LinkedIn Automatic Job Application Bot

Project by: Luke Rowe

Advised by: Dr. Jane Lehr

California Polytechnic University SLO

5/16/23

The need for a job bot:

In my own world I have been steadily applying for jobs. The bad part about it however is I spend hours a day doing the same task over and over and I definitely miss a lot of opportunities. If 99% of the fortune 500 companies are already using Application Tracking Systems (ATS) to screen applications automatically (Business Insider), I figure that I should do the same for this project and apply automatically. This bot would be a beneficial bot to use as it could apply, at least as fast as me, throughout the day and save me dozens of hours of time. This could then be broadened to the general job market. Imagine a world where both employers and employees could automatically connect with each other, no more worthless applications or scrolling through thousands of listings. Of course I can't make something at that scale, however I can make a bot that uses the current system in a way to make it one step closer.

Existing projects:

There are other projects that are similar to this one. From what I've seen the bots that are pushed out work, and some even have extremely valuable bits of code that I found to be useful to know about (such as the ability to filter via URL and not by clicking filter buttons). I found similar projects both on the internet with gated code and paid use, as well as public github repositories. While these projects were already made and available, I wanted to create something better and more efficient. These are the following projects I found:

https://github.com/nicolomantini/LinkedIn-Easy-Apply-Bot

https://github.com/NathanDuma/LinkedIn-Easy-Apply-Bot

Honing my project:

Both projects effectively use selenium like I did. Both projects both have a lot of code. What both projects suffer from is inefficiency and overcomplicating the ease of creating a bot. Both projects are average 600 lines of code… for the main bot file, this does not include other supporting files. My code is 200 lines and shares most of the same features. Creating easy examples is useful for both people to follow, as well as to learn from. KISS is an important design principle to follow within engineering. KISS is simply, Keep It Simple, Stupid! It means that if it can be made simply, make it simply, do not needlessly overcomplicate it (something a lot of STEM people find themselves doing). I decided that I wanted to redo what they did but better, and the goal of this was to make a simple bot that does a simple task.

Criterium for success:

- Apply to 50 jobs

Pseudo apply (get up until the submit button) to 50 jobs to indicate that the program works with the following criterion. The criterium is tracked and monitored by outputting trackers to a CSV (see CSV image). This criterion was successful, though the CSV image is only a glimpse to the amount "applied" to. See figure 2.

    - Fail rate < 90%

For the purposes of this project, a fail means that there are code related exceptions or throws due to stale requests, bad requests, or other non linked issues. This criterion passed in that the program will occasionally fetch a stale element on the page a couple times in a row, which will lead to the program to falsely skip the application. This only occurred in the program when there was an unstable internet connection, thus achieving the fail rate threshold.

- Improper job type < 10%

Ensure that the jobs applied to are the jobs indicated by the user. This criterion ended up not being as useful due to built-in filters. This filtering was up to LinkedIn to handle, however ensuring that the filters were properly applied was monitored.

- Accurate information submitted ~100%

Accurate information was given into the applications. This was vital to get correct due to lying on an application being a good way to get blacklisted from future applications. Due to the sheer amount of varying prompts and AI not being a viable solution, this criterion was accomplished by only pushing applications with previously filled information or with no additional filling being needed. When unknown information was requested, the bot would throw and skip the application marking it as unsubmitted.

- Not rate limited / rate limit bypassed

Make sure that the program can run efficiently and not be throttled due to LinkedIn thinking it was a scraper or something against their TOS. Make it compliant to the degree it needs to be to not be rate limited. This criterion ended up not being an issue as I was never flagged for malicious use. The only rate limiting that occurs is waiting for certain page elements to load so that the bot can progress. Reading more into this and LinkedIn's TOS, I saw that the only action they do not like is web scraping to be used for other purposes, not my own use.

How it all works:

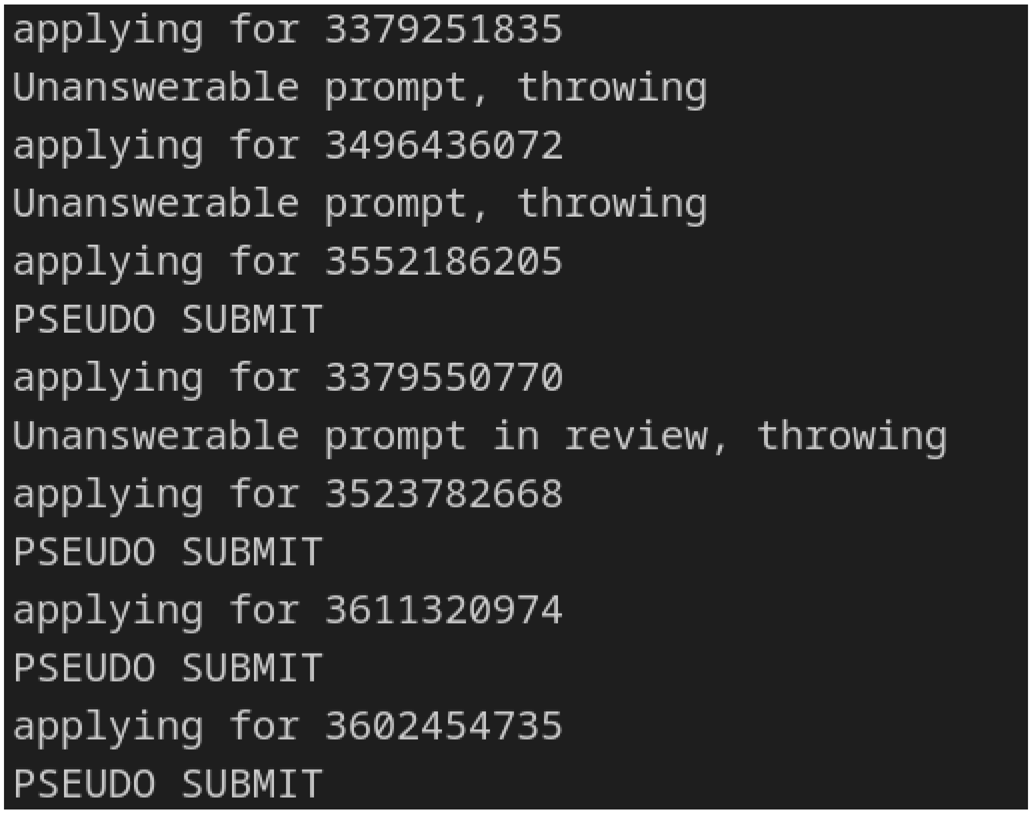
To start, as with all projects I used Git for version control, history, and general cloud storage. It is a public repository so that others may use it to expand upon or look at it for reference: https://github.com/RogShotz/CSC-491

I used python as the programming language and used the selenium package. Selenium is a web dev testing schema and provides many useful functions. Such functions include the automated use of web browsers, in my case, chromium (an open source version of Chrome). Via the functions in Selenium I was able to look and apply for jobs that I wanted to apply to.

Technical Feats:

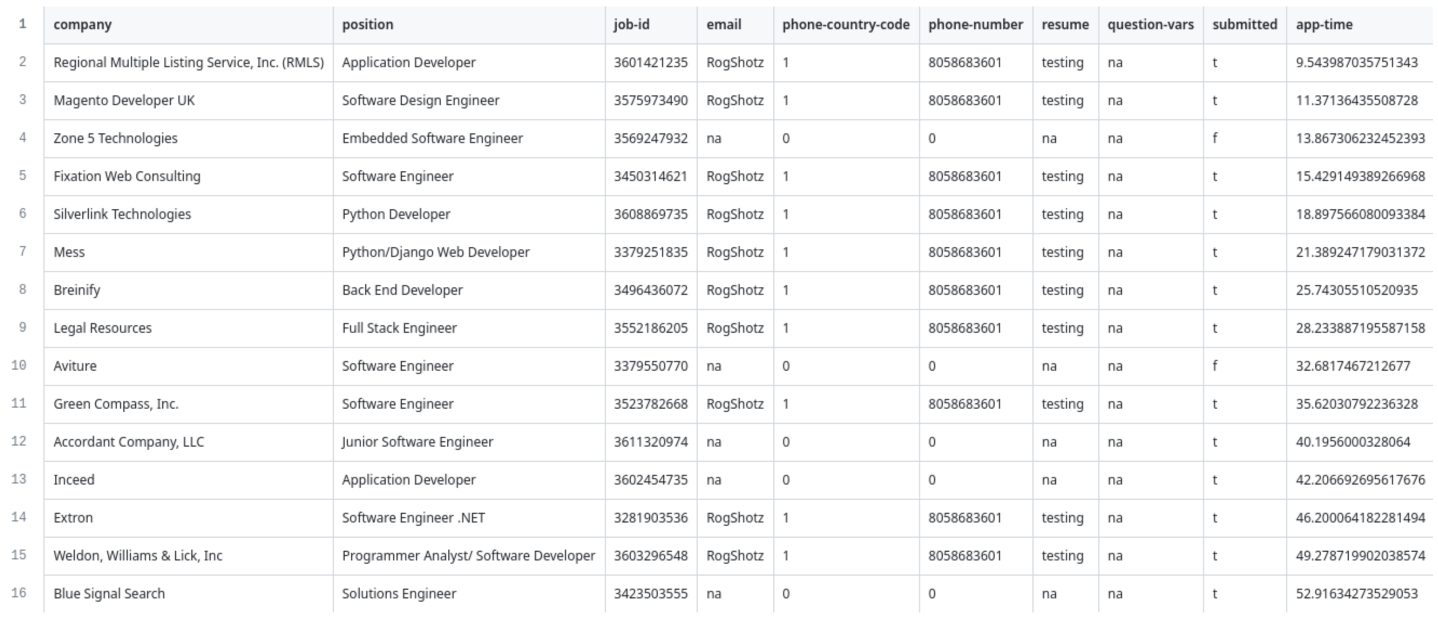
- Less than 200 lines of code
- Fully completed criterium for success
- Full use of selenium features
- Automatic login via cookies
- Easy filter adding and modification

Proof:



```
applying for 3379251835
Unanswerable prompt, throwing
applying for 3496436072
Unanswerable prompt, throwing
applying for 3552186205
PSEUDO SUBMIT
applying for 3379550770
Unanswerable prompt in review, throwing
applying for 3523782668
PSEUDO SUBMIT
applying for 3611320974
PSEUDO SUBMIT
applying for 3602454735
PSEUDO SUBMIT
```

1. The program applies for independent job ID’s. PSEUDO SUBMIT indicates that the final submit prompt button was “clicked” and thus submitted. In actuality for testing purposes it was not actually submitted. Other unerasable prompts indicate an error within the page due to an unknown prompt, thus throwing an error and moving onto the next scraped job ID.



company	position	job-id	email	phone-country-code	phone-number	resume	question-vars	submitted	app-time
Regional Multiple Listing Service, Inc. (RMLS)	Application Developer	3601421235	RogShotz	1	8058683601	testing	na	t	9.543987035751343
Magento Developer UK	Software Design Engineer	3575973490	RogShotz	1	8058683601	testing	na	t	11.37136435508728
Zone 5 Technologies	Embedded Software Engineer	3569247932	na	0	0	na	na	f	13.867306232452393
Fixation Web Consulting	Software Engineer	3450314621	RogShotz	1	8058683601	testing	na	t	15.429149389266968
Silverlink Technologies	Python Developer	3608869735	RogShotz	1	8058683601	testing	na	t	18.897566080093384
Mess	Python/Django Web Developer	3379251835	RogShotz	1	8058683601	testing	na	t	21.389247179031372
Breinify	Back End Developer	3496436072	RogShotz	1	8058683601	testing	na	t	25.74305510520935
Legal Resources	Full Stack Engineer	3552186205	RogShotz	1	8058683601	testing	na	t	28.233887195587158
Aviture	Software Engineer	3379550770	na	0	0	na	na	f	32.6817467212677
Green Compass, Inc.	Software Engineer	3523782668	RogShotz	1	8058683601	testing	na	t	35.62030792236328
Accordant Company, LLC	Junior Software Engineer	3611320974	na	0	0	na	na	t	40.1956000328064
Inceed	Application Developer	3602454735	na	0	0	na	na	t	42.206692695617676
Extron	Software Engineer .NET	3281903536	RogShotz	1	8058683601	testing	na	t	46.200064182281494
Weldon, Williams & Lick, Inc	Programmer Analyst/ Software Developer	3603296548	RogShotz	1	8058683601	testing	na	t	49.278719902038574
Blue Signal Search	Solutions Engineer	3423503555	na	0	0	na	na	t	52.91634273529053

2. Relevant datum on each independent job as well as application time. To note, the app-time average is around 3 seconds.

Conclusion/ going forward:

Going forward I would like to make this app something that can be integrated with multiple different services. The addition of a GUI/ web interface would also make it more usable for the general public. This system has the ability to be easily expanded upon as a base template. Other future additions would be creating a more stabilized system. As of now errors are common within the program right now and still have to be monitored and tweaked, with a broader user base these errors have to be resolved. These errors typically come with the use of selenium, as I am a self taught selenium user a much more robust system could definitely be created. Ideally, my own program would not be needed anymore in a world where instead of employees looking for companies, companies finding employees with exactly what they need on a centralized site, or a handful of sites with similar formats.

Citations:

ATS 99% of companies citation

Rennolds, Daniel Cáceres, Nathan. “Screening Filters May Be Sabotaging Your Job Prospects, but 3 Tricks May Help Your Resume Get Past Them.” *Business Insider*, www.businessinsider.com/3-keys-to-ensure-hiring-managers-read-your-resume-2021-9?op=1. Accessed 5 June 2023.

Selenium

“SeleniumHQ Browser Automation.” Www.selenium.dev, www.selenium.dev/.

Stack exchange, thank you to all of the different posts who helped throughout the dev process

Stack Overflow. “Stack Overflow - Where Developers Learn, Share, & Build Careers.” Stack Overflow, 2022, stackoverflow.com/.

Geeksforgeeks, various examples and walkthroughs

GeeksforGeeks. “GeeksforGeeks | a Computer Science Portal for Geeks.” GeeksforGeeks, 2019, www.geeksforgeeks.org/.