Job Search Application

This project is a job search application that uses the JSearch API to fetch job listings and display them on a web page. The application allows users to save job searches and load them later.

How It Works

The application consists of a frontend and a backend:

- **Frontend**: The frontend is a simple HTML page that displays job listings and allows users to save and load searches.
- **Backend**: The backend is a Node.js server that interacts with the JSearch API to fetch job listings and save/load searches to/from JSON files.

JSearch API

The application uses the JSearch API by OpenWeb Ninja to fetch job listings. JSearch is a fast, reliable, and comprehensive jobs API that provides real-time job postings and salary information from Google for Jobs. You can learn more about the API and get an API key by creating a free account here.

How to Use

1. **Get an API Key**: Sign up for a free account on RapidAPI and get your API key for the JSearch API.
2. **Add the API Key**: Add your API key to line 48 in the `server.js` file:

```
headers: {
  "x-rapidapi-key": "your-api-key-here",
  "x-rapidapi-host": "jsearch.p.rapidapi.com",
},
```

3. **Run the backend**: Start the Node.js server by running the following command in the terminal:

```
cd frontend
npm install
node server.js
```

The server will start running on `http://localhost:3000`.

Project Structure

- **frontend/**: Contains the frontend code.
  - **data.json**: Stores the fetched job data.
  - **public/**: Contains public assets.
    - **index.html**: The main HTML file that displays job listings.
  - **saved_searches/**: Directory to store saved searches.

- **server.js**: The Node.js server file.
- **jsearchapi.py**: Python script to interact with the JSearch API (optional).

API Overview

JSearch by OpenWeb Ninja is a fast, reliable, and comprehensive jobs API. It provides real-time job postings and salary information from Google for Jobs. The API supports various filters and options to customize job searches. For more details, refer to the JSearch API documentation. https://rapidapi.com/letscrape-6bRBa3QguO5/api/jsearch

The `jsearchapi.py` file is a separate Python script that interacts with the JSearch API. It is used to fetch job listings and export them to a JSON file. This script is independent of the main JavaScript project and can be used as an alternative way to interact with the JSearch API.

How It Works

1. **Define Query Parameters**: The script defines query parameters such as the search query, page number, number of pages, country, and date posted.
2. **Set Headers**: The script sets the headers for the HTTP request, including the API key and host.
3. **Make API Request**: The script makes a GET request to the JSearch API using the `requests` library.
4. **Process Response**: The script processes the JSON response and prettifies it for readability.
5. **Export to JSON**: The script exports the prettified JSON response to a file named `data.json` in the frontend directory.

Running the Script

To run the `jsearchapi.py` script, follow these steps:

1. **Install Dependencies**: Ensure you have the `requests` library installed. You can install it using pip:

   ```
   pip install requests
   ```

2. **Run the Script**: Execute the script using Python:

   ```
   python jsearchapi.py
   ```

   The script will fetch job listings from the JSearch API and save them to `data.json`.

Security Vulnerabilities

This prototype project has several potential security vulnerabilities:

- **API Key Exposure**: The API key is hardcoded in the server.js file, which can be exposed if the code is shared publicly. Consider using environment variables to store sensitive information.
- **Lack of Input Validation**: The project does not validate user inputs, which can lead to security issues such as SQL injection or cross-site scripting (XSS). Implement input validation to ensure data integrity.

- **No Authentication**: The project does not implement any authentication or authorization mechanisms, making it vulnerable to unauthorized access. Add authentication to secure access.
- **Insecure Data Storage**: Job data and saved searches are stored in JSON files without encryption, which can be accessed and modified by unauthorized users. Use encryption to secure stored data.

Contact

For API integration support, you can reach out to OpenWeb Ninja:

- **Email**: support@openwebninja.com
- **Discord**: Join Discord