COMP-8547 Advanced Computing Concepts

Final Project - Salary Stratos

Variant 9: Top Paid Job Analysis

Group – 4 : Data Wizards

Team Members:

110127749 - Deon Victor Lobo (Team Lead)

110123330 - Gagandeep Singh

110126400 - Rishit Devang Bhojak

110128309 - Rahul Harish Patel

110124682 - Smit Hiteshkumar Patel

Submitted to: Prof.(Dr.) Olena Syrotkina

Contents

Variant 9

Salary Stratos - Top Paid Job Analysis

The "Top Paid Job Analysis" project digs deep into understanding the connection between job details and pay structures. Its main goal is to find the highest-paying jobs in a specific field using data from three major job listing websites: Simply Hired, Remote Ok, and Glass Door.

Salary Stratos is all about having clear search rules and following project instructions closely to get data efficiently. The main focus is to use optimal data structures and algorithms to store the data and get results which help the users search for jobs crawled from multiple sites. The big picture goal is to find meaningful patterns in the data, giving a solid understanding of the top-paying jobs in the chosen field.

What makes this Salary Stratos special is the easy-to-use interface for users to search for jobs using keywords. The website shows results from all three sites, using page ranking to highlight the most relevant and high-paying jobs. This feature is helpful for job seekers, employers, and anyone interested in the industry, providing a methodical and valuable look into the job market.

TASKS:

1. Choose at least three websites to crawl (Simply Hired, Remote Ok, Glass Door)
2. Specify the details of your search (jobTitle, companyName, salary, etc.)
3. Implement required features.
4. Show relevant jobs from all websites based on user search.

FEATURES:

1. Web crawler
2. Data validation using regular expressions;
3. HTML parser;
4. Inverted indexing : Inverted indexing enables rapid searches without the need to scan through entire files. This is achieved through an index data structure that stores a mapping from content, such as words or numbers, to its respective locations within a collection of documents.
5. Frequency count : The frequency count displays the user the instances of a word within a particular URL.
6. Page ranking : Page ranking gauges the significance of a search result by considering the frequency of occurrences. If search keywords appear more frequently within a web page, that page receives a higher ranking. Sorting, heaps, or other data structures can be employed to carry out the ranking of web pages.
7. Spell checking : Spell checking is accomplished by creating a vocabulary derived from all words present in text files. If no matches are identified, the system should offer alternative word suggestions. The edit distance algorithm is employed to compare the user's input with words found in the source files.
8. Word completion : Provide word suggestions when the user has entered text.
9. Search frequency : Capability to display previously searched words along with the corresponding search frequency.
10. Finding patterns using regular expressions.
11. Compare sorting algorithms for Page ranking
12. Recommend top paid jobs based on search results

Roles of group members:

Name	ID	Feature	Java Files and Function Names	Description of work
Rishit Devang Bhojak	110126400	Web Crawling for SimplyHired, RemoteOK and GlassDoor	**SimplyHiredScrapper.java** - crawlWebPage **RemoteOk.java** - crawlWebPage **GlassDoorScrapper.java** - crawlWebPage	Crawl the website using selenium and get the page source
Smit Hiteshkumar Patel	110124682	HTML Parser for SimplyHired, RemoteOK and GlassDoor	**SimplyHiredScrapper.java** - scrapeWebPage **RemoteOk.java** - scrapeWebPage **GlassDoorScrapper.java** - scrapeWebPage	Parsing and extracting data from the page source like title, company name, salary etc using jsoup.
Gagandeep Singh	110123330	Web Crawling for SimplyHired and GlassDoor	**SimplyHiredScrapper.java** - scrapJobLinks **GlassDoorScrapper.java** - scrapJobLinks **ScrapperBot.java** - initializeScraperBot, getScraperBot, getScraperBotWithWait	Extract all the job links from the search page so that each job link can be crawled individually. Create a Singleton for scraper bot so that there is only one running instance of the chrome driver throughout the program
Deon Victor Lobo	110127749	Web Crawling for Remote Ok	**RemoteOk.java** - scrapeWebPage **ScrapperBot.java** - saveAndAppendToJson, readJsonFile **Jobs.java**	Extracting elements using jsoup. Maintain a database.json in the resources folder containing all the jobs Create a model for the Job to store
Deon Victor Lobo	110127749	Data validation using regular expressions	**DataValidation.java** - validateDataForOneObject, validateField	This will take a job object as input and return true only if all fields of the job are valid. We add only those Jobs to the database,json which are validated

Gagandeep Singh, Deon Victor Lobo	110123330, 110127749	Trie DataStructure, Ignore stopwords	**TrieNode.java -** TrieNode **JobDataTrie.java -** initializeTrie	Each trie node has children, jobIds and a TreeMap that stores frequencies and jobIds for any given word. Ignore any stop words during insertion.
Rahul Harish Patel	110128309	Inverted indexing	**JobDataTrie.java -** initializeTrie, insertIntoTrie	Insert word into the trie and maintain a index of the word that is inserted and the corresponding jobId location
Smit Hiteshkumar Patel	110124682	Frequency count	**JobDataTrie.java -** initializeTrie, insertIntoTrie	When inserting into the trie keep track of the word, the frequency of the word and the jobId location
Rahul Harish Patel	110128309	Page Ranking	**PageRanking.java -** searchInvertedIndexedData	Fetch the searchTerms and frequencies from the Trie and rank based on frequency.
Gagandeep Singh	110123330	Spell checking	**SpellChecker.java -** initializeSpellChecker, suggestSimilarWord	Maintains a dictionary of words, if word matches the search terms it returns the word is valid else it uses EditDistance Algorithm to give the closest match to the word
Gagandeep Singh	110123330	Word completion	**WordCompletion.java -** getWordSuggestions **TrieDS.java -** getWordSuggestions, collectWords	Searches the trie for prefix of the word. Retrieves all the suggestions from the trie and sorts based on frequency.
Gagandeep Singh	110123330	Sorted Array Data Structure	**SortedArray.java**	Maintains an Array List data structure of objects which uses comparator and binary search algorithm to insert elements in the array in sorted order.

Deon Victor Lobo	110127749	Search frequency	**SearchFrequency.java -** displaySearchFrequencies, updateSearchFrequency	Maintains a LRUCache of all the words that were searched. Deletes the least recently used search word if the cache exceeds the limit. It also returns the last 10 recently used words and their frequencies in sorted order.
Rishit Devang Bhojak	110126400	Finding patterns using regular expressions	**SimplyHiredScrapper.java** - scrapeWebPage **RemoteOk.java** - scrapeWebPage **GlassDoorScrapper.java** - scrapeWebPage	Used regular expression to extract salaries in various formats and convert them to yearly format. Eg: Use RegEx to match salaries in hourly format and convert them to yearly format.
Deon Victor Lobo	110127749	Compare sorting algorithms for Page ranking	**CompareRunTimes.java**	Implement inverted indexing using Binary Search, MergeSort, Quick Sort and compare the runtimes.
Deon Victor Lobo, Gagandeep Singh	11012774, 110123330	Recommend top paid jobs based on search results	**PageRanking.java -** searchInvertedIndexedDataBySalary	Get data from trie and sort based on salary to get the top salaries. If multiple jobs have the same salary sort based on frequency. Hence the cost for sorting is salary and frequency.
Gagandeep Singh, Deon Victor Lobo	11012333, 110127749	Integration of all the features. User Interface of all the features	**Feature Integration - FeatureController.java**	The most tricky part of integration was done using Spring and React. We integrated the features using Spring Boot java to create APIs. UI is created using React which provides an excellent user interface for users to interact.

Description of Data structure and Algorithms for Features:

Features	Data Structure/Algorithm and Explanation according to features
Web Crawling primarily by Rishit Devang Bhojak	**Queue :** A Queue is used to store job links. In this case, it holds the links to job pages that will be processed. The queue data structure is suitable here because it follows the First In, First Out (FIFO) principle, allowing you to process job links in the order they are discovered. **Set :** A Set is used to store unique job identifiers. This helps in ensuring that duplicate jobs are not processed. The HashSet implementation is chosen because it provides constant-time complexity for basic operations, such as add, contains, and remove **Summary :** We are using Selenium to crawl the data of various jobs from different websites and save the page source. This page source is later used from scrapping.
HTML Parser primarily by Smit Hiteshkumar Patel	**Jsoup Library :** Library used for parsing HTML documents. It provides methods like select() to extract data from HTML documents in a convenient way. **Summary :** The scrapeJobData method takes the HTML source code of a job page. It uses Jsoup to parse the HTML, extracts relevant information such as job title, company name, location, salary, and job description. The salary information is parsed based on various regular expressions representing different salary formats. Finally, a Job object is created, populated with the extracted information, and returned. The method is designed to scrape job data from page source and prepare it for further processing or storage.
Data validation using regular expressions primarily by Deon Victor Lobo	**AtomicBoolean :** AtomicBoolean is used to ensure thread safety when updating the allValid status across different fields. It ensures that the boolean variable can be safely updated by multiple threads. **Regular Expressions (regex) :** Regular expressions are used for pattern matching and validating various fields such as job title, company name, link, salary, location, and description. Regex is a powerful tool for defining search patterns and validating strings against those patterns. **Summary :** The code ensures that all fields are validated, and if any field fails validation, the overall validation status is set to false. The method returns a boolean indicating whether all fields are valid or not. During scraping the jobs objects are only added to the database.json if all data fields are valid.

Inverted Indexing primarily by Rahul Harish Patel	**Trie Data Structure :** The main data structure used here is a trie, represented by the TrieDS class. It is employed to efficiently store and retrieve words from the job descriptions. **Set<String> stopWords:** A HashSet is used to store a set of common English stop words. This set is used to filter out irrelevant words during trie initialization. **TrieNode:** Each node in the trie is represented by the TrieNode class. It contains a map of child nodes (children), a set of job IDs (jobIds) and a boolean flag (isEndOfWord) indicating the end of a word. **Job IDs and Trie Node:** The jobIds set in each trie node is used to store the unique identifiers of jobs where the corresponding word appears. This facilitates quick retrieval of jobs containing specific words during searches. **Summary :** I used a trie for inverted indexing to efficiently store words from job descriptions. Trie allows for quick prefix searches and is suitable for storing and retrieving words. Inverted indexing is achieved by constructing a trie where each node represents a character in a word. The words are indexed in a way that allows for efficient prefix searches. For each word, a set of job IDs is stored, indicating the jobs in which the word appears.
Frequency count primarily by Smit Hiteshkumar Patel	**TrieNode:** Each node in the trie is represented by the TrieNode class. It contains a TreeMap to store word frequencies (wordFrequency). **TreeMap (wordFrequency) :** It stores word frequencies for each job ID, facilitating easy retrieval and analysis of word occurrences. **Summary :** This facilitates quick retrieval of jobs and their frequencies containing specific words during searches. It also allows for sorting words based on their frequencies.
Page ranking primarily by Rahul Harish Patel	**Trie (TrieDS):** The TrieDS object is used for performing searches on inverted indexed data. It efficiently retrieves information related to search terms from the trie. **SortedArray (SortedArray<Job>):** A sorted array is employed to maintain a collection of Job objects sorted based on their word frequencies. The Comparator.comparingInt(Job::getWordFrequency) ensures sorting in ascending order of word frequency. It uses binary search to sort when inserting into the array based on the frequency comparator. This is fast because the insertion happens with a worst case linear time complexity. **Summary :** The searchInvertedIndexedData method performs page ranking by

	considering the word frequencies of the search terms within the inverted index. It iterates through the search terms, retrieves relevant information from the trie, and updates the SortedArray<Job> with ranked jobs. The ranking is based on the cumulative word frequencies of the search terms within each job, allowing the most relevant jobs to be presented first in the result set.
Spell checking primarily by Gagandeep Singh	**Map<String, Integer> dict :** Initially we set the dictionary with the the default words taken from dictionaryOfWords.txt. Then, whenever we are initializing the trie for the scraped data words we input improve words dictionary by adding the word and its corresponding frequency to the map. This map is used to store the frequency count of each word in the dictionary. It aids in prioritizing suggestions based on the frequency of occurrence. **EditDistanceAlgo :** The edit distance algorithm is employed to calculate the similarity between two words by counting the minimum number of operations (insertions, deletions, or substitutions) required to transform one word into another. **TreeMap<Integer, TreeMap<Integer, TreeSet<String>>> :** This nested TreeMap structure organizes suggested words based on edit distance and word frequency. It allows for efficient retrieval and sorting of suggestions during the spell-checking process. **Summary :** The SpellChecker component provides a robust spell-checking mechanism by leveraging a Trie data structure for efficient word storage and retrieval. The use of edit distance calculations helps identify potential corrections for misspelled words, and the frequency information ensures that suggestions are prioritized based on the popularity of words in the dictionary. The resulting data structures and algorithms collectively contribute to an effective and accurate spell-checking functionality.
Word Completion primarily by Gagandeep Singh	**TrieDS (Trie Data Structure) :** The Trie is employed to efficiently store and retrieve words related to job descriptions. The searchInTrieWithPrefix method is utilized to find words with a given prefix, which aids in providing context-specific suggestions. **SortedArray<WordFrequency> :** This custom data structure is employed to maintain a sorted list of word frequencies. The suggestions are sorted based on the frequency of occurrence in job descriptions, providing relevant and context-aware suggestions. **Summary :** The WordCompletion component leverages a Trie data structure to efficiently fetch word suggestions based on validated search terms. The use of a custom SortedArray ensures that suggestions are sorted by frequency, allowing for the prioritization of more relevant words. This approach provides a responsive and effective word suggestion mechanism, enhancing

	the user experience when interacting with job-related search terms.
Search frequency primarily by Deon Victor Lobo	**LRUCache (Least Recently Used Cache) :** The LRUCache is implemented as a custom class extending LinkedHashMap. It is employed to store search terms and their frequencies, with a constraint on the maximum number of entries. The removeEldestEntry method is overridden to enforce the LRU behavior, ensuring that the least recently used entry is removed when the cache exceeds the specified size. **SortedArray<WordFrequency> :** This custom data structure maintains a sorted list of WordFrequency objects based on their frequency counts. It is used to display the most recent search term frequencies in a sorted order. **Summary :** The SearchFrequency component effectively utilizes an LRUCache to keep track of search term frequencies. By updating the cache with each search term, it ensures that the most recent and frequently used terms are retained. The use of a sorted array further enhances the display by presenting the search terms in descending order of frequency. This approach provides a concise and dynamically updated view of recent search term activities, facilitating insights into user search patterns.
Finding patterns using regular expressions primarily by Rishit Devang Bhojak	**Regular Expressions (Regex):** Regular expressions are powerful tools for pattern matching in strings. During the scraping of jobs salary, each regex pattern is crafted to capture specific salary formats. For example, regexYearlyWithK targets the format "$XK - $YK a year." The matches method checks if the provided job salary string adheres to a particular pattern. **String Manipulation and Conversion:** After matching a pattern, the code uses replaceAll to extract relevant numeric values from the salary string. These values are then processed and converted to integers. For instance, the yearly salary patterns involving "K" (thousands) multiply the numeric values by 1000 to ensure consistent representation. **Summary :** The salary parsing mechanism demonstrates a thoughtful approach to handle diverse formats of job salary strings. By leveraging regular expressions tailored to specific patterns, the code robustly identifies and extracts numeric values. The subsequent conditional logic ensures that the salary information is appropriately converted to a consistent representation, facilitating meaningful analysis and comparison across different job listings. This approach enhances the accuracy of extracting salary data for further processing in the application.
Compare sorting algorithms for Page	**Merge Sort :** I employ the Merge Sort algorithm to perform the initial sorting of the jobs

ranking by Gagandeep Singh, Deon Victor Lobo	retrieved from the inverted index. Merge Sort is a stable, comparison-based sorting algorithm known for its reliability and efficiency **Binary Search :** Binary search during insertion allows you to find the correct position for a new element efficiently in a sorted array. Performs exceptionally well when the data is already partially sorted, as is the case when maintaining a sorted array. The time complexity of binary search is O(log n), which is faster than the average-case time complexity of QuickSort (O(n log n)) or MergeSort (O(n log n)). **QuickSort :** For the final sorting operation, I apply the QuickSort algorithm to the jobs obtained from the inverted index. QuickSort is a widely-used, in-place sorting algorithm known for its average-case time complexity. **Summary :** In this code snippet, I integrate the PageRanking functionality with different sorting algorithms and compare their runtimes. This is valuable for assessing the performance of sorting algorithms in the context of page ranking, where the sorting operation is crucial for presenting relevant and ranked search results.
Recommend top paid jobs based on search results primarily by Deon Victor Lobo, Gagandeep Singh	**Trie Data Structure :** I use a TrieDS (Trie Data Structure) to build an inverted index of job-related terms. Each node in the trie represents a character in a term, and the trie is structured to efficiently store and retrieve information associated with each term. The searchInTrie method efficiently locates the trie node corresponding to a given search term. **SortedArray Data Structure:** I utilize a SortedArray to maintain a sorted collection of jobs based on their calculated cost, which is a combination of the word frequency and the maximum salary associated with each job. The sorted array ensures that jobs are ordered by their cost, facilitating quick identification of top-ranking jobs. **Summary :** This function returns a sorted array of jobs based on salaries in descending order, if there are more than one jobs with the same salary it sorts it by the frequency. The algorithm efficiently combines maximum salary and word frequency to determine the cost of each job, and the use of a sorted array ensures that jobs are maintained in descending order of cost (Basically it's grouped by salary and frequency). This approach enables the rapid identification of top-ranking jobs, providing a streamlined mechanism for users to discover and explore jobs that are likely to be most relevant or desirable based on salary-related criteria.

Code Explanation, Outputs And Screenshots :

1. Web Crawler :

Code Explanation :

Scrapping in Simply hired and GlassDoor. The crawlWebPage method is part of a web scraping application that extracts job-related information from a website based on specified search terms. It utilizes Selenium WebDriver with a browser to navigate through the website, search for jobs corresponding to each term, and scrape relevant details from individual job pages. The method iterates through the provided search terms, accesses the search results page for each term, extracts job links, and continues to subsequent result pages until at least min job links are collected. It then proceeds to visit each job page, extracts job data using the scrapeJobData method, performs data validation, and stores the validated job information in a collection. The scrapJobLinks method uses Jsoup to parse the HTML source of the search results page, extracts job links, and adds them to a queue. The uniqueJobs set ensures that duplicate job entries are not included in the final collection. The extracted job data is then saved and appended to a JSON file using the saveAndAppendToJson method of the ScraperBot class. The WebDriver is closed at the end of the process.

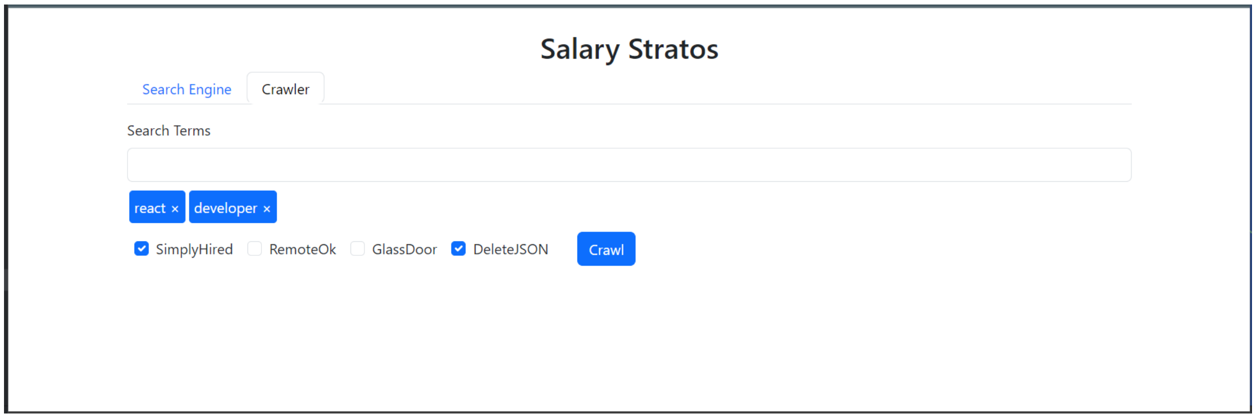
The only difference with scrapping Remote Ok is that the data we need is already existing on the search page and we don't need to open each and every job to get all the data.

Output:

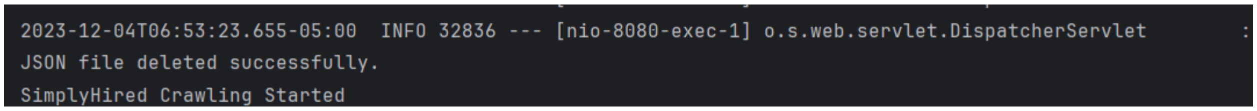
Users can enter the search terms, which website they want to crawl and whether they want to delete the json or append to the same database.json. The driver opens the chrome browser and starts crawling the website with the search terms to get the page source.

Output 1: User wants to delete the database.json and crawl only Simply Hired for search teams "react", "developer"

User clicks on crawl

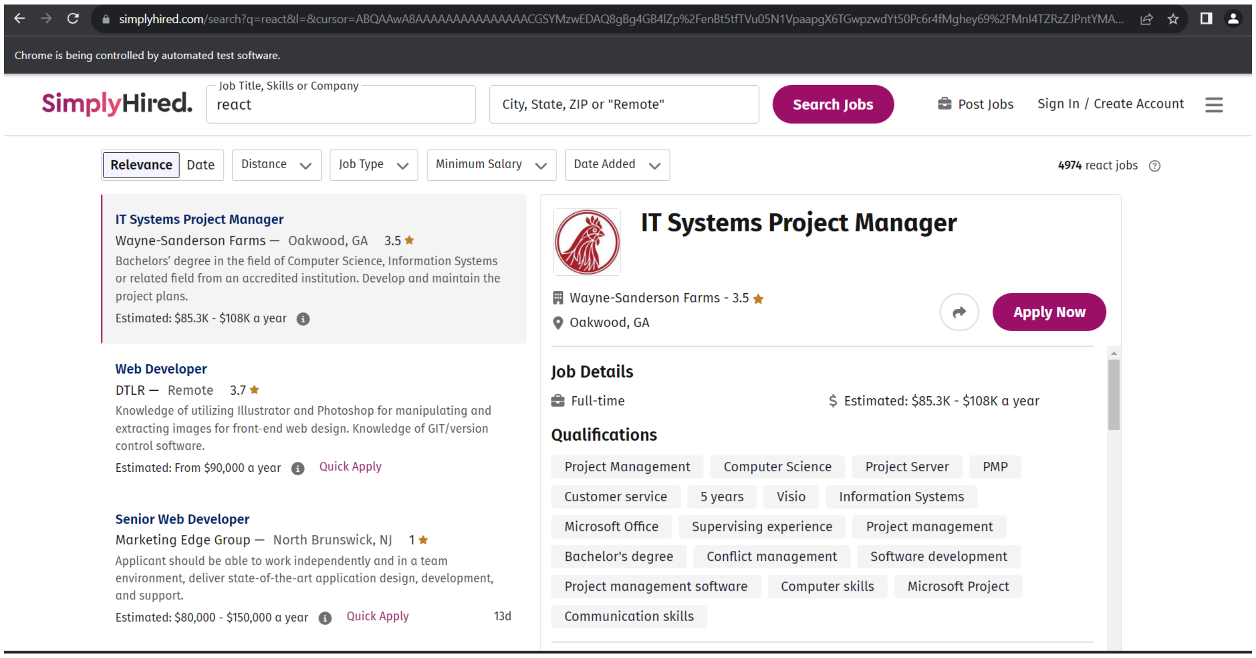


File deleted successfully



```
2023-12-04T06:53:23.655-05:00  INFO 32836 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
JSON file deleted successfully.
SimplyHired Crawling Started
```

It searches for react and gets all the links on that page



simplyhired.com/search?q=react&l=&cursor=ABQAAwA8AAAAAAAAAAAAAAAACGSYMzwEDAQ8gBg4GB4lZp%2FenBt5tfTVu05N1VpaapgX6TGwpzwdYt50Pc6r4fMghey69%2FMnl4TZRzZJPntYMA...

Chrome is being controlled by automated test software.

SimplyHired. Job Title, Skills or Company: react | City, State, ZIP or "Remote" | Search Jobs | Post Jobs | Sign In / Create Account

Relevance | Date | Distance | Job Type | Minimum Salary | Date Added — 4974 react jobs

IT Systems Project Manager
Wayne-Sanderson Farms — Oakwood, GA 3.5 ★
Bachelors' degree in the field of Computer Science, Information Systems or related field from an accredited institution. Develop and maintain the project plans.
Estimated: $85.3K - $108K a year

Web Developer
DTLR — Remote 3.7 ★
Knowledge of utilizing Illustrator and Photoshop for manipulating and extracting images for front-end web design. Knowledge of GIT/version control software.
Estimated: From $90,000 a year — Quick Apply

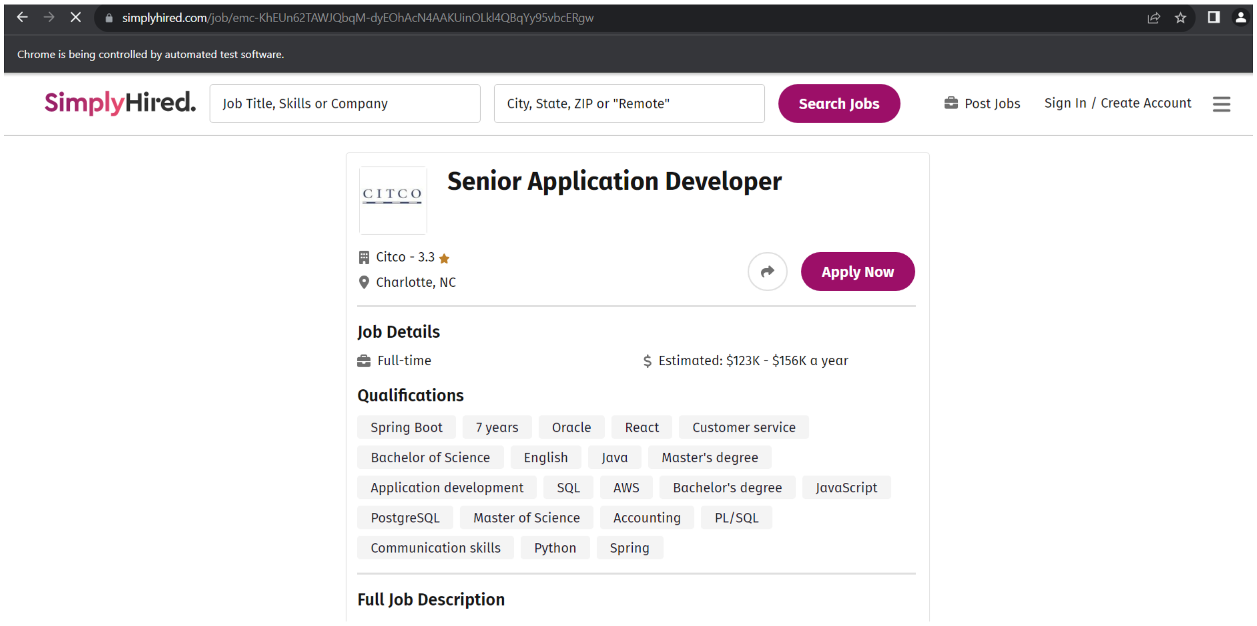
Senior Web Developer
Marketing Edge Group — North Brunswick, NJ 1 ★
Applicant should be able to work independently and in a team environment, deliver state-of-the-art application design, development, and support.
Estimated: $80,000 - $150,000 a year — Quick Apply — 13d

IT Systems Project Manager
Wayne-Sanderson Farms - 3.5 ★
Oakwood, GA
Apply Now

Job Details
Full-time — Estimated: $85.3K - $108K a year

Qualifications
Project Management, Computer Science, Project Server, PMP, Customer service, 5 years, Visio, Information Systems, Microsoft Office, Supervising experience, Project management, Bachelor's degree, Conflict management, Software development, Project management software, Computer skills, Microsoft Project, Communication skills

It opens each link and gets all the job data from that link



simplyhired.com/job/emc-KhEUn62TAWJQbqM-dyEOhAcN4AAKUinOLkl4QBqYy95vbcERgw

Chrome is being controlled by automated test software.

SimplyHired. Job Title, Skills or Company | City, State, ZIP or "Remote" | Search Jobs | Post Jobs | Sign In / Create Account

Senior Application Developer
Citco - 3.3 ★
Charlotte, NC
Apply Now

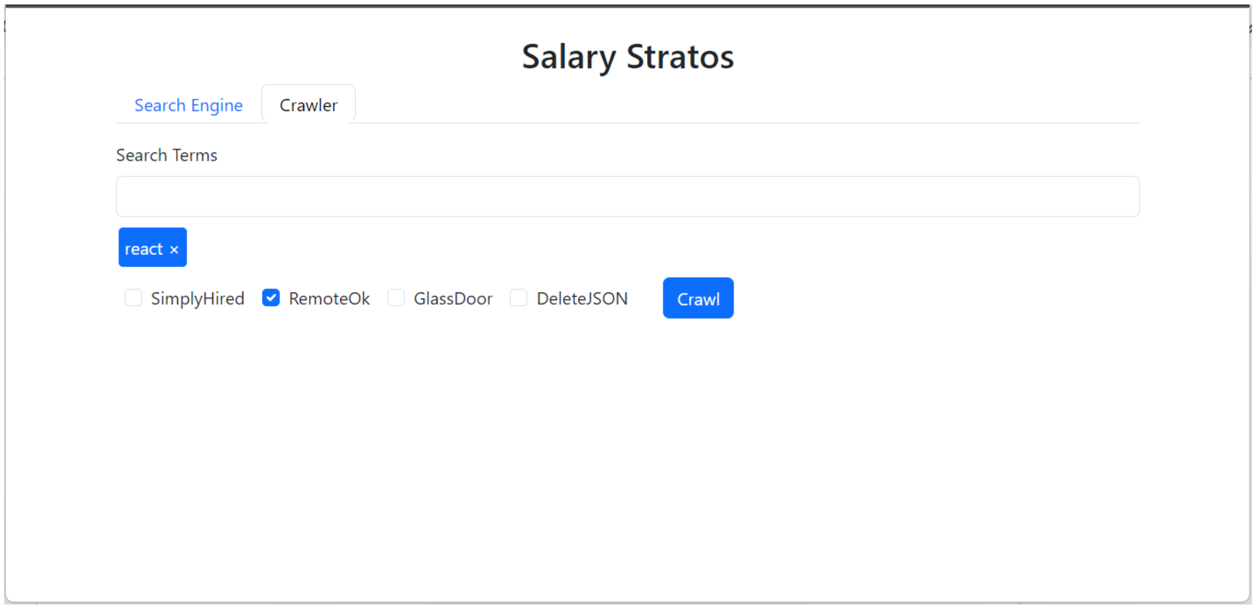
Job Details
Full-time — Estimated: $123K - $156K a year

Qualifications
Spring Boot, 7 years, Oracle, React, Customer service, Bachelor of Science, English, Java, Master's degree, Application development, SQL, AWS, Bachelor's degree, JavaScript, PostgreSQL, Master of Science, Accounting, PL/SQL, Communication skills, Python, Spring

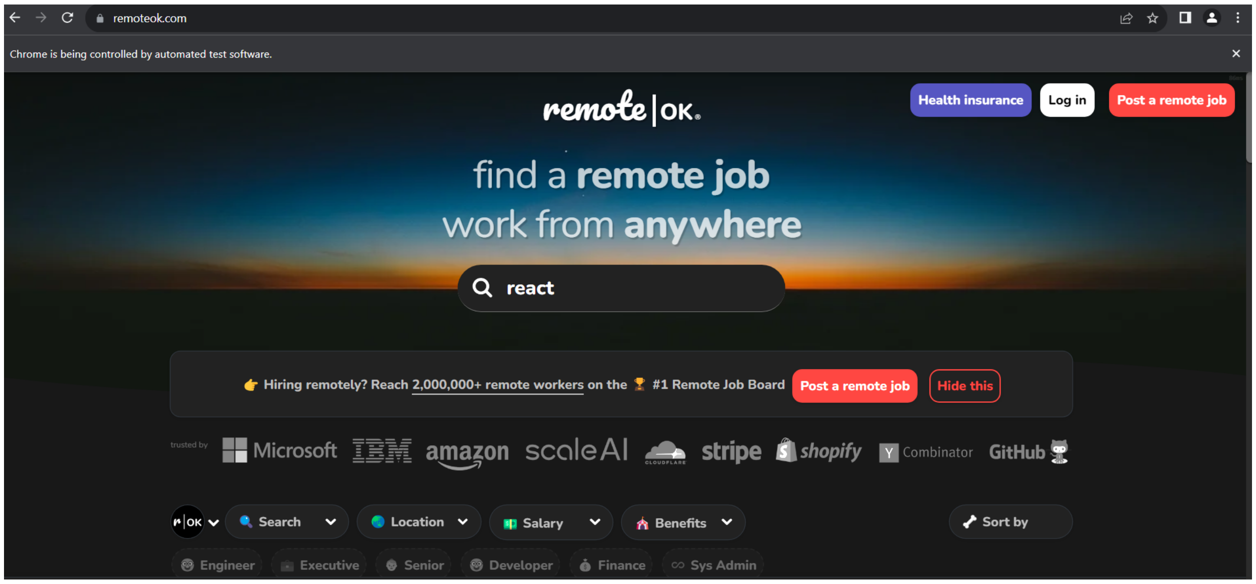
Full Job Description

<u>Output 2: User wants to crawl only Remote Ok for search teams "react", "developer" and wants to append to the database.json</u>

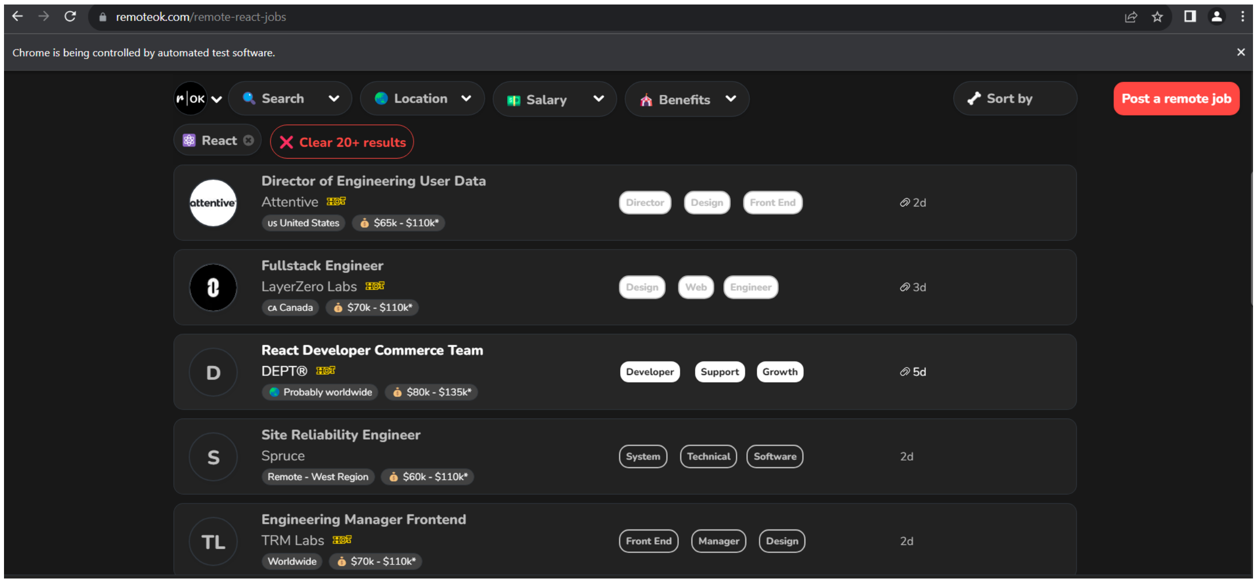
User clicks on crawl



Search for “react”



It gets all the jobs on this page


remoteok.com/remote-react-jobs

Chrome is being controlled by automated test software.

Search | Location | Salary | Benefits | Sort by | Post a remote job

React | Clear 20+ results

Director of Engineering User Data
Attentive
us United States | $65k - $110k*
Director | Design | Front End | 2d

Fullstack Engineer
LayerZero Labs
ca Canada | $70k - $110k*
Design | Web | Engineer | 3d

React Developer Commerce Team
DEPT®
Probably worldwide | $80k - $135k*
Developer | Support | Growth | 5d

Site Reliability Engineer
Spruce
Remote - West Region | $60k - $110k*
System | Technical | Software | 2d

Engineering Manager Frontend
TRM Labs
Worldwide | $70k - $110k*
Front End | Manager | Design | 2d

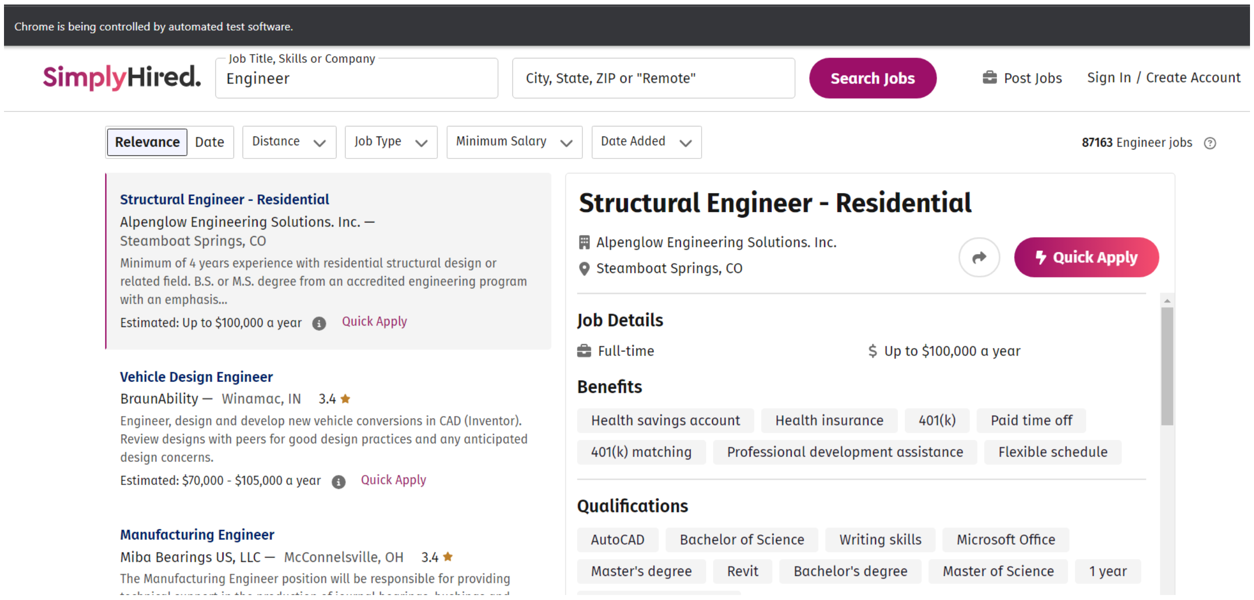
<u>Output 3 : If file does not exist when starting the application it will crawl the data for default search terms and create the database.json file.</u>

```
String[] searchTermsList = new String[]{
          "Engineer", "Exec", "Senior", "Developer", "Finance", "Sys
Admin", "JavaScript", "Backend", "Golang", "Cloud", "Front End"
};
```

File was not found

```
2023-12-04T07:04:59.747-05:00  INFO 41512 --- [  restartedMain] c.a.S.
File does not exist re-crawl data for default search terms
Bot loaded for first time
```

Re-crawl data for the default search terms


Chrome is being controlled by automated test software.

SimplyHired. Job Title, Skills or Company: Engineer | City, State, ZIP or "Remote" | Search Jobs | Post Jobs | Sign In / Create Account

Relevance | Date | Distance | Job Type | Minimum Salary | Date Added | 87163 Engineer jobs

Structural Engineer - Residential
Alpenglow Engineering Solutions. Inc. —
Steamboat Springs, CO
Minimum of 4 years experience with residential structural design or related field. B.S. or M.S. degree from an accredited engineering program with an emphasis...
Estimated: Up to $100,000 a year | Quick Apply

Vehicle Design Engineer
BraunAbility — Winamac, IN 3.4 ★
Engineer, design and develop new vehicle conversions in CAD (Inventor). Review designs with peers for good design practices and any anticipated design concerns.
Estimated: $70,000 - $105,000 a year | Quick Apply

Manufacturing Engineer
Miba Bearings US, LLC — McConnelsville, OH 3.4 ★
The Manufacturing Engineer position will be responsible for providing

Structural Engineer - Residential
Alpenglow Engineering Solutions. Inc.
Steamboat Springs, CO
Quick Apply

Job Details
Full-time | Up to $100,000 a year

Benefits
Health savings account | Health insurance | 401(k) | Paid time off | 401(k) matching | Professional development assistance | Flexible schedule

Qualifications
AutoCAD | Bachelor of Science | Writing skills | Microsoft Office | Master's degree | Revit | Bachelor's degree | Master of Science | 1 year

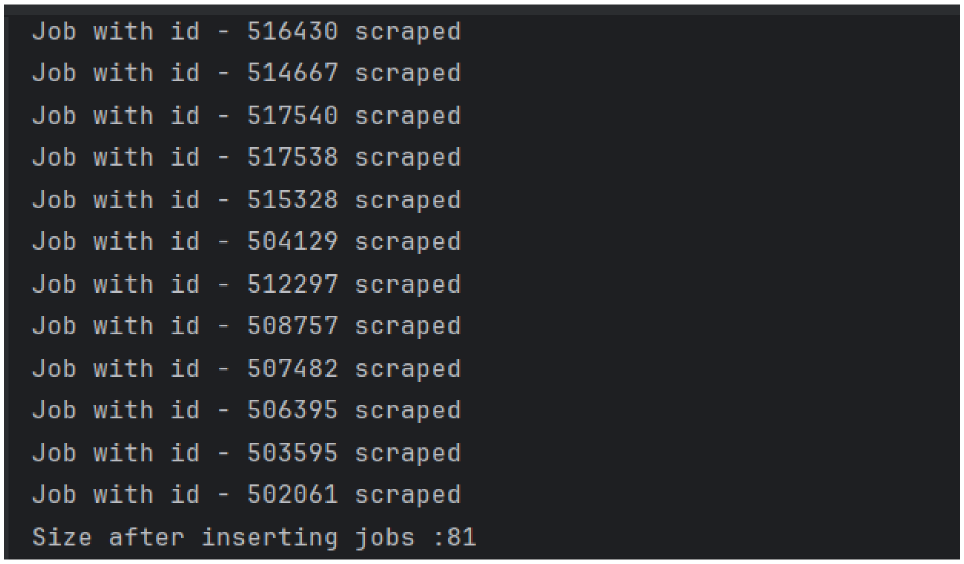
2. HTML Parser :

Code Explanation :

The scrapeJobData method is part of a web scraping application designed to extract job-related information from a webpage. It uses the Jsoup library to parse the HTML source of the page and retrieve details such as job ID, title, company name, location, website name, salary, and job description. The method applies regular expressions to extract numerical salary information and converts it into minimum and maximum salary values. Various regular expression patterns are used to handle different salary formats, including daily, weekly, monthly, yearly, and hourly rates. The extracted data is then used to create a Job object, which encapsulates the job-related information, and this object is returned by the method. Overall, the method provides a structured way to extract and organize job data from the HTML source of a job posting webpage.

Output 1 :

The jobs with the below jobIds got scrapped



```
Job with id - 516430 scraped
Job with id - 514667 scraped
Job with id - 517540 scraped
Job with id - 517538 scraped
Job with id - 515328 scraped
Job with id - 504129 scraped
Job with id - 512297 scraped
Job with id - 508757 scraped
Job with id - 507482 scraped
Job with id - 506395 scraped
Job with id - 503595 scraped
Job with id - 502061 scraped
Size after inserting jobs :81
```

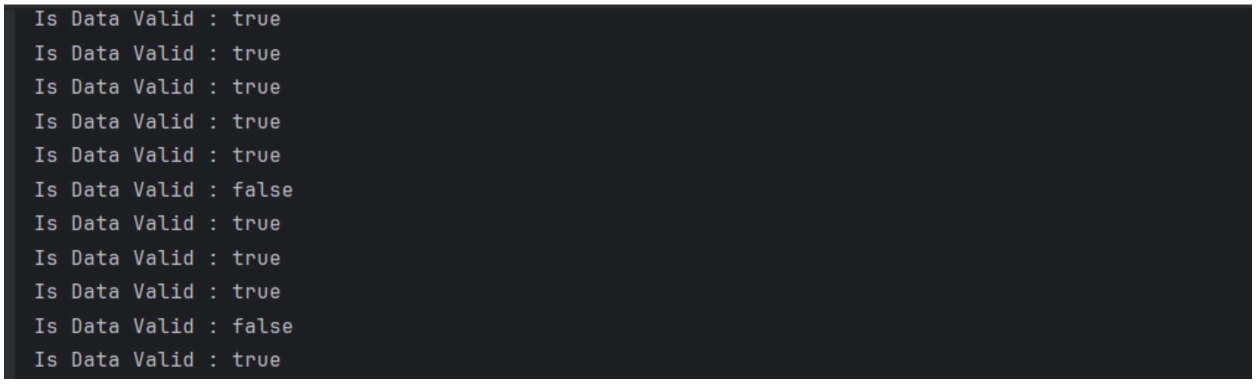
3. Data validation using regular expressions :

Code Explanation :

The code implements data validation for job-related information extracted during web scraping. It defines a set of regular expressions for validating different fields, such as job title, company name, website link, salary, location, and job description. The validateField method is a generic function that checks whether a given field matches the specified regular expression. The validateDataForOneObject method utilizes this generic validator for each field of a Job object, setting validation results for individual fields and an overall validation status. If any field fails validation, the allValid flag is set to false. During job data processing, only objects with valid information, as determined by the data validation process, are added to the jobsCollection. This ensures that only well-formed and compliant job data is included in the final collection, enhancing the overall data quality.

<u>Output 1 :</u>

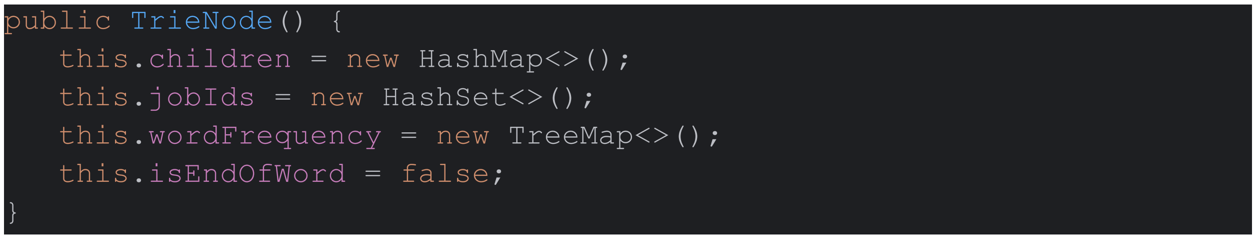
Validate data and only add to the database.json if the data is valid



```
Is Data Valid : true
Is Data Valid : true
Is Data Valid : true
Is Data Valid : true
Is Data Valid : true
Is Data Valid : false
Is Data Valid : true
Is Data Valid : true
Is Data Valid : true
Is Data Valid : false
Is Data Valid : true
```

4. Inverted Indexing and Frequency count :

<u>Code Explanation :</u>



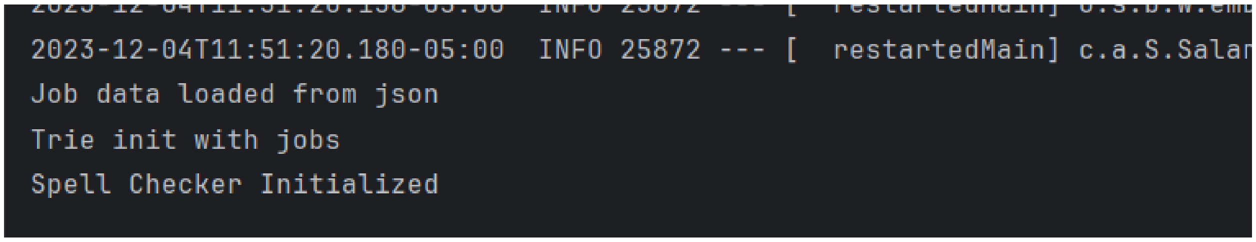
```java
public TrieNode() {
   this.children = new HashMap<>();
   this.jobIds = new HashSet<>();
   this.wordFrequency = new TreeMap<>();
   this.isEndOfWord = false;
}
```

The code implements inverted indexing for efficient search operations in a large dataset of job descriptions. In the initialization of the trie (TrieDS), each job's description is processed to extract relevant tokens, excluding common English stop words. These processed tokens are then inserted into the trie data structure using the insertIntoTrie method. The trie enables quick retrieval of job IDs associated with specific words, forming an inverted index. This facilitates the search process by providing immediate access to job entries containing the queried words. The TrieNode class represents each node in the trie and includes a set of job IDs, establishing a direct link between words and their occurrences in job descriptions.

For frequency counting, the code utilizes the TrieNode's wordFrequency TreeMap within the insertIntoTrie method. This TreeMap stores the frequencies of each word within the job descriptions, organized by job ID. As words are inserted into the trie, their frequencies are updated in the corresponding TrieNode. This frequency count mechanism allows for efficient analysis of word occurrences across the dataset. During subsequent searches or data analysis, the Trie structure provides immediate access to both the job IDs containing specific words and the frequencies of those words within each job description. Overall, the combination of inverted indexing and frequency counting in the trie enhances the speed and efficiency of search operations in large-scale job datasets.

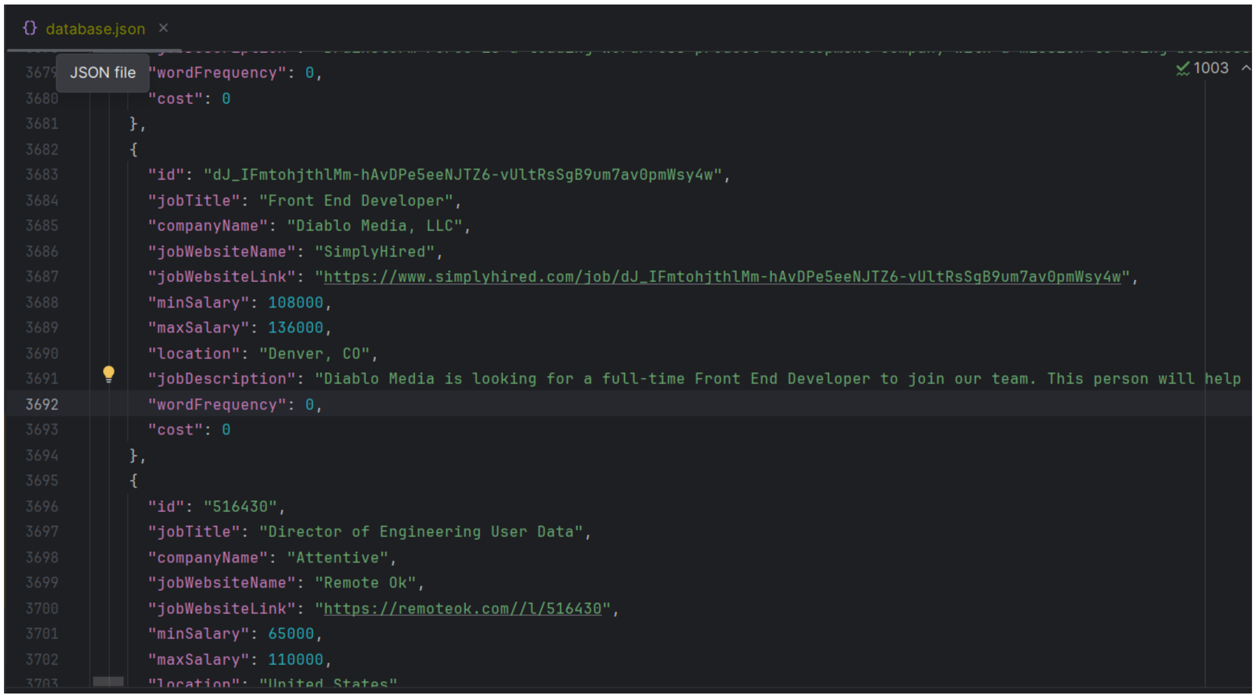
Output 1 :

The trie gets initialized with database.json every time the application starts



```
2023-12-04T11:51:20.180-05:00  INFO 25872 --- [  restartedMain] c.a.S.Salar
Job data loaded from json
Trie init with jobs
Spell Checker Initialized
```

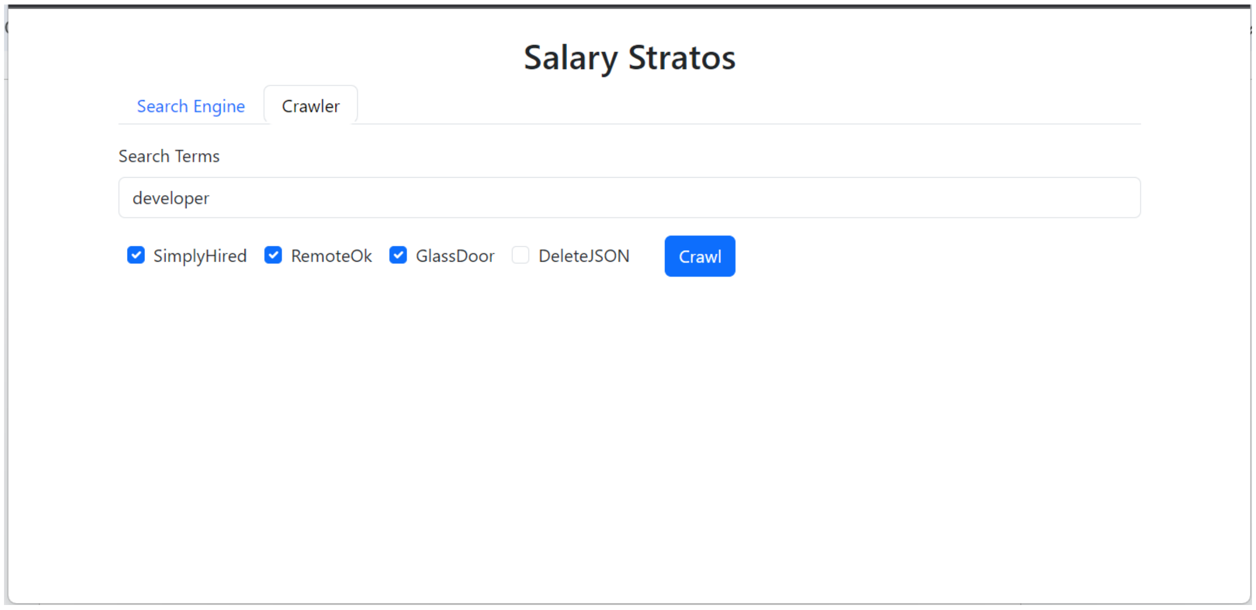
The words from database.json is used to initialize the trie



```
      "wordFrequency": 0,
      "cost": 0
    },
    {
      "id": "dJ_IFmtohjthlMm-hAvDPe5eeNJTZ6-vUltRsSgB9um7av0pmWsy4w",
      "jobTitle": "Front End Developer",
      "companyName": "Diablo Media, LLC",
      "jobWebsiteName": "SimplyHired",
      "jobWebsiteLink": "https://www.simplyhired.com/job/dJ_IFmtohjthlMm-hAvDPe5eeNJTZ6-vUltRsSgB9um7av0pmWsy4w",
      "minSalary": 108000,
      "maxSalary": 136000,
      "location": "Denver, CO",
      "jobDescription": "Diablo Media is looking for a full-time Front End Developer to join our team. This person will help
      "wordFrequency": 0,
      "cost": 0
    },
    {
      "id": "516430",
      "jobTitle": "Director of Engineering User Data",
      "companyName": "Attentive",
      "jobWebsiteName": "Remote Ok",
      "jobWebsiteLink": "https://remoteok.com//l/516430",
      "minSalary": 65000,
      "maxSalary": 110000,
      "location": "United States",
```

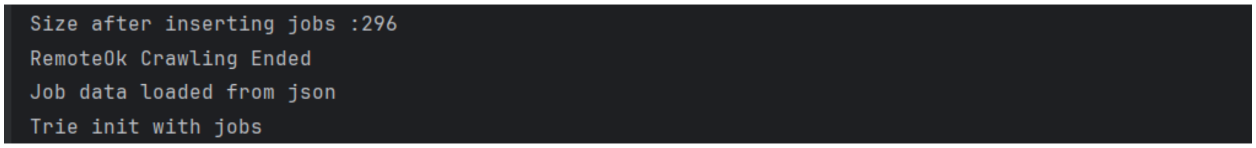
Output 2 :

The trie gets reinitialized whenever we scrape the data from the ui



Search Terms: developer

[✓ SimplyHired] [✓ RemoteOk] [✓ GlassDoor] DeleteJSON



```
Size after inserting jobs :296
RemoteOk Crawling Ended
Job data loaded from json
Trie init with jobs
```

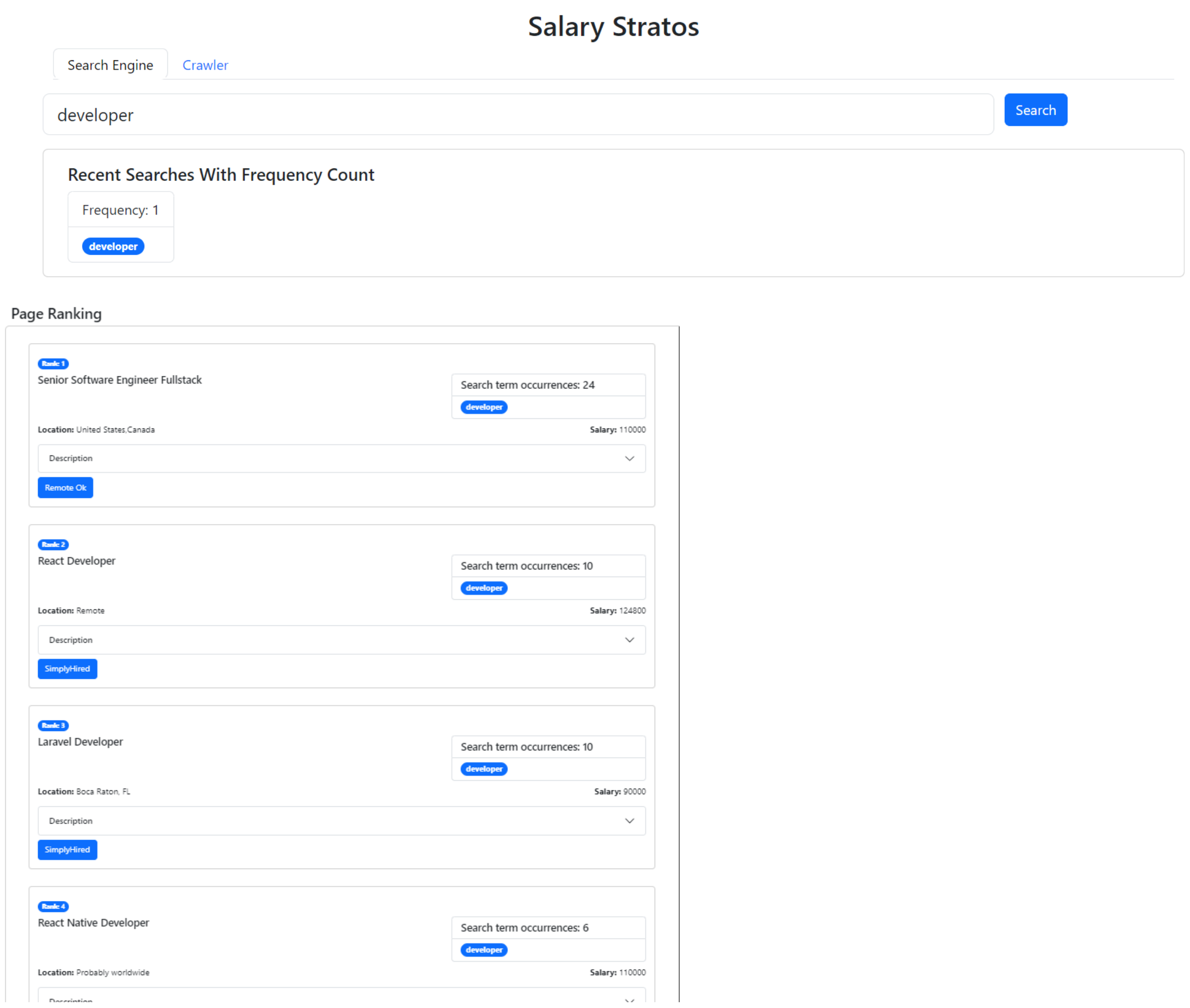
5. Page Ranking :

Code Explanation :

The PageRanking class implements page ranking by searching inverted indexed data stored in a Trie data structure. It takes an array of search terms, retrieves their frequencies from the Trie, and ranks jobs based on the cumulative word frequencies of the search terms within each job's description. For each term, it iterates over the corresponding Trie node to obtain job IDs and their word frequencies. Then, for each job ID, it retrieves the corresponding job details from the stored job data, updating the job's accumulated word frequency and word field. The jobs are then inserted into a SortedArray, sorted based on their word frequencies. If a job already has a non-zero frequency, it is temporarily removed from the sorted structure for updating before being reinserted. This is done because if we don't remove the word it will create duplication of jobs in the response array if we search for multiple search terms. The resulting SortedArray contains jobs ranked according to the cumulative frequency of the search terms within their descriptions, providing a page ranking reflecting the relevance of jobs to the search criteria.

Output 1 :

Result on the ui for one search term


Salary Stratos

Recent Searches With Frequency Count

Frequency: 1
developer

Page Ranking

Rank: 1
Senior Software Engineer Fullstack
Search term occurrences: 24
developer
Location: United States,Canada
Salary: 110000

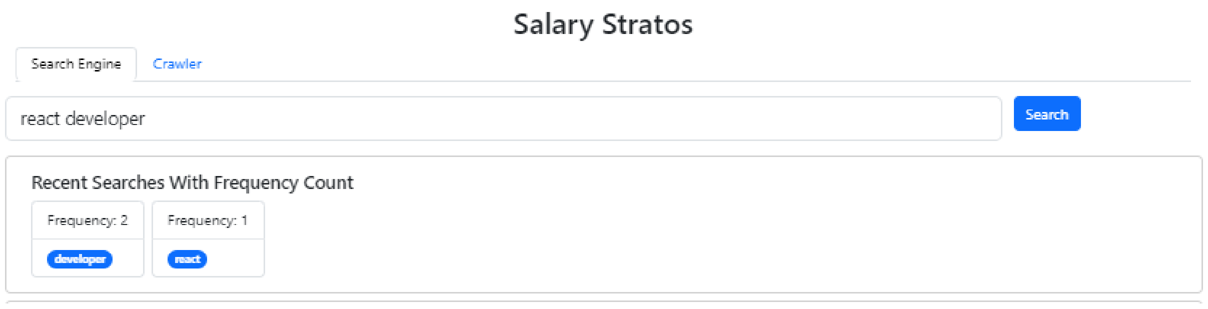
Rank: 2
React Developer
Search term occurrences: 10
developer
Location: Remote
Salary: 124800

Rank: 3
Laravel Developer
Search term occurrences: 10
developer
Location: Boca Raton, FL
Salary: 90000

Rank: 4
React Native Developer
Search term occurrences: 6
developer
Location: Probably worldwide
Salary: 110000

<u>Output 2 :</u>

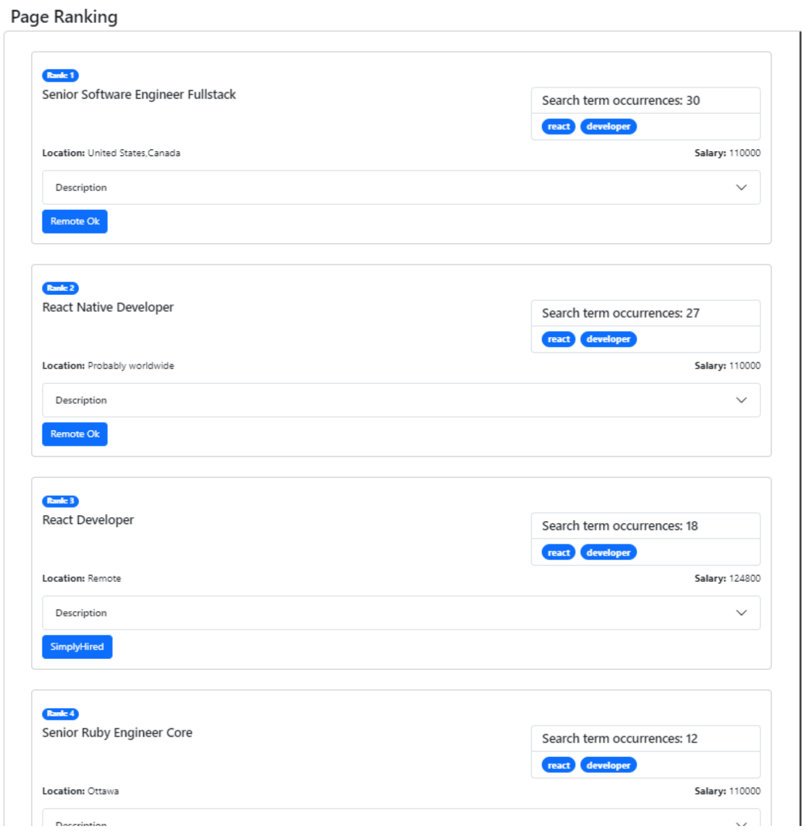
Result on the ui for two search terms


Salary Stratos

Recent Searches With Frequency Count

Frequency: 2
developer

Frequency: 1
react


Page Ranking

Rank: 1
Senior Software Engineer Fullstack
Search term occurrences: 30
react developer
Location: United States,Canada
Salary: 110000
Description
Remote Ok

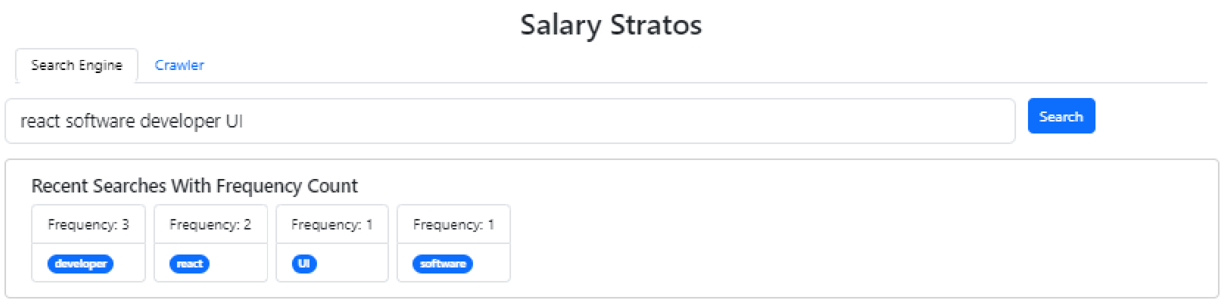
Rank: 2
React Native Developer
Search term occurrences: 27
react developer
Location: Probably worldwide
Salary: 110000
Description
Remote Ok

Rank: 3
React Developer
Search term occurrences: 18
react developer
Location: Remote
Salary: 124800
Description
SimplyHired

Rank: 4
Senior Ruby Engineer Core
Search term occurrences: 12
react developer
Location: Ottawa
Salary: 110000
Description

<u>Output 3 :</u>

Result on the ui for multiple search terms


Salary Stratos

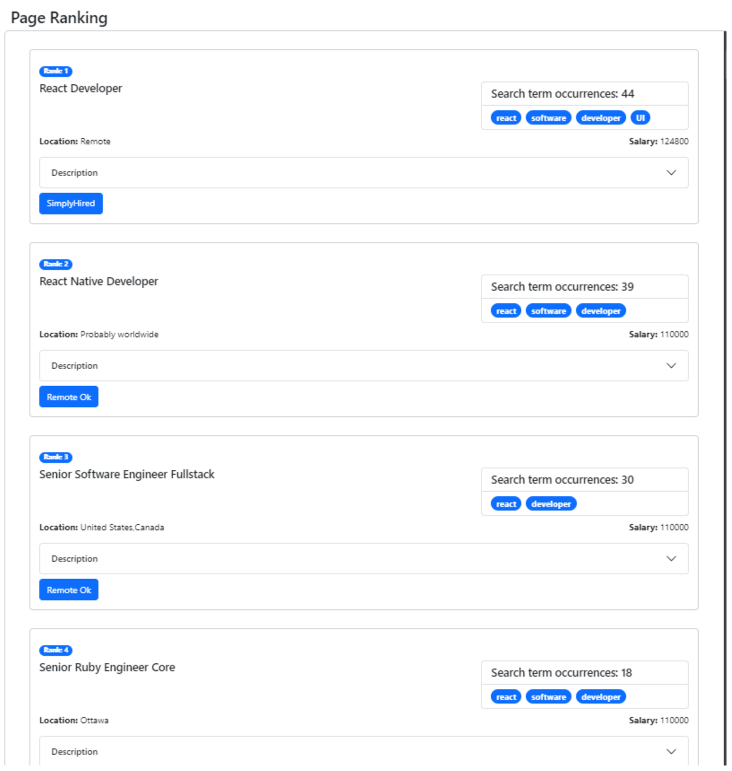
Search Engine Crawler

react software developer UI

Search

Recent Searches With Frequency Count

Frequency: 3	Frequency: 2	Frequency: 1	Frequency: 1
developer	react	UI	software


Page Ranking

Rank: 1
React Developer
Search term occurrences: 44
react software developer UI
Location: Remote
Salary: 124800
Description
SimplyHired

Rank: 2
React Native Developer
Search term occurrences: 39
react software developer
Location: Probably worldwide
Salary: 110000
Description
Remote Ok

Rank: 3
Senior Software Engineer Fullstack
Search term occurrences: 30
react developer
Location: United States,Canada
Salary: 110000
Description
Remote Ok

Rank: 4
Senior Ruby Engineer Core
Search term occurrences: 18
react software developer
Location: Ottawa
Salary: 110000
Description

6. Spell checking :

Code Explanation :

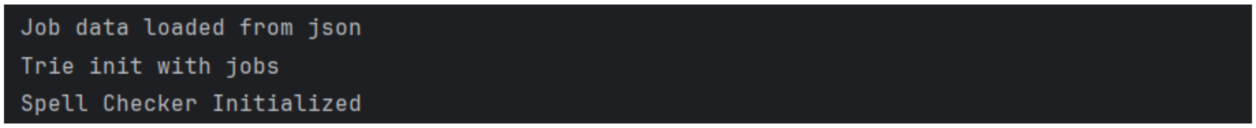
The SpellChecker class is designed to perform spell checking by maintaining a dictionary of valid words and employing the EditDistanceAlgo for suggesting similar words in case of misspelled input. The initialization process, handled by the initializeSpellChecker method, reads words from an external dictionary file (dictionaryOfWords.txt) and populates both a Trie data structure (TrieDS) and a frequency map (dict). The dict also contains all the words and the frequencies of the word from the database.json. The TrieDS efficiently stores words for quick retrieval and supports autocomplete features.

In the suggestSimilarWord method, given an input word, the code first checks if it's empty or belongs to a predefined list of invalid words. If the word is valid, it searches for an exact match in the Trie. If no match is found, it utilizes the EditDistanceAlgo to calculate the edit distance between the input word and all words in the dictionary. The results are organized in a nested TreeMap structure, where the outer TreeMap sorts suggestions based on edit distance, and the inner TreeMap further organizes them by word frequency. This prioritization ensures that suggestions are not only linguistically similar but also ranked by their prevalence in the dictionary, contributing to more accurate and contextually relevant spell-checking suggestions.

In summary, the spell-checking functionality leverages a Trie data structure, an external dictionary, and the EditDistanceAlgo to offer efficient and accurate suggestions for potential corrections based on word similarity and frequency. The data structures and algorithms employed contribute to a robust spell-checking mechanism within the provided code.

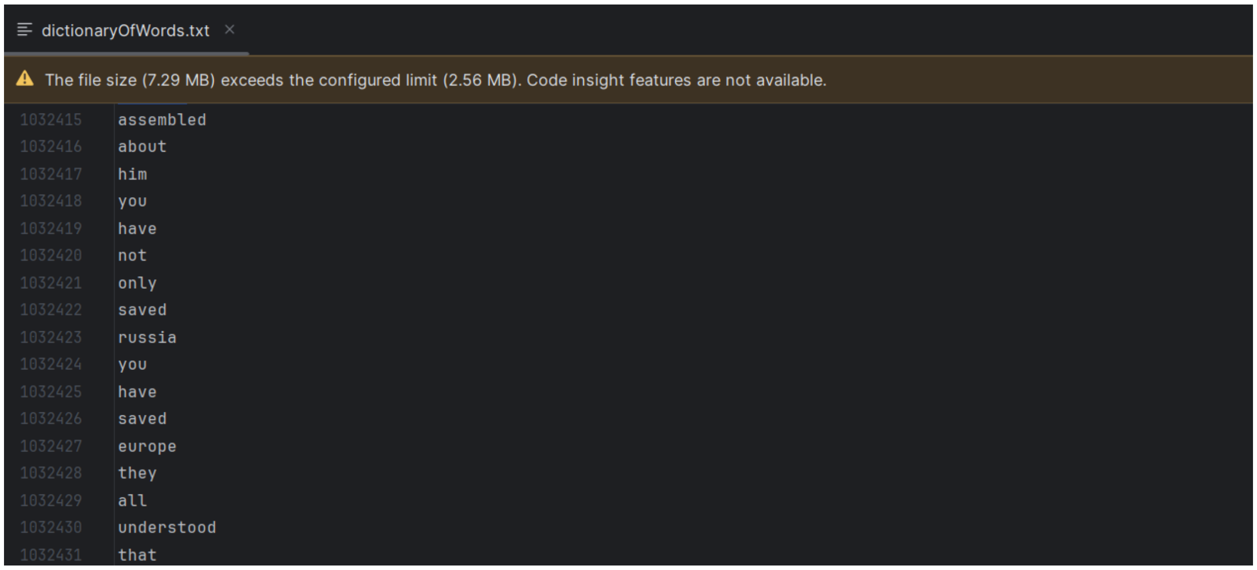
<u>Output 1 :</u>

When the program starts the spell checker dictionary is initialized.



```
Job data loaded from json
Trie init with jobs
Spell Checker Initialized
```

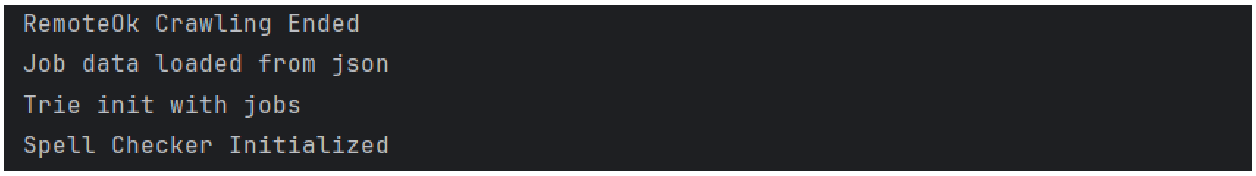
Data gets initialized from the dictionaryOfWords.txt



```
dictionaryOfWords.txt ×
⚠ The file size (7.29 MB) exceeds the configured limit (2.56 MB). Code insight features are not available.
assembled
about
him
you
have
not
only
saved
russia
you
have
saved
europe
they
all
understood
that
```

<u>Output 2 :</u>

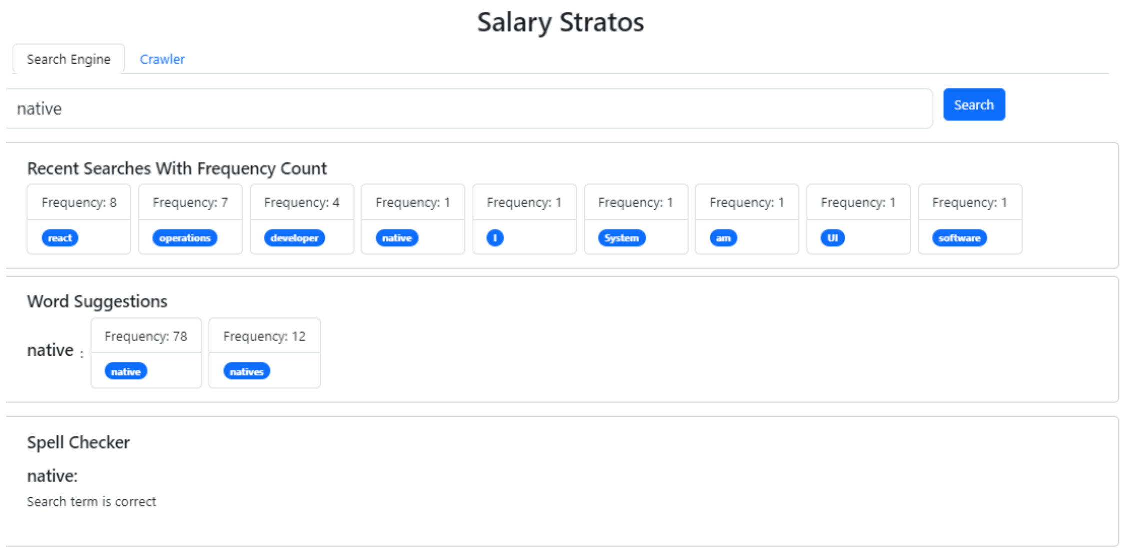
When the Scrapper is called, the spell checker dictionary is initialized.



```
RemoteOk Crawling Ended
Job data loaded from json
Trie init with jobs
Spell Checker Initialized
```

<u>Output 3 :</u>

Spell checking for one search term when the user hits enter. The search term native is correct.


Salary Stratos

Search Engine Crawler

native

Search

Recent Searches With Frequency Count

Frequency: 8	Frequency: 7	Frequency: 4	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	operations	developer	native	I	System	am	UI	software

Word Suggestions

native :

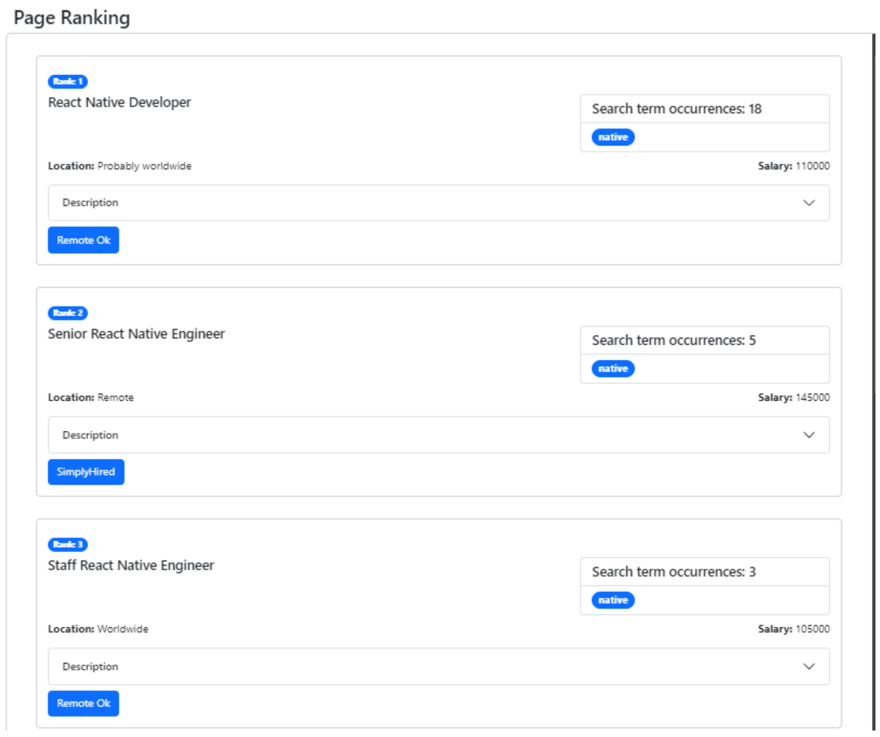
Frequency: 78	Frequency: 12
native	natives

Spell Checker

native:

Search term is correct

It will search for native


Page Ranking

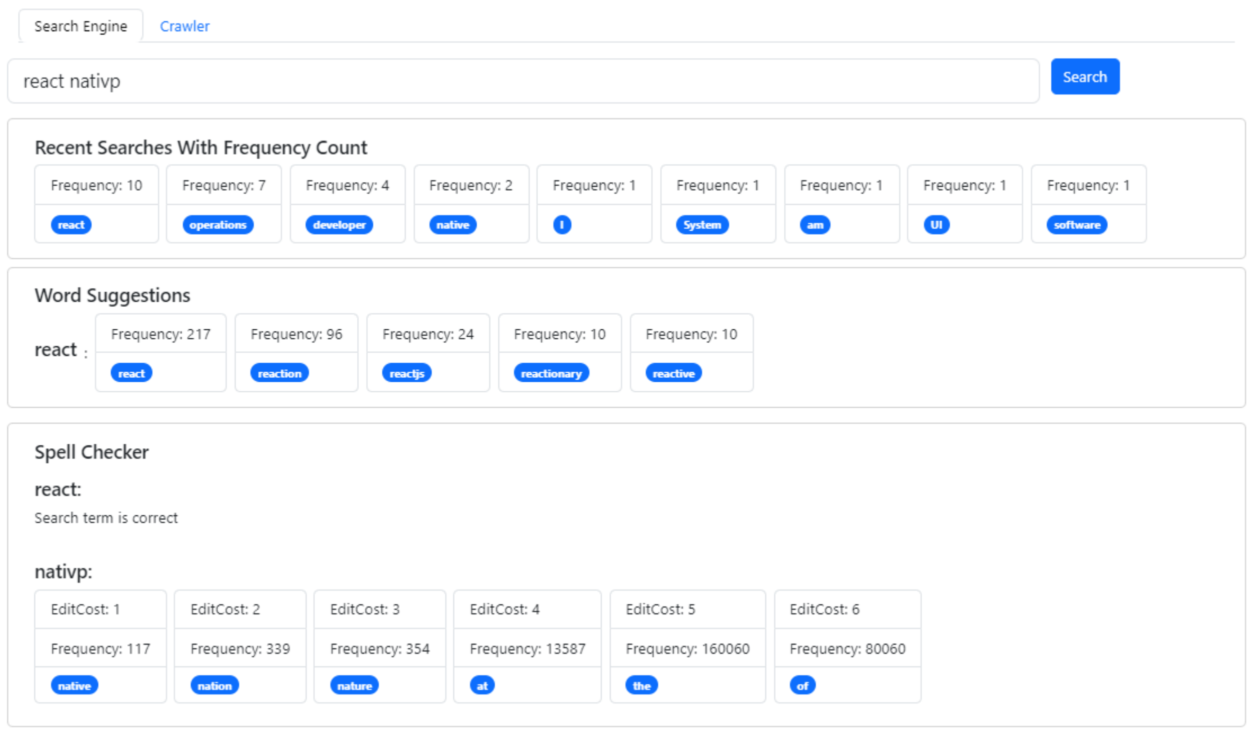
Rank: 1
React Native Developer
Search term occurrences: 18
native
Location: Probably worldwide
Salary: 110000
Description
Remote Ok

Rank: 2
Senior React Native Engineer
Search term occurrences: 5
native
Location: Remote
Salary: 145000
Description
SimplyHired

Rank: 3
Staff React Native Engineer
Search term occurrences: 3
native
Location: Worldwide
Salary: 105000
Description
Remote Ok

Output 4 :

Spell checking for two search terms when the user hits enter. The second search term nativp is not correct but the closest word to this is native so it suggested that based on edit distance


Search Engine | Crawler

react nativp

Search

Recent Searches With Frequency Count

Frequency: 10	Frequency: 7	Frequency: 4	Frequency: 2	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	operations	developer	native	I	System	am	UI	software

Word Suggestions

react :

Frequency: 217	Frequency: 96	Frequency: 24	Frequency: 10	Frequency: 10
react	reaction	reactjs	reactionary	reactive

Spell Checker

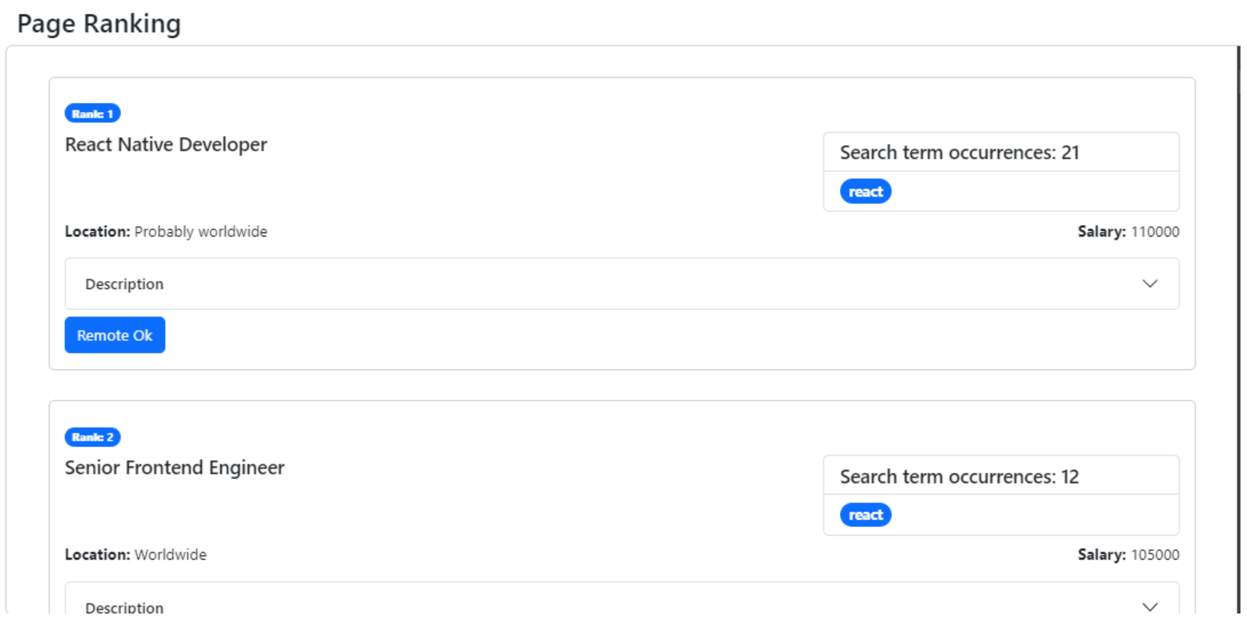
react:

Search term is correct

nativp:

EditCost: 1	EditCost: 2	EditCost: 3	EditCost: 4	EditCost: 5	EditCost: 6
Frequency: 117	Frequency: 339	Frequency: 354	Frequency: 13587	Frequency: 160060	Frequency: 80060
native	nation	nature	at	the	of

It will only search for react in this case as the second term is wrong.


Page Ranking

Rank: 1

React Native Developer

Search term occurrences: 21

react

Location: Probably worldwide

Salary: 110000

Description

Remote Ok

Rank: 2

Senior Frontend Engineer

Search term occurrences: 12

react

Location: Worldwide

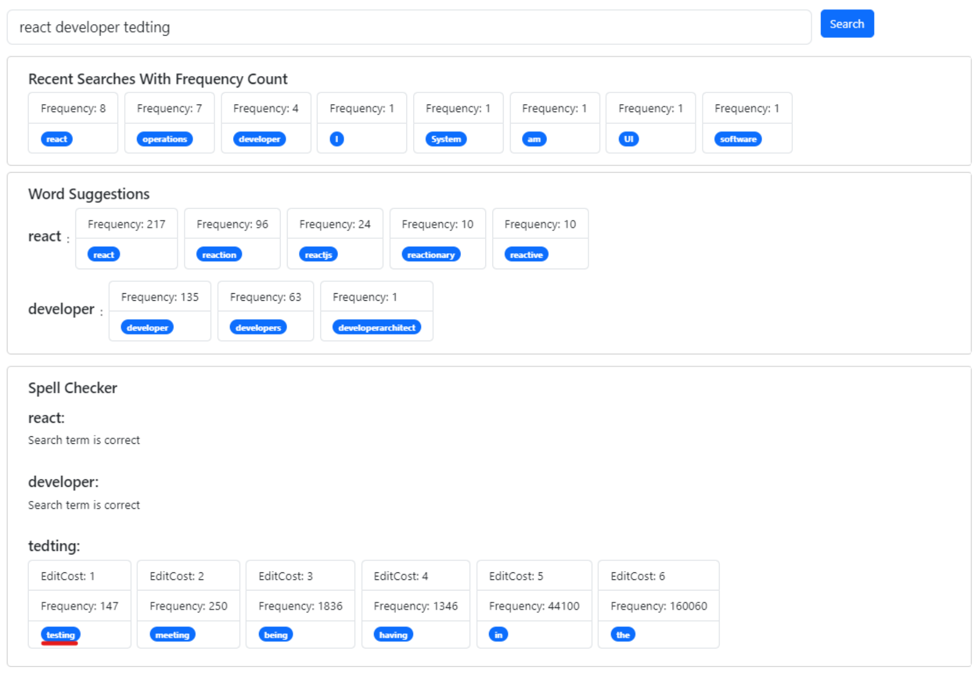
Salary: 105000

Description

<u>Output 5 :</u>

Spell checking for multiple search terms when the user hits enter.

If we observe below we entered 3 words “react developer tedting”. We can see that react and operations are correct words but tesdting is a wrong word. It also gave the closest word to tesdtion i.e. testing based on Edit Distance.


react developer tedting

Search

Recent Searches With Frequency Count

Frequency: 8	Frequency: 7	Frequency: 4	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	operations	developer	I	System	am	UI	software

Word Suggestions

react :

Frequency: 217	Frequency: 96	Frequency: 24	Frequency: 10	Frequency: 10
react	reaction	reactjs	reactionary	reactive

developer :

Frequency: 135	Frequency: 63	Frequency: 1
developer	developers	developerarchitect

Spell Checker

react:

Search term is correct

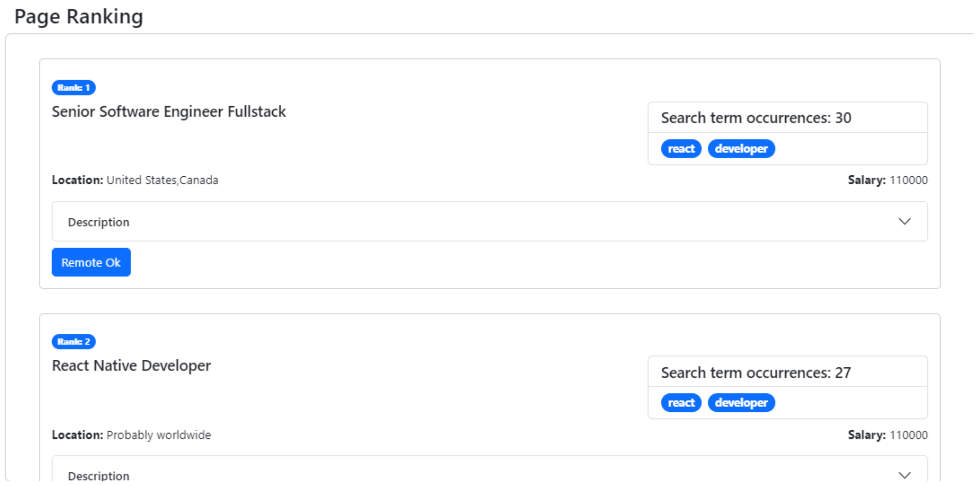
developer:

Search term is correct

tedting:

EditCost: 1	EditCost: 2	EditCost: 3	EditCost: 4	EditCost: 5	EditCost: 6
Frequency: 147	Frequency: 250	Frequency: 1836	Frequency: 1346	Frequency: 44100	Frequency: 160060
testing	meeting	being	having	in	the

The search page ranking is done only for two words in this case "react developer"


Page Ranking

Rank: 1

Senior Software Engineer Fullstack

Search term occurrences: 30

react developer

Location: United States,Canada

Salary: 110000

Description

Remote Ok

Rank: 2

React Native Developer

Search term occurrences: 27

react developer

Location: Probably worldwide

Salary: 110000

Description

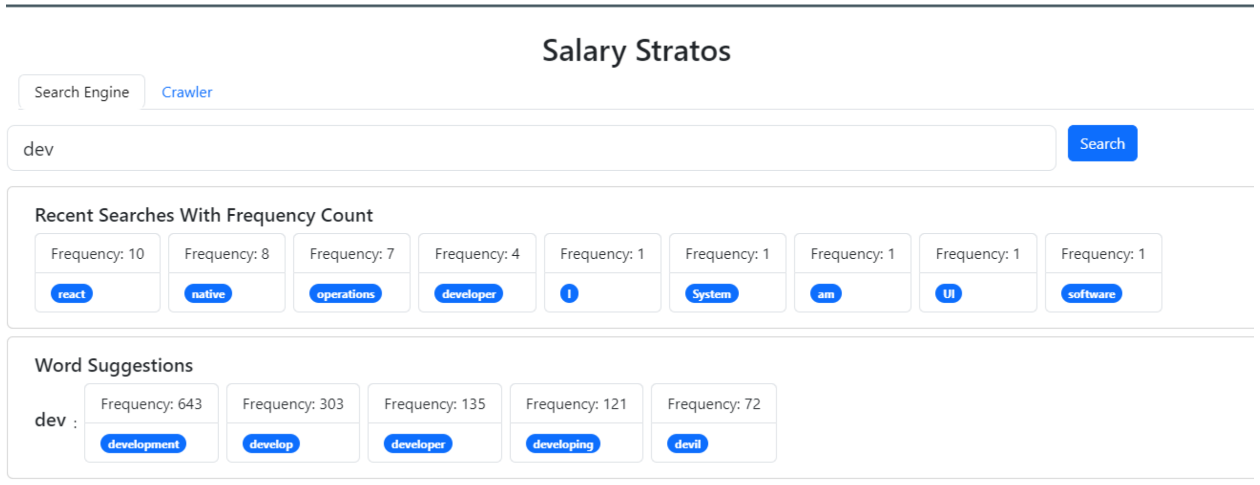
7. Word Completion :

Code Explanation :

The WordCompletion class provides functionality to generate word suggestions based on validated search terms using a Trie data structure. The getWordSuggestions method takes a list of validated search terms, a Trie structure (JobDataTrie), and the desired count of suggestions. It iterates through each search term, retrieves word suggestions from the Trie using the searchInTrieWithPrefix method, and constructs a response containing the original search term and a SortedArray of suggested words along with their frequencies. The searchInTrieWithPrefix method traverses the Trie to find words with the specified prefix, and the collectWords recursive function gathers word frequencies from the Trie nodes, constructing a SortedArray of WordFrequency objects sorted by frequency. The resulting WordSuggestionResponse encapsulates the original search terms, whether the response is valid, and a list of suggested words with their frequencies.

Output 1 :

Word Suggestion for one search term when the user enters a partial word. It's not necessary click on enter to see the word suggestions



Salary Stratos

Search Engine | Crawler

dev

Search

Recent Searches With Frequency Count

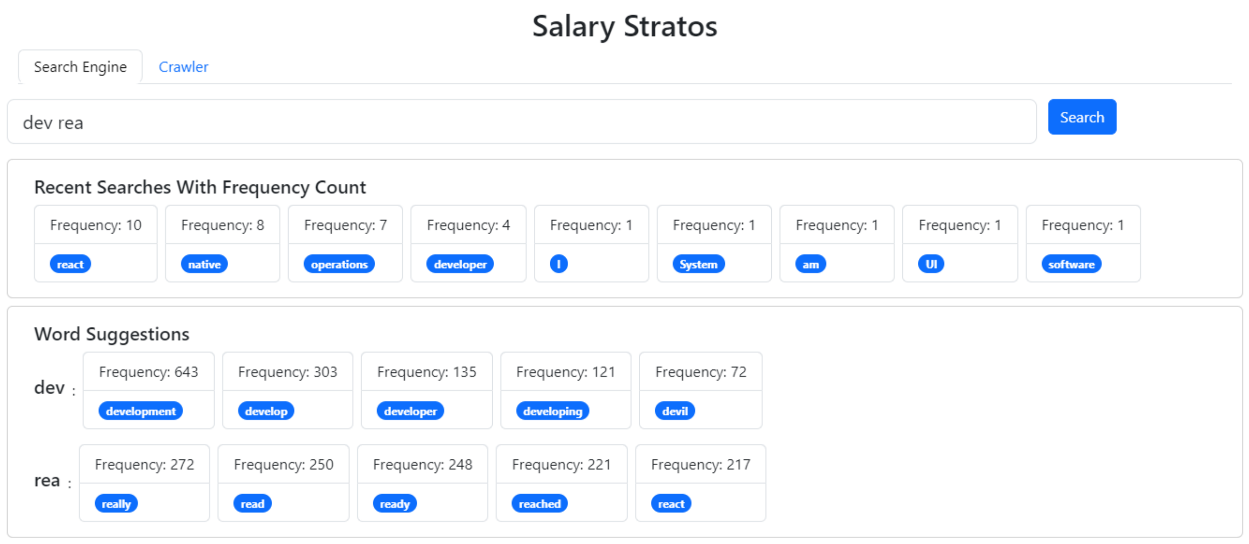
Frequency: 10	Frequency: 8	Frequency: 7	Frequency: 4	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	native	operations	developer	I	System	am	UI	software

Word Suggestions

	Frequency: 643	Frequency: 303	Frequency: 135	Frequency: 121	Frequency: 72
dev :	development	develop	developer	developing	devil

Output 2 :

Word Suggestion for two search terms when the user enters a partial word



Salary Stratos

Search Engine | Crawler

dev rea

Search

Recent Searches With Frequency Count

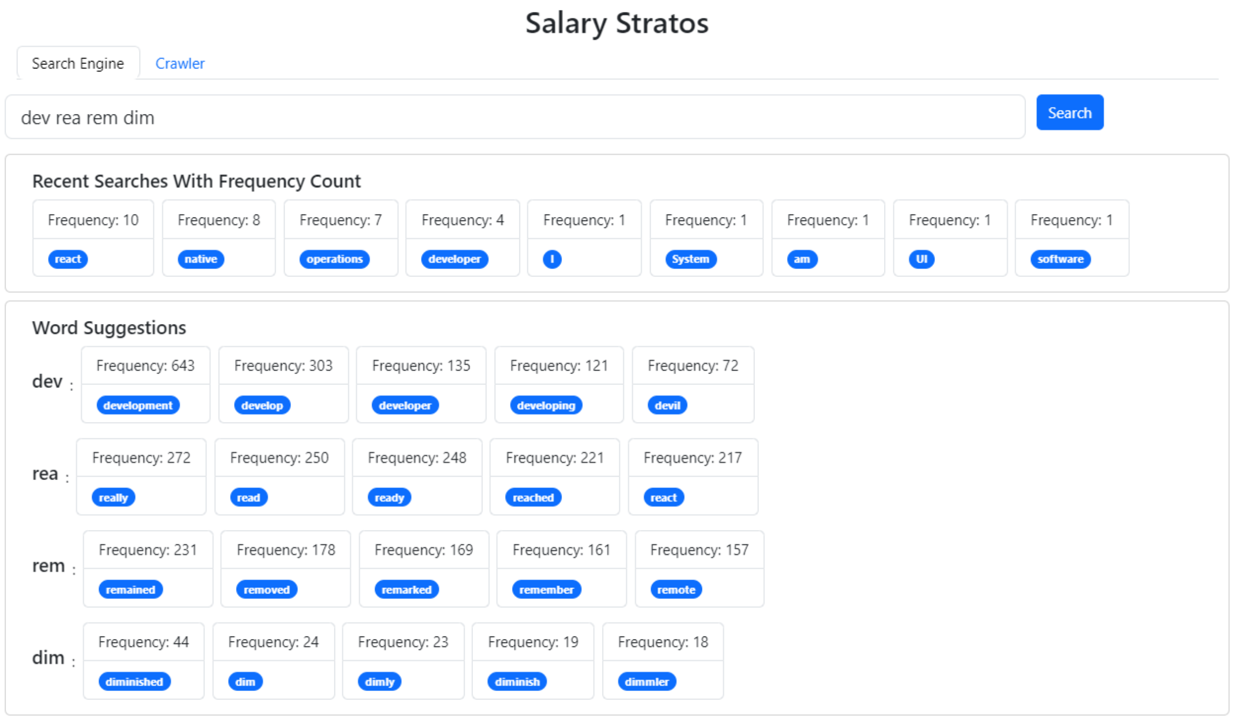
Frequency: 10	Frequency: 8	Frequency: 7	Frequency: 4	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	native	operations	developer	I	System	am	UI	software

Word Suggestions

dev :	Frequency: 643 development	Frequency: 303 develop	Frequency: 135 developer	Frequency: 121 developing	Frequency: 72 devil
rea :	Frequency: 272 really	Frequency: 250 read	Frequency: 248 ready	Frequency: 221 reached	Frequency: 217 react

Output 3 :

Word Suggestion for multiple search terms when the user enters a partial word



dev rea rem dim

Recent Searches With Frequency Count

Frequency: 10	Frequency: 8	Frequency: 7	Frequency: 4	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1	Frequency: 1
react	native	operations	developer	I	System	am	UI	software

Word Suggestions

dev :	Frequency: 643 development	Frequency: 303 develop	Frequency: 135 developer	Frequency: 121 developing	Frequency: 72 devil
rea :	Frequency: 272 really	Frequency: 250 read	Frequency: 248 ready	Frequency: 221 reached	Frequency: 217 react
rem :	Frequency: 231 remained	Frequency: 178 removed	Frequency: 169 remarked	Frequency: 161 remember	Frequency: 157 remote
dim :	Frequency: 44 diminished	Frequency: 24 dim	Frequency: 23 dimly	Frequency: 19 diminish	Frequency: 18 dimmler

8. Search frequency :

Code Explanation :

The SearchFrequency class manages search term frequencies using an LRUCache, maintaining a limited size to track the most recent and frequently used search terms. The LRUCache, implemented as a custom class extending LinkedHashMap, enforces a Least Recently Used behavior. It stores search terms and their frequencies, automatically removing the least recently used entry when the cache exceeds the specified size. The displaySearchFrequencies method retrieves the last 10 search term frequencies in descending order using a custom SortedArray data structure, providing a dynamically updated view of recent search term activities. The updateSearchFrequency method ensures accurate tracking of user interactions by updating the search term frequencies based on an array of search terms. Overall, this component efficiently combines LRUCache and sorted arrays to offer insights into recent user search patterns.

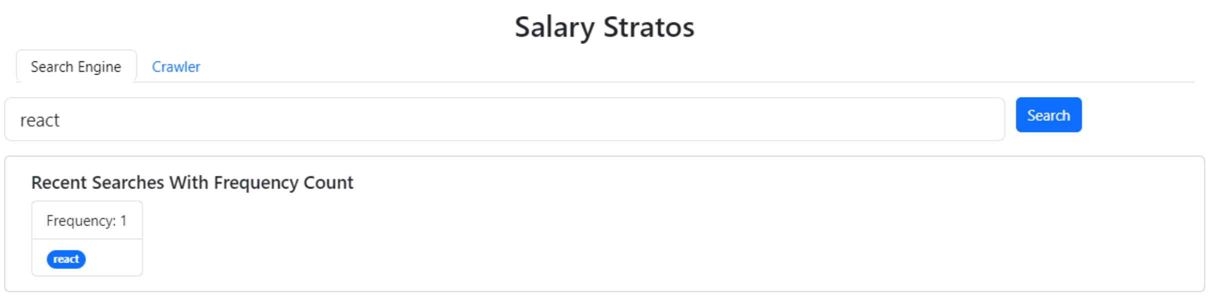
LRU Cache size is 50

The return array size to ui is 10

If the Cache size becomes bigger than 50 it will remove the least recently used word from the LRU Cache

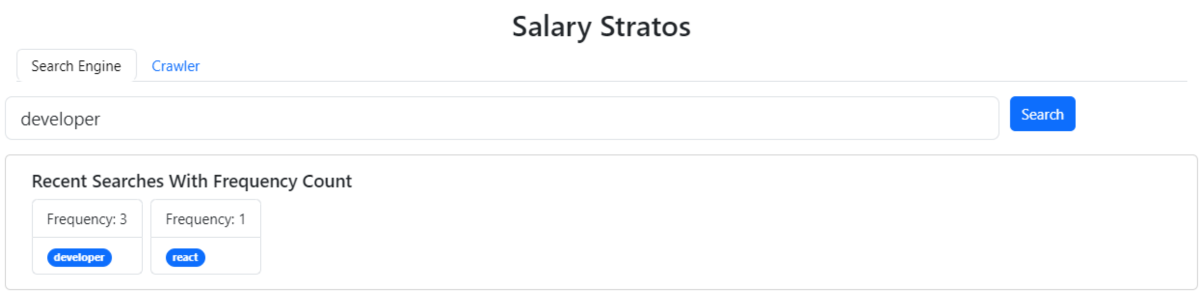
Output 1 :

First word entered into the LRU Cache “react”



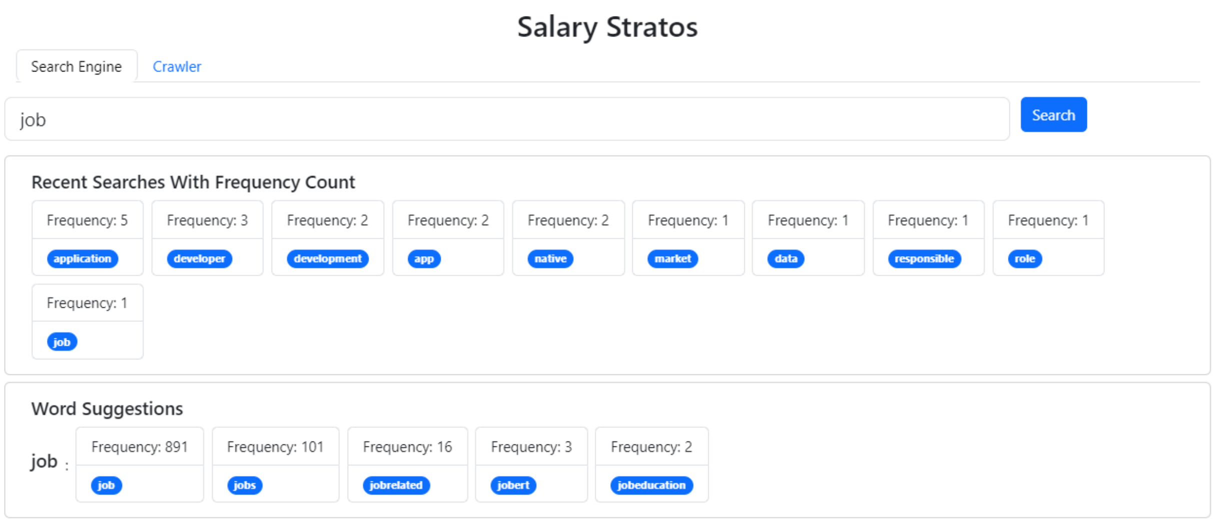
Output 2 :

Second word entered into the LRU cache "developer" which is searched 3 times



Output 3 :

After inserting 10 words we see that "react" is not shown in the ui anymore because it is not used for a while, but it still remains in the cache till the cache size becomes 50, when this happens if "react" is the least recently used word it will get removed from the cache.



9. Finding patterns using regular expressions :

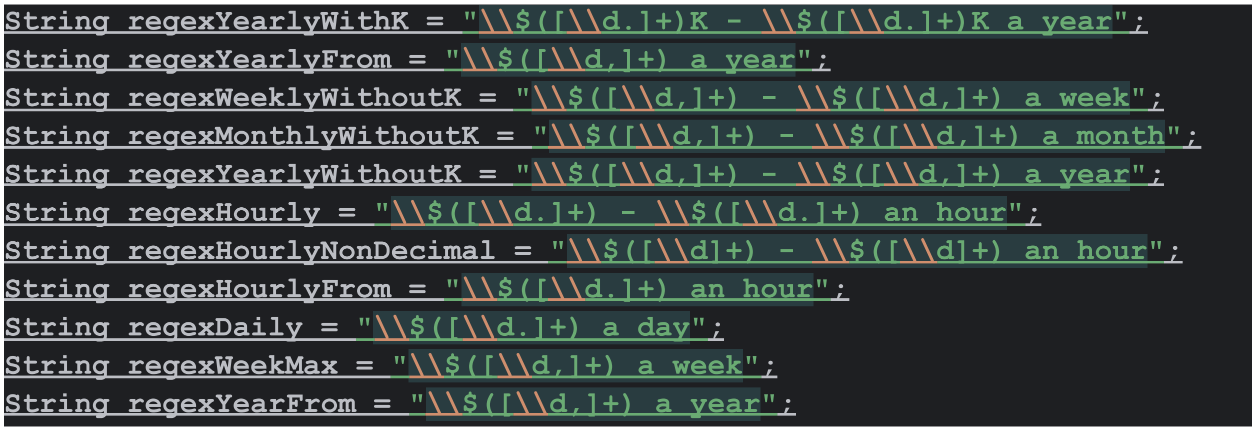
Code Explanation :

The scrapeWebPage method in classes such as SimplyHiredScrapper, RemoteOk, and GlassDoorScrapper utilizes regular expressions (Regex) to identify various salary formats and convert them into a consistent yearly format. The code initializes multiple regular expression patterns (regexYearlyWithK, regexYearlyFrom, etc.) designed to match different salary structures, such as hourly, weekly, monthly, or yearly, with or without the use of 'K' (thousands) and other variations.

After extracting the raw salary string from the web page, the code checks which regex pattern matches the format and proceeds with appropriate processing. For instance, it replaces the matched pattern with specific placeholders and then uses replaceAll to extract numeric values. It further adjusts these values based on the specific time unit (hours, weeks, etc.) to convert them into a standardized yearly representation. The resulting minimum and maximum salary values are then utilized in the application for analysis and comparison. This approach ensures accurate extraction and uniform representation of salary data, providing consistency in further processing and analysis

Output 1 :

If salary in Simply Hired is anything other than the below it will print the failure message



```
String regexYearlyWithK = "\\$([\\d.]+)K - \\$([\\d.]+)K a year";
String regexYearlyFrom = "\\$([\\d,]+) a year";
String regexWeeklyWithoutK = "\\$([\\d,]+) - \\$([\\d,]+) a week";
String regexMonthlyWithoutK = "\\$([\\d,]+) - \\$([\\d,]+) a month";
String regexYearlyWithoutK = "\\$([\\d,]+) - \\$([\\d,]+) a year";
String regexHourly = "\\$([\\d.]+) - \\$([\\d.]+) an hour";
String regexHourlyNonDecimal = "\\$([\\d]+) - \\$([\\d]+) an hour";
String regexHourlyFrom = "\\$([\\d.]+) an hour";
String regexDaily = "\\$([\\d.]+) a day";
String regexWeekMax = "\\$([\\d,]+) a week";
String regexYearFrom = "\\$([\\d,]+) a year";
```

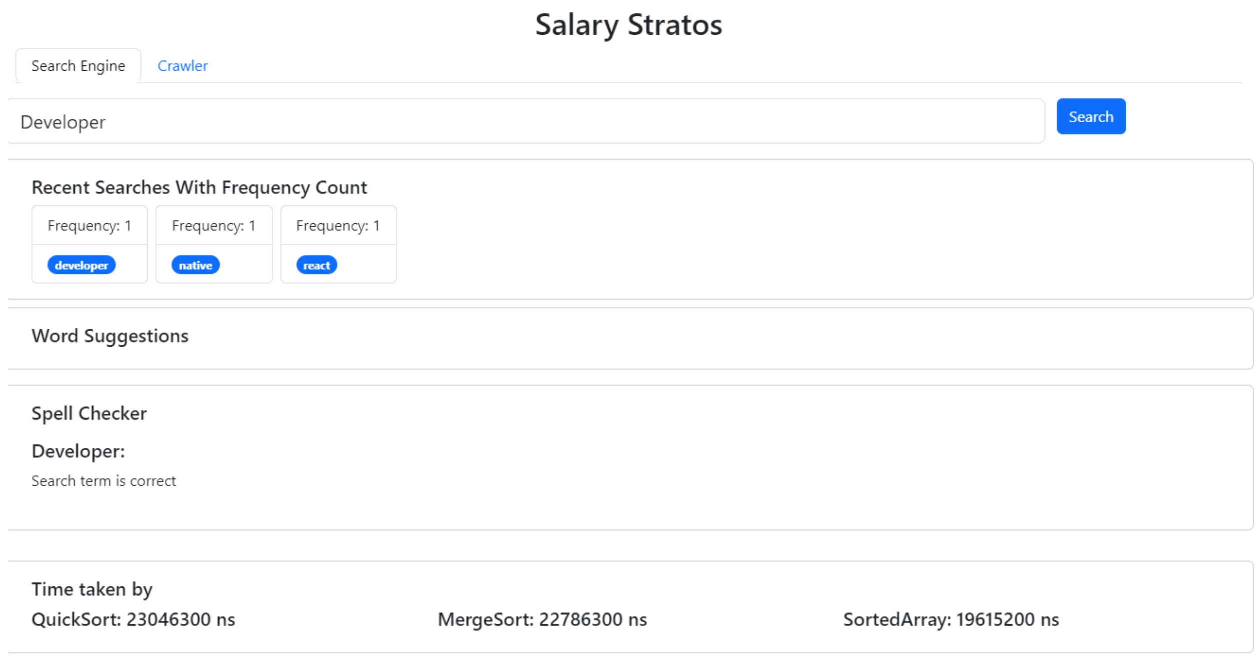
10. Compare sorting algorithms for Page ranking :

Code Explanation :

The CompareRunTimesData class is designed to store and manage runtimes for different sorting algorithms, particularly Merge Sort, Binary Search, and QuickSort, within the context of inverted indexing and page ranking. Each sorting algorithm's runtime is tracked through dedicated variables (mergeSort, binarySearch, quickSort) along with respective getter and setter methods. This implementation is intended to facilitate runtime comparison between these algorithms when applied to the specific task of sorting jobs retrieved from the inverted index. By setting and retrieving the runtimes for each sorting operation, this class provides a streamlined way to evaluate and contrast the efficiency of these sorting methods in the context of page ranking tasks.

In the context of maintaining a sorted array for inverted indexing and page ranking, binary search holds advantages over Merge Sort and QuickSort. Binary search excels in scenarios where the data is already partially sorted, which aligns with the situation encountered when maintaining a sorted array. Given its time complexity of O(log n) for finding the correct insertion position in a sorted array, binary search proves notably faster than both QuickSort (O(n log n)) and MergeSort (O(n log n)) in this specific context. Binary search's efficiency in locating insertion positions within a sorted array directly complements the task of maintaining order in the context of inverted indexing, making it a more suitable choice in scenarios where incremental sorting is required while minimizing time complexity.

Output 1 :


Salary Stratos

Developer

Recent Searches With Frequency Count

Frequency: 1 developer
Frequency: 1 native
Frequency: 1 react

Word Suggestions

Spell Checker

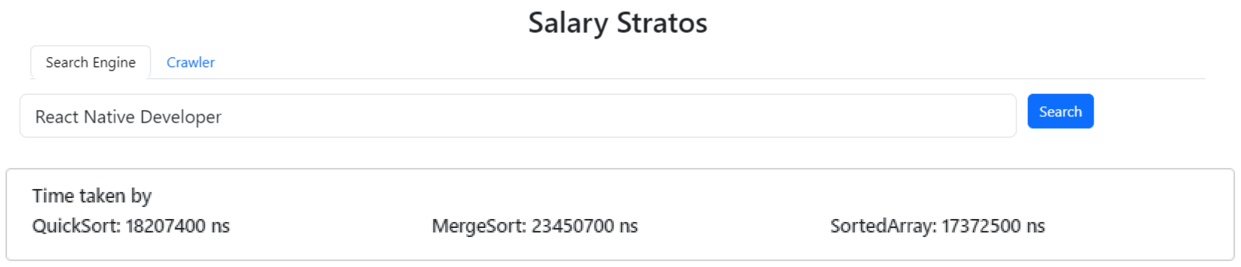
Developer:

Search term is correct

Time taken by

QuickSort: 23046300 ns
MergeSort: 22786300 ns
SortedArray: 19615200 ns

Output 2 :


Salary Stratos

React Native Developer

Time taken by

QuickSort: 18207400 ns
MergeSort: 23450700 ns
SortedArray: 17372500 ns

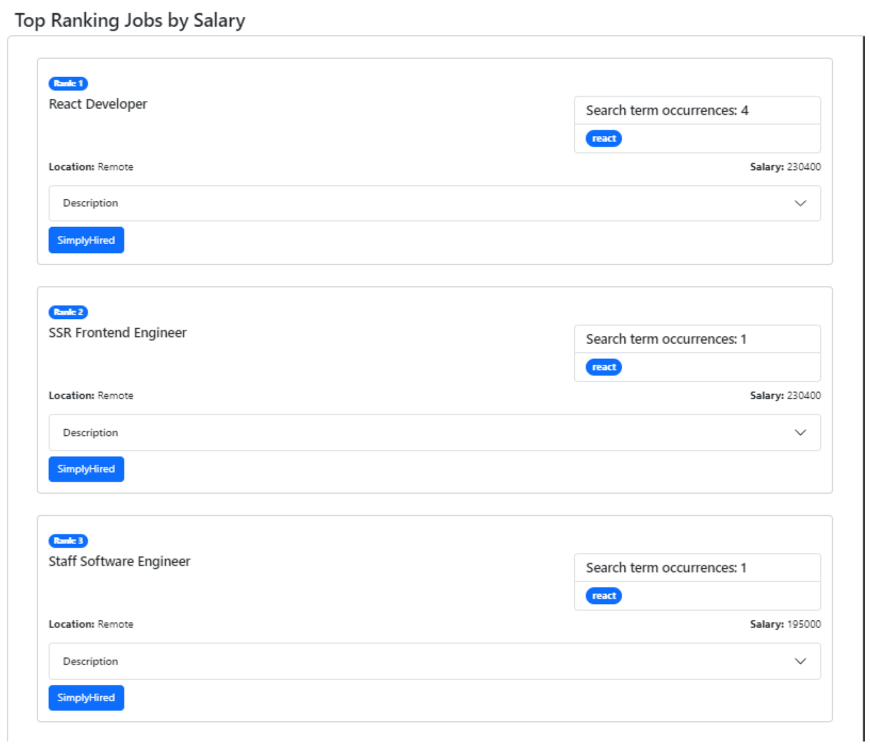
10. Recommend top paid jobs based on search results :

Code Explanation :

The searchInvertedIndexedDataBySalary method in the PageRanking.java class efficiently ranks jobs based on a combination of their maximum salary and word frequency. Leveraging a Trie data structure (TrieDS) for inverted indexing of job-related terms, the function iterates through the provided search terms, retrieves relevant information from the trie, and calculates the cost for each job. The cost is determined by adding the maximum salary and the word frequency associated with each job. Jobs are then inserted into a SortedArray based on their calculated cost, ensuring that the array remains sorted in descending order of cost. This approach enables the quick identification of top-ranking jobs, and in cases where multiple jobs have the same salary, they are further sorted based on their word frequency. The result is an efficient mechanism for users to explore and discover jobs that align with their preferences, particularly focusing on the combined factors of salary and word frequency.

Output 1 :

Top paid jobs for “React”



Top Ranking Jobs by Salary

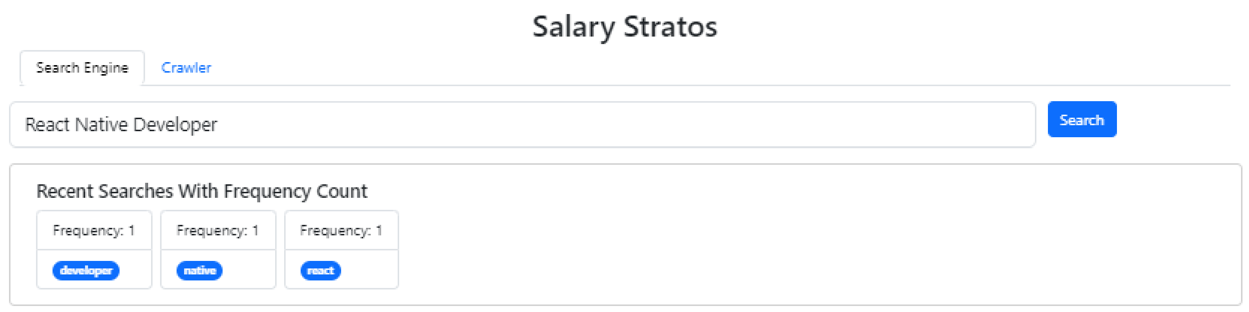
Rank: 1
React Developer
Search term occurrences: 4
react
Location: Remote
Salary: 230400
Description
SimplyHired

Rank: 2
SSR Frontend Engineer
Search term occurrences: 1
react
Location: Remote
Salary: 230400
Description
SimplyHired

Rank: 3
Staff Software Engineer
Search term occurrences: 1
react
Location: Remote
Salary: 195000
Description
SimplyHired

Output 2 :

Top paid jobs for “React Native Developer”



Salary Stratos

Search Engine | Crawler

React Native Developer

Search

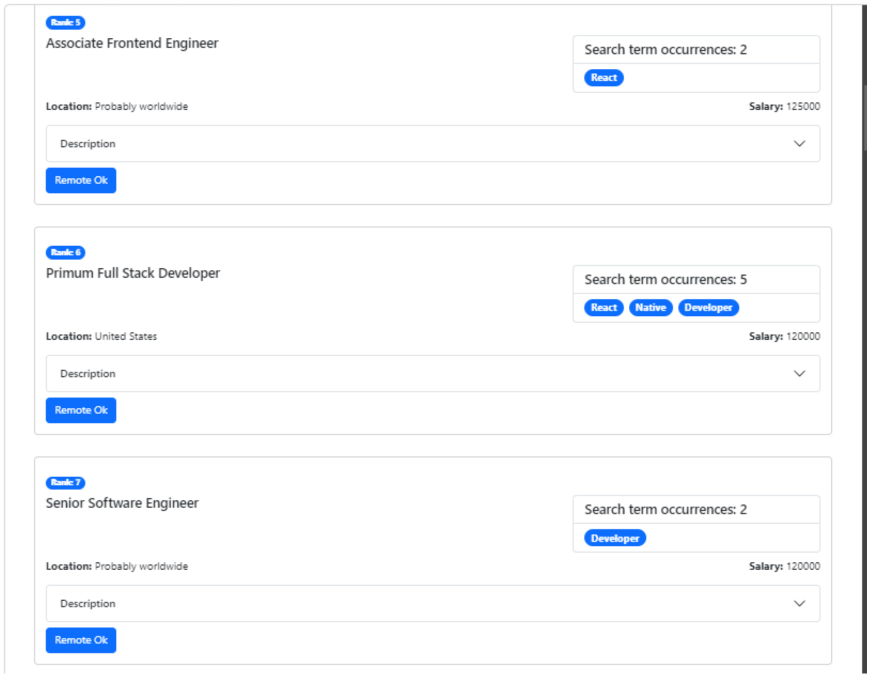
Recent Searches With Frequency Count

Frequency: 1
developer

Frequency: 1
native

Frequency: 1
react

Top Ranking Jobs by Salary



Rank: 5

Associate Frontend Engineer

Search term occurrences: 2

React

Location: Probably worldwide

Salary: 125000

Description

Remote Ok

Rank: 6

Primum Full Stack Developer

Search term occurrences: 5

React Native Developer

Location: United States

Salary: 120000

Description

Remote Ok

Rank: 7

Senior Software Engineer

Search term occurrences: 2

Developer

Location: Probably worldwide

Salary: 120000

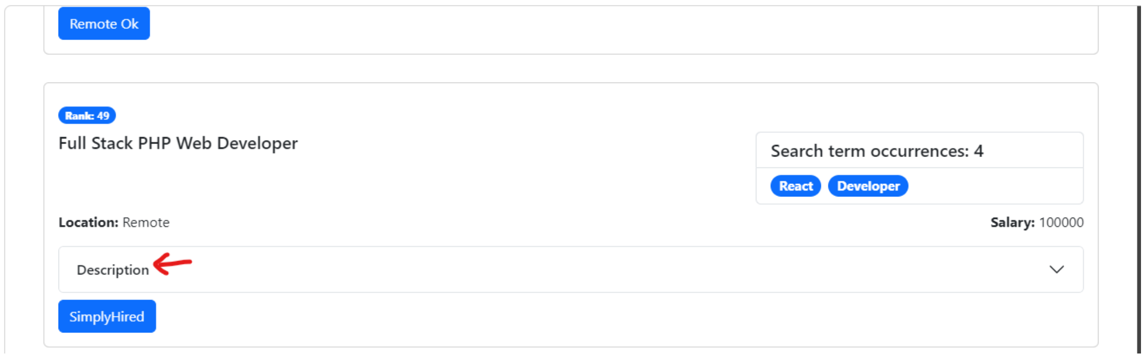
Description

Remote Ok

Output 3 :

We also show the description of the job along with the location, title, salary etc. When we click on the description it shows the job description.

Top Ranking Jobs by Salary



Remote Ok

Rank: 49

Full Stack PHP Web Developer

Search term occurrences: 4

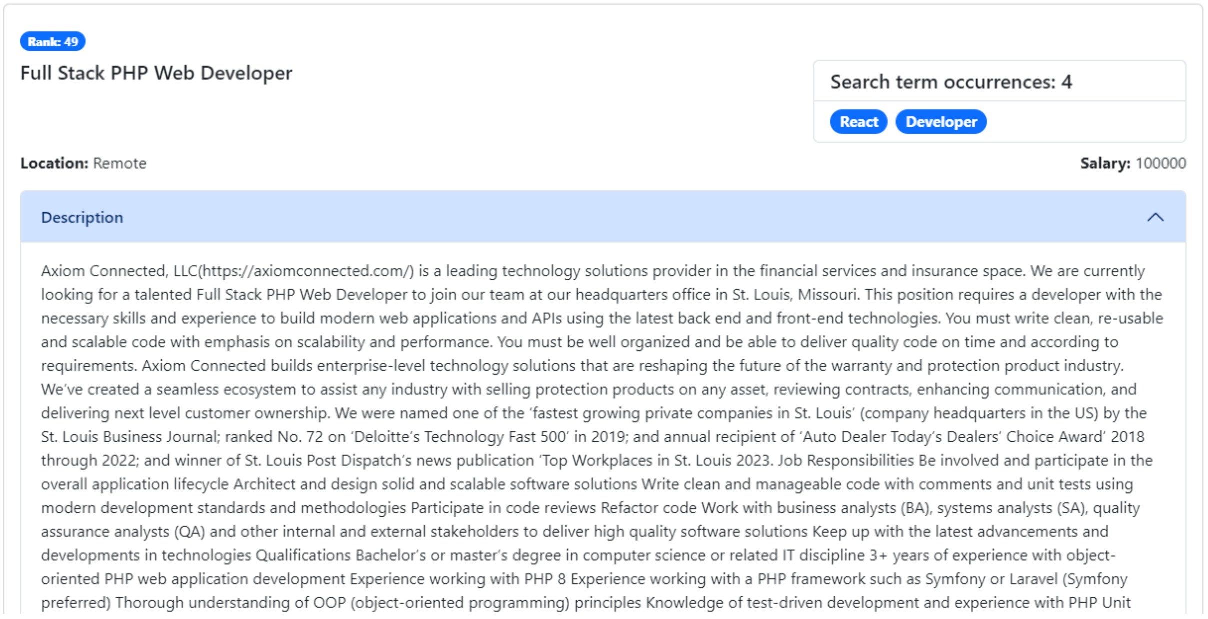
React Developer

Location: Remote

Salary: 100000

Description

SimplyHired



Rank: 49

Full Stack PHP Web Developer

Search term occurrences: 4

React Developer

Location: Remote

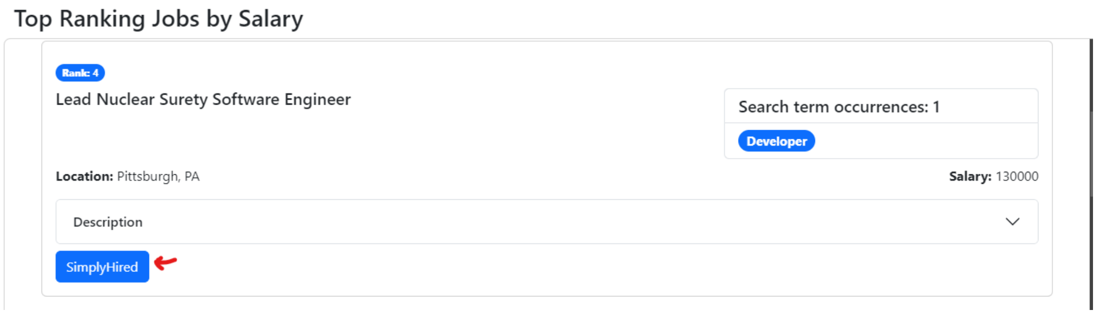
Salary: 100000

Description

Axiom Connected, LLC(https://axiomconnected.com/) is a leading technology solutions provider in the financial services and insurance space. We are currently looking for a talented Full Stack PHP Web Developer to join our team at our headquarters office in St. Louis, Missouri. This position requires a developer with the necessary skills and experience to build modern web applications and APIs using the latest back end and front-end technologies. You must write clean, re-usable and scalable code with emphasis on scalability and performance. You must be well organized and be able to deliver quality code on time and according to requirements. Axiom Connected builds enterprise-level technology solutions that are reshaping the future of the warranty and protection product industry. We've created a seamless ecosystem to assist any industry with selling protection products on any asset, reviewing contracts, enhancing communication, and delivering next level customer ownership. We were named one of the 'fastest growing private companies in St. Louis' (company headquarters in the US) by the St. Louis Business Journal; ranked No. 72 on 'Deloitte's Technology Fast 500' in 2019; and annual recipient of 'Auto Dealer Today's Dealers' Choice Award' 2018 through 2022; and winner of St. Louis Post Dispatch's news publication 'Top Workplaces in St. Louis 2023. Job Responsibilities Be involved and participate in the overall application lifecycle Architect and design solid and scalable software solutions Write clean and manageable code with comments and unit tests using modern development standards and methodologies Participate in code reviews Refactor code Work with business analysts (BA), systems analysts (SA), quality assurance analysts (QA) and other internal and external stakeholders to deliver high quality software solutions Keep up with the latest advancements and developments in technologies Qualifications Bachelor's or master's degree in computer science or related IT discipline 3+ years of experience with object-oriented PHP web application development Experience working with PHP 8 Experience working with a PHP framework such as Symfony or Laravel (Symfony preferred) Thorough understanding of OOP (object-oriented programming) principles Knowledge of test-driven development and experience with PHP Unit

<u>Output 4 :</u>

The job website name is a button which has the link to the job. When clicked on the job website name it takes to the job website where the job is posted.



Top Ranking Jobs by Salary

Rank: 4

Lead Nuclear Surety Software Engineer

Search term occurrences: 1

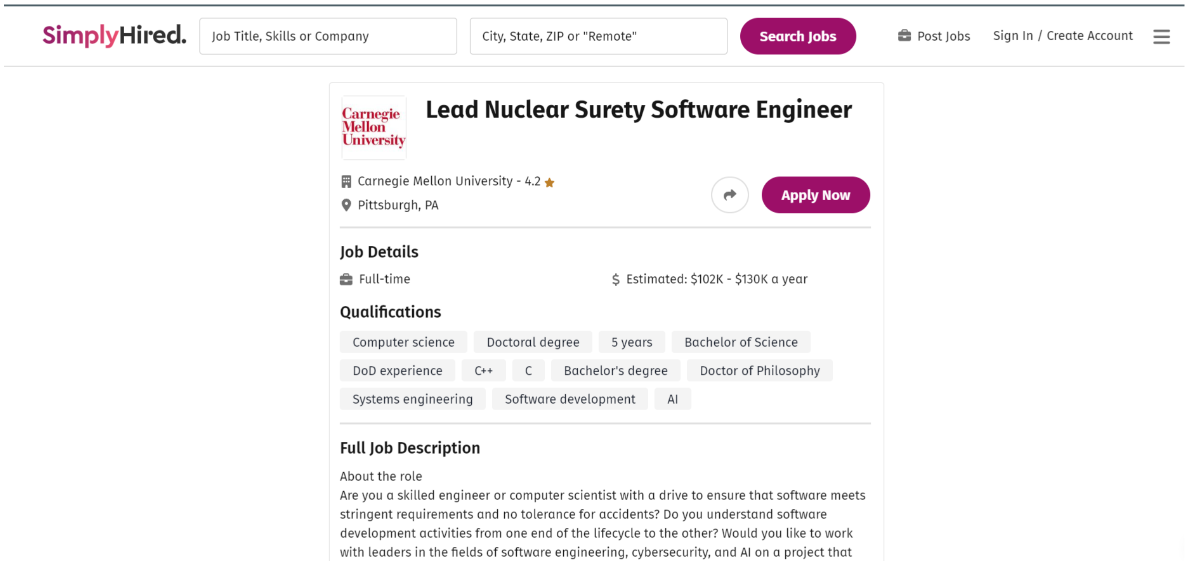
Developer

Location: Pittsburgh, PA

Salary: 130000

Description

SimplyHired



SimplyHired.
Job Title, Skills or Company
City, State, ZIP or "Remote"
Search Jobs
Post Jobs
Sign In / Create Account
Carnegie Mellon University

Lead Nuclear Surety Software Engineer

Carnegie Mellon University - 4.2 ★

Pittsburgh, PA

Apply Now

Job Details

Full-time

$ Estimated: $102K - $130K a year

Qualifications

Computer science | Doctoral degree | 5 years | Bachelor of Science | DoD experience | C++ | C | Bachelor's degree | Doctor of Philosophy | Systems engineering | Software development | AI

Full Job Description

About the role

Are you a skilled engineer or computer scientist with a drive to ensure that software meets stringent requirements and no tolerance for accidents? Do you understand software development activities from one end of the lifecycle to the other? Would you like to work with leaders in the fields of software engineering, cybersecurity, and AI on a project that

References :

1. Link to Selenium: The Selenium Browser Automation Project
2. Jsoup Link: Parse a document from a String: jsoup Java HTML parser
3. Edit Distance : Edit Distance - GeeksforGeeks
4. Trie Data Structure Link: Trie Data Structure in Java | Baeldung
5. Regular Expression Link: JavaScript RegExp Reference.
6. Simply Hired: SimplyHired
7. Remote Ok: Remote OK
8. Glass Door: Glassdoor