**Job Finder API Documentation**

**Overview:** The Job Finder API is a FastAPI-based service that enables users to search for relevant job listings from multiple job portals, specifically LinkedIn and Glassdoor. It leverages web scraping techniques combined with advanced natural language processing (NLP) models to semantically analyze job descriptions and match them against user-defined criteria, returning only the most relevant job opportunities.

**Technologies and Libraries Used:**

• FastAPI: Web framework for building the API.

• Pydantic: Data validation and settings management using Python type annotations.

• Selenium WebDriver: Browser automation tool used for scraping dynamic job listings.

• WebDriver Manager: Automatically manages browser driver binaries.

• BeautifulSoup: HTML parsing library for extracting job details from page source.

• SentenceTransformers: Provides pre-trained transformer models for generating sentence embeddings.

• SciPy: Used for calculating cosine similarity between embeddings.

• Transformers & PyTorch (optional for LLaMA integration): For loading and using LLaMA language models.

• Requests: For HTTP requests (e.g., fetching location IDs from Glassdoor).

• Concurrent Futures: For parallel execution of scraping tasks.

• NumPy: Numerical operations, used in similarity calculations.

• Uvicorn: ASGI server for running FastAPI applications.

**--> Install all the dependencies first**

**--> Code with Hugging Face Transformers as well as Meta Llama 3.2 1B is provided**

**(Llama code is commented)**

**---> Indeed uses Cloudfare captcha that cannot be bypassed using selenium or any other open source API that’s why I have not scrapped Indeed. In place of Indeed I have scrapped Glassdoor.**

**API Endpoint:**

**POST /search\_jobs**

Request Body (JSON): { "position": "string", // Job title or position to search for "experience": "string", // Required experience level (e.g., "2+ years") "salary": "string", // Desired salary (e.g., "100,000 PKR") "jobNature": "string", // Job nature (e.g., "onsite", "remote") "location": "string", // Job location (e.g., "Islamabad, Pakistan") "skills": "string" // Required skills (e.g., "Python, React") }

Response Body (JSON): { "relevant\_jobs": [ { "job\_title": "Full Stack Engineer", "company": "XYZ Pvt Ltd", "experience": "2+ years", "jobNature": "onsite", "location": "Islamabad, Pakistan", "salary": "100,000 PKR", "apply\_link": "https://linkedin.com/job123" }, ... ] }

**How Job Matching Works:**

**1. Scraping:**

* The API uses Selenium WebDriver to automate browser interactions and scrape job listings from LinkedIn and Glassdoor.
* It handles dynamic content loading, modal dialogs, and pagination to collect comprehensive job data.
* For each job, detailed information is extracted including job title, company, location, job description, salary, and application link.

**2. Semantic Relevance Analysis:**

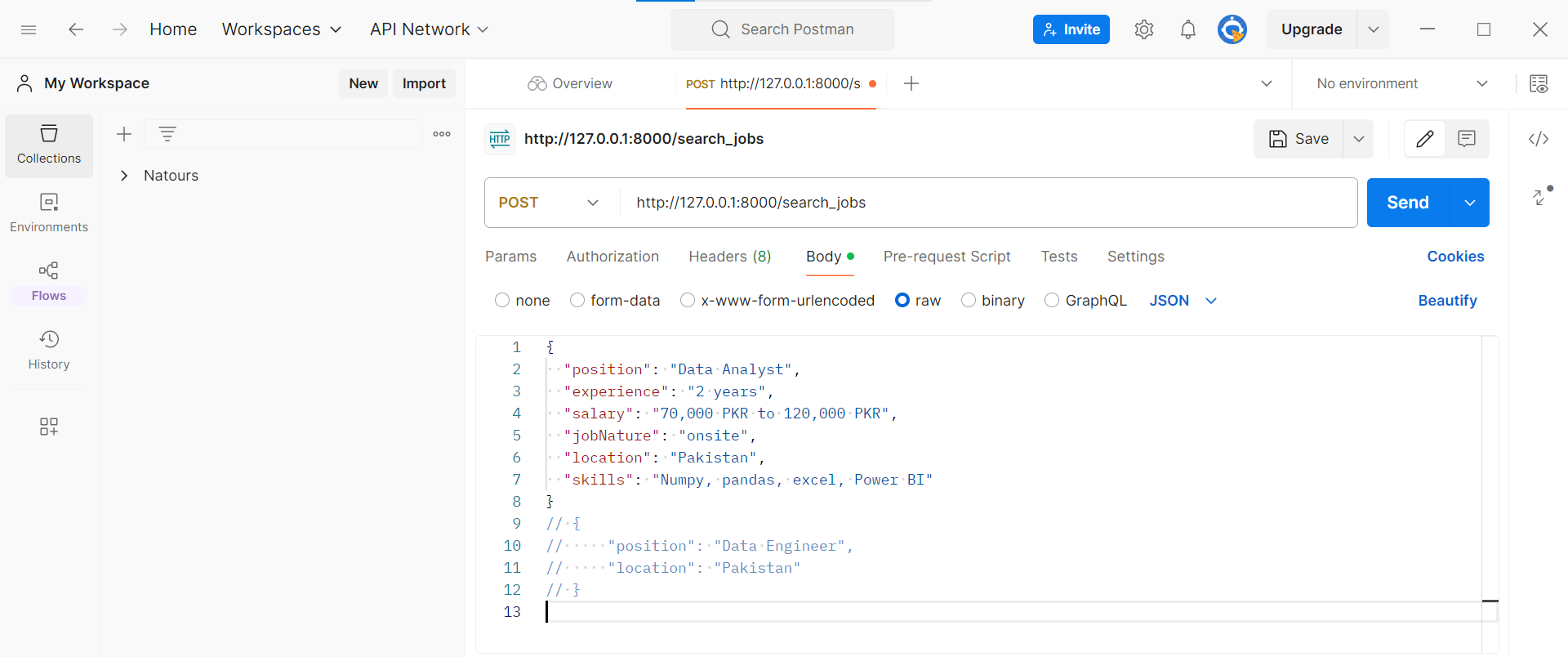
* The user’s search criteria (position and skills) and each job’s description are converted into vector embeddings using a pre-trained SentenceTransformer model ('all-MiniLM-L6-v2').
* Cosine similarity between the user query embedding and job description embedding is computed.
* Jobs with similarity above a threshold (0.5) or where the job title contains the position keyword are considered relevant.

**3. Response Construction:**

* Only relevant jobs are included in the API response.
* The response JSON contains an array of job objects with standardized fields for easy consumption by clients.
* Optional LLaMA Model Integration:
* The system can be adapted to use a LLaMA language model for embedding generation instead of Sentence Transformer.
* This involves loading the LLaMA model via Hugging Face transformers and computing embeddings with mean pooling of hidden states.
* The similarity calculation and filtering logic remain the same.

**Summary:** This API combines robust web scraping with state-of-the-art NLP to provide users with highly relevant job listings tailored to their specific criteria. It abstracts the complexity of multiple job portals and semantic matching into a simple, unified interface.

**INPUT RESPONSE**



**OUTPUT RESPONSE**

