns the results.

- **Fallback with OpenAI**: If HugChat fails or encounters issues, the API can fall back on the OpenAI GPT model to filter jobs. This fallback mechanism works similarly by processing the job listings and comparing them to the user's criteria.

4. Returning Results

Once the jobs are filtered, the relevant ones are returned in the response body under the relevant_jobs field. This list includes the jobs that match the user's criteria, as determined by the HugChat bot or OpenAI model.

Screenshots:

- **Job Search Request**: The following is a screenshot showing the process of sending a POST request with search criteria:

Request Body:

```
{
    "position": "Data Scientist",
    "experience": "5+ years",
    "salary": "100000",
    "jobNature": "Full-time",
    "location": "San Francisco",
    "skills": "Python, Machine Learning"
}
```

- **Job Search Response**: After processing, the following is an example of a successful job search response:


```
{
"relevant_jobs": [
{
"job_title": "Senior Data Scientist",
"company": "DataTech Corp",
"experience": "5+ years",
"jobNature": "Full-time",
"location": "San Francisco",
"salary": "100000 USD",
"apply_link": "https://company.com/job/98765"
}
]
}
```

Conclusion:

This API simplifies the process of job searching by automating the job matching and filtering based on user preferences. Through the integration of HugChat or OpenAI, the system intelligently filters out irrelevant job postings, ensuring that users receive the most suitable job recommendations based on their input criteria.