**Digital Skills For All – Job Portal Data Science Specification**

On our web application, we will have a “Job Portal” that allows users (primarily students) to browse job listings, which will be retrieved from various sites dedicated to showing job listings.

We want to get job listing data to be able to show in the Job Portal, as well as to analyze:

* Which types of entry level technical jobs there are
* Details about the different technical jobs
* Which are the most prevalent
* And which technologies are used in each different type of job

For now, we will be focused only on the job site Indeed. Although there are many other job sites as well, Indeed is the most used. (<https://www.indeed.com/>).

To gather the job listing data, we must scrape Indeed’s website for job listings and any data associated with them.

Once we have the raw data, we must clean the data to identify which job roles there are (e.g. Web Developer, Data Scientist, etc.) as well as identifying which skills/technologies are most prevalent in each.

Once we have this, we’ll be able to conduct data analysis to gather statistics and insights from the job data and create visualizations.

**1 – Scraping the Data: Introduction**

The first step will be scraping the raw data from Indeed’s website.

Here is a tutorial that generally follows what we want to do, however the scale of ours will be larger:

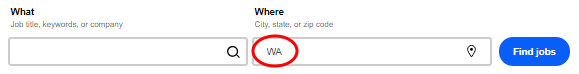
<https://jessesw.com/Data-Science-Skills/>

The link focuses only on Data Science job listings, while we want to collect and later categorize listings for all different kinds of jobs – for example, Web Developer, Data Scientist and more. We aren’t sure of what roles there are yet as there may be roles we don’t even know about yet, which we will find out during the data cleaning phase later.

First, we’d like to get all the raw data available for job listings in Washington.

**What do we need/What don’t we need?**

For now, we’re only interested in getting job listings in Washington State,



That are entry level (we aren’t interested in mid level or senior level job listings):



So a good place to start scraping would be here: <https://www.indeed.com/jobs?l=WA&explvl=entry_level>

Note that there are over 60,000 entry level jobs in Washington, but not all of them will be technical, for example, dog walking wouldn’t exactly be technical.

Later, we’ll want to have a way to get only the technical jobs.

I believe that the best way to do this is to not focus on filtering non-technical jobs out just yet. We will just get all entry-level Washington State jobs at first, then later there’s a step where we organize them all into consolidated roles (like instead of “Front End Web Developer” and “React Front End Web Developer”, it would just be consolidated into “Web Developer”) based on keywords present in the job listing’s role title.

There could be a negative state where if the title does not contain any of the keywords like “Developer” or “Database” or “Graphic Design” then we can assume it is not a technical job.

It’s up to our discretion what we consider a technical job to be, but it can come later after we discover all the different kinds of jobs there are by scraping.

But we don’t have to worry about this for now – we’ll get to it in the cleaning phase.

From each job listing – on a page like <https://www.indeed.com/jobs?l=WA&explvl=entry_level> we want to gather any information that may be useful about each listing.

We also want to get the job types, which isn’t always easy to find on each job listing page. However, what we can do instead is search for each job type one at a time e.g. <https://www.indeed.com/jobs?q=developer&l=WA&jt=fulltime&explvl=entry_level> and mark each of those entries under the appropriate job type (in the example case “Full-time”).

**2 – Scraping the Data from the main results page**

From the job listing results pages themselves, we can get this information without going into each individual page:



|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Column name** | **Description** | **Data type** |
| 1 | RoleTitle | The title of the job listing on Indeed. Not necessarily a role that can be organized by our systems, since employers can use whatever title they want here.  Later (but not now), we’ll want to change all these unique role titles into more general ones that we can organize. | String |
| 2 | Company | The name of the company who created the job listing. | String |
| 3 | FullLocation | The entire 3rd row, which may or may not include state, city, zip code and area. | String |
| 4 | LocationState | From the FullLocation field, we should be able to get the State, which will probably always be formatted as a state code (WA, OR, CA, NY, etc.) | String |
| 5 | LocationCity | From the FullLocation field. City seems to be always there, but it may not be (Seattle, Bellevue, Redmond, etc.) | String |
| 6 | LocationZipCode | From the FullLocation ifeld, for example 98052 or 98101. Not always present. | String (Optional) |
| 7 | LocationArea | From the FullLocation field, seems to always be within parentheses. Not always present. (“Downtown area”, “Lake Hills” area, etc.) | String (Optional) |
| 8 | MinimumSalary | Minimum salary for the job, in dollars (e.g. 50000 corresponsd to $50,000) | Numeric |
| 9 | MaximumSalary | Maximum salary for the job, in dollars (e.g. 70000 corresponds to $70,000) | Numeric |
| 10 | SalaryTimeUnits | Some are in hours, some are displayed in years, so we want to detect if it is displayed hourly, or yearly. | String |
| 11 | ShortDescription | Each job listing on the results page has a brief version of the full description that appears on the individual page, we’d like to retrieve that blurb. | String |
| 12 | HasExpressApply | I believe Express Apply means you can apply directly through Indeed, rather than going to the employer’s website first. If the listing has express apply, mark this as true. | Boolean |
| 13 | IsSponsored | Whether or not the job listing is sponsored to appear at the top of the list rather than later. | Boolean |
| 14 | IndeedJobListingURL | The URL that takes you to the individual job listing page (e.g. <https://www.indeed.com/cmp/PeopleTech/jobs/Front-End-Developer-df5ab874b500fca2?sjdu=QwrRXKrqZ3CNX5W-O9jEvRfQ2IAUyuwhAgxPW4jiRzUAwyCOOyMoWlIGk-pbmeU7YQoPYjJpt8r5jYxm0vamYnj8uvPZf3Mc9ur7IlSa6Oc&tk=1dd9mmtsd1d5l008&adid=248343861&vjs=3>)  You’ll have to scrape through this URL as well to retrieve even more information about the job listing. See next step. | String |
| 15 | ExperienceLevel | Entry Level/Mid Level/Senior Level.  This should be “Entry Level” for every listing to indicate that we are only looking at entry level listings. | String |

**2 - Scraping the data part 2: Fields from each individual page**

In addition to the results on the main page, there are a few more fields of information can be found on the individual job listing pages that do not require much data cleaning.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Column name** | **Description** | **Data type** |
| 16 | FullDescription | The whole description, preferably as just HTML so that we can read it more easily by loading it on a webpage. | String |
| 17 | JobType | Full-time/Contract/Part-time/Internship/Temporary/Commission  The easiest way to get this information seems to be to first sort by Job Type such as <https://www.indeed.com/jobs?q=developer&l=WA&jt=fulltime&explvl=entry_level> and mark every entry there as Full Time, then do the same for all other job types. (Internship, Contract, etc.) | String |

There may be others than the ones that will be listed, so if you see anything else that is retrievable from the individual job listing page, feel free to let us know or add any.

**3 – Data Cleaning: Deriving and Categorizing Information about each Job Listing**

After getting all the raw data, we want to derive information about the job listings that is not immediately apparent:

* From the job listing’s RoleTitle, we want to organize different job listings that are in the same field, but were given different RoleTitles by the employers, who could write whatever they want.
* And from the job listing’s individual page’s Full Description, we want to find all the Skills/Technologies that are used in each. For example, for a Web Developer job listing, we would hope to find at least HTML, CSS, and JavaScript as a technology that is commonly used for Web Developer jobs. This will be difficult as employers can write whatever role title they want, they can also write any description they want, so job listings will not always follow a standard format. However, there are common themes, such as a Responsibilities and Qualifications sections which are usually denoted by an HTML list.

**3.1 Categorizing the RoleTitles and Filtering out Non-Technical Jobs**

While the titles that employers have given all the different job listings vary greatly, we want to organize them each into a more general term.

We aren’t sure what roles are out there. We have some estimates like Web Developer, Data Scientist, UI/UX etc., but there could be more that we haven’t considered.

So we should take all the titles from the raw data, and use some sort of keywords system to organize them into different roles that we see fit – Web Developer, Data Scientist, etc.

This should be in a new field for each column – something like “NewRoleTitle” would suffice to indicate that it’s based on “RoleTitle”.

During this step would be a good time to remove any non-technical jobs. Any jobs that don’t fall under the keywords that organize it into our consolidated list of role titles should be considered non-technical, but we should be careful to avoid false negatives.

**3.2 Getting the skills associated with each role**

This is a large task.

We want to extract the skill requirements from every technical job description to create a list of skills such as HTML, CSS, JavaScript, etc.

What I would recommend is on a role by role basis for our categorized roles, (e.g. looking at Web Developer role specifically) look through the description pages.

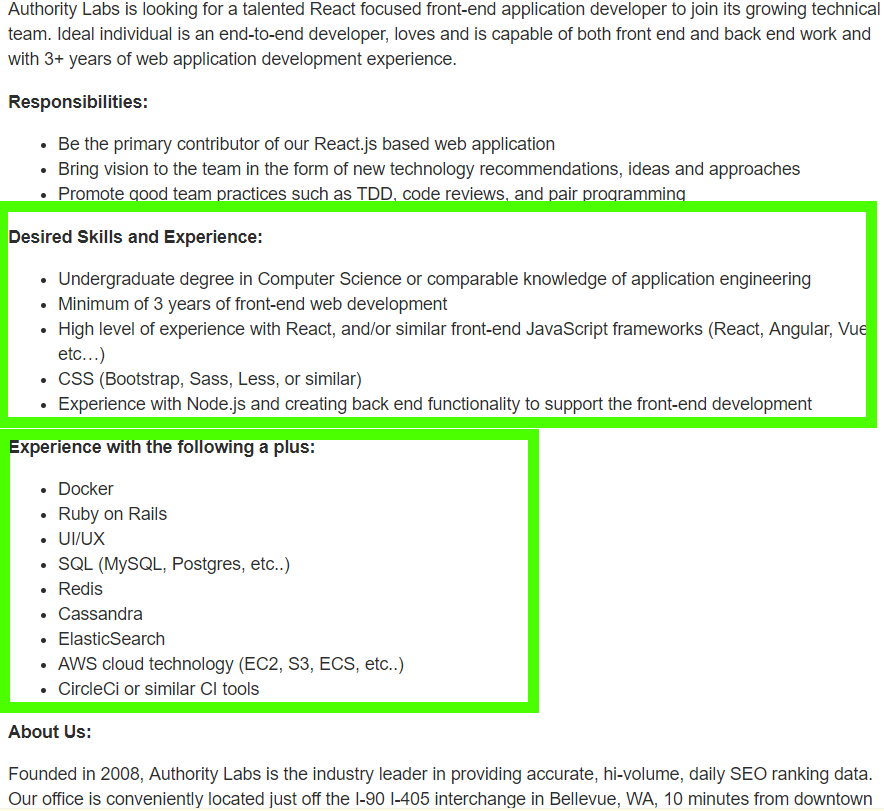
Then we’ll have to have a way to find the Qualifications/requirements/etc. sections where skills/technology prerequisites are listed.

From there, I suggest making a word frequency count dataframe that focuses only on one role type (e.g. Web Developer) then find any words that are commonly used, like “React” or “HTML” (I noticed that some employers don’t capitalize these sometimes, but usually they should be capitalized – if that’s the case, we could maybe only find Capitalized words to verify) then create a list of these skills. Any words that are not actually technologies (“then”, “and”, “or”, etc.) should of course be ignored.

Then go through each job’s qualification section of the description and check if any of the words appear – if they are, then the skill/technology is associated with the job listing.

However, most listing pages should have a “Skills and Qualifications” area which should have the bulk of the skills needed.

Most of the time, they will be in a list such as these:

* 

Of course, the list titles will be different as they are written by employers. But “Experience”, “Skills”, “Qualifications” lists seem to be fairly common for tech jobs, so I would recommend only checking there.

I would recommend trying to get this skills information while scraping the HTML of the webpages, rather than getting the descriptions first then attempting to extract them from the description field.

Many job listing pages have both a “required” list of skills, as well as a “good to have” list of skills. So if possible, we want to have a separate field for each so for data analysis we can see which skills are required vs. which ones are required.

**4 - Data Analysis: Statistics, Visualizations, Insights**

Hopefully this is the fun part

Now that we have all the job listings organized by the job titles, that are only entry level Tech jobs in Washington, as well as skills associated with each, we can start data analysis.

Here are some questions that we’d like to have answered based on the cleaned data. Visualizations and charts should be made for each.

Some questions I can think of (There will likely be more questions raised than just these)

1. Which job roles are the most popular for entry level tech? (How many job listings are there for each)
2. What are the mean & median salaries for each tech job role?
3. Which cities have the most total tech jobs?
4. For each job role, what are the skills/technologies that are most requested by employers?
   * Which ones are considered required, and which are considered not required but good to have?
5. Is there a difference in salary between entry-level tech jobs and entry-level non-tech jobs?

Data exploration is very welcome! You’re encouraged to come up with questions for yourself as well, then answer them using the data and draw insights from it.

**5 – Changing CSV Files into a Database**

While the CSV files are sufficient and ideal for data analysis, we also want to include the job data in the Job Portal aspect of the Digital Skills For All application.

We will have designed an Entity Relationship Diagram (ERD) that includes the Job table, but the fields may have changed based on what data we are able to get out of Indeed.

The database will likely be made in NoSQL.

We’d like to change the CSV job data into data in the NoSQL database.

**6 - Detecting if/when the page structure of Indeed changes, such that any scripts no longer correctly scrape the data**

We’d like to keep the data up to date, which is not always easy when dealing with scraped data.

This will involve 2 steps, the first of which is making sure the job listing data can always be retrieved from Indeed’s website.

We should have a check in the scripts you use that detects if there is a change to Indeed’s website that makes data no longer scrapable. Many websites redesign from time to time, which may cause our web scraping scripts to no longer work.

If the script repeatedly fails to get information that it previously had no problem getting, then it should alert us and print which variables failed in retrieving.

**7 – Keeping the database up to date**

To make sure our job data on our website is always up to date, we want to attach the scraping script to a web server.

Every week or so, the web server should run the script again.

* We want to check if each job listing already exists, and if it does, we want to update the data rather than add it again. This can be done by checking things like the URL and the ID of the job on Indeed.
* We want to check for if job listings have been removed – if you go to this page: <https://www.indeed.com/viewjob?jk=d71d41d27bc93677&from=tp-serp&tk=1ddbo4almbhpa803> you’ll see that it says that the job posting is no longer available. In this case, we want to find the job listing entry in our database then mark it as no longer available, so that we can hide it from the job portal.
* And if there are new job listings that we know for sure aren’t already present in the database, then we want to create new entries for them.
* Every job listing should have a field for what date it was added to the database on.

**8 – Creating a Recommendation Engine**

This is a stretch goal that we probably won’t be able to get to, but we’d like to create a recommendation engine that recommends jobs to students if we have time.