NATURAL LANGUAGE PROCESSING BOTS FOR JOB SEARCHING INTEGRATED WITH THE JOB RECOMMENDATION SYSTEM AND NOTIFICATION ALERTING SYSTEM

Sairam Nimmakayala*, Jyoshna Korada*, Geeshitha Mamidi*, Konusothu Sai Surendra Naik*,
Computer Science Engineering, Vignan's Institute of Information and Technology
Chintam Anusha Asst.Professor, Computer Science Engineering, Vignan's Institute of Information and Technology : anusha.rohini07@gmail.com

Abstract
Many organizations announce their job openings through different portals. This might become a difficult task for people who are in search of jobs. This also increases the complexity of updating the data and filling the forms every time. Our creation here lets users search for their desired job by reducing the burden of rushing over thousands of websites every time. This project scraps the data from different websites and makes the search jobs available for the user dynamically. This project also includes bots which interact with the users. This project also provides an alerting system for users by sending a notification whenever the new job openings are available based on user preferences**.**

Index Terms- Job Recommendation, Micro-Service, Notification System, NLP Bots, Multi-Threading, Web Scraping, Dialogue flow.

I. INTRODUCTION

Today the world is running at its fastest pace in every field. With the advancement in technology, every field started adapting virtualization. And with the rapid development, many jobs craved their path into existence and thereby, the competition has increased gradually for every job.
With this transformation most of the companies made their recruitment process online. The first step of any recruitment is posting the job vacancies. One of the ways to post job vacancies is through websites. Many companies have their own portals for recruiting, and few companies choose websites like linkedin, glassdoor, naukri, and many others. And for a person who is looking for a job, the task might become complex, as searching on each and every website is time taking and the person might miss a company in a hurry.
So, our portal is the solution for this problem. Our portal acts as an interface for users to apply for jobs which are posted in different websites, just by a search in our portal. Our portal recommends jobs to users through preferences that users set and sends notification alerts whenever the preferred job is posted, it also provides chat bot for WhatsApp users. Users can save the preferred jobs for future reference; users can also save their data in the portal.

II. OBJECTIVE

Its foremost objective is to make the task of searching jobs for every individual easier by scraping different websites and making them available just by single search based on categories like job title, domain, company, location. It also alerts users by sending notifications to the mail id that user has given so that the user will not miss an opportunity to grab a position, and will also send updates through WhatsApp chatbot and notifies through mail.

III. TECHNOLOGY STACK

A. ***Web Scraping***

Web scraping is an algorithm which is used to obtain data from websites. It can either obtain a chunk of data or the entire webpage. It returns the data in a raw format, unstructured. It is an automatic method

where data extraction is done dynamically. The unstructured format here can refer to the HTML format, this data can be used by various applications only after converting that into structured data and this transformation of structure can take place in a database or spreadsheet [1].

This scraping uses crawler and scraper for web scraping, where crawler is an AI algorithm which searches the data across different links based on a search keyword and scraper is a tool driven by crawler and extracts the data that is found by crawler from the website [2]. The design of scraper changes dynamically based on the scope and complexity of task or project its handling, so that the accuracy and quick response will not be affected [3].

B. Selenium

Selenium is a tool which is used for both testing and running headless browsers. It supports browsers like Chrome, Edge, Firefox, Internet Explorer. It can even simulate human actions on web browsers like searching, running on a server and many more. Actions that selenium can perform are automated testing, getting screenshots, getting cookies and many more through the "WebDriver" module. This module also allows us to run the unstructured html data obtained by scraping [4].

C. Celery

Celery is a job queue or a task queue which follows the technique of distributed message passing. It is an open-source asynchronous queue. It operates on real time events by scheduling the jobs. Task here is referred to as execution events. These tasks are executed concurrently on worker nodes (more than one). They use gevent or eventlet which enables multiprocessing. This celery is used to process millions of tasks every day in the production system. Tasks in celery can be executed synchronously or asynchronously [5].

D. Redis

Redis acts as an in-memory key value database. Remote Dictionary Server is referred to as redis. It's a memory data structure store. It's a distributed data structure. It acts as cache and message broker, with optional durability. Its supports various ADT like lists, maps, sets, strings, sets, bitmaps, sorted sets, streams, HyperLogLogs, bitmaps and many more [5].

IV. METHODOLOGY

The method this project follows to search for jobs is scraping. This project scrapes the data from different websites as mentioned above using the keyword that users enter in the search bar. This project extracts these websites which are headless browsers. So, they need a server to execute these headless browsers, which is provided by selenium, a python library.

The Figure.1 gives a basic idea of how our portal data traverses between different resources. The data from job portal is extracted and transformed for a linear usage which is then stored in the database. Server accesses the data from the database and handles the data to the portal when a user requests.

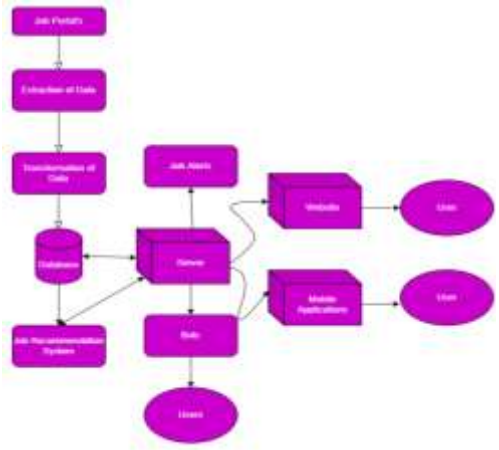

Fig. 1. Data flow of portal

This project provides a WhatsApp chat bot in the portal, which enables the communication between user and the server [6]. The chatbot communicates using a dialogue flow where the user query is sent as a json object as a request over the http protocol and receives the result as a fulfillment and intent, which is parsed into a json object and given back to the user as a reply [7].

The Figure.2 depicts the sequence diagram of WhatsApp chat bot working and its request flow from user to Twilio sandbox, from there to box handle micro services, and then to dialog flow, to scraping microservice.

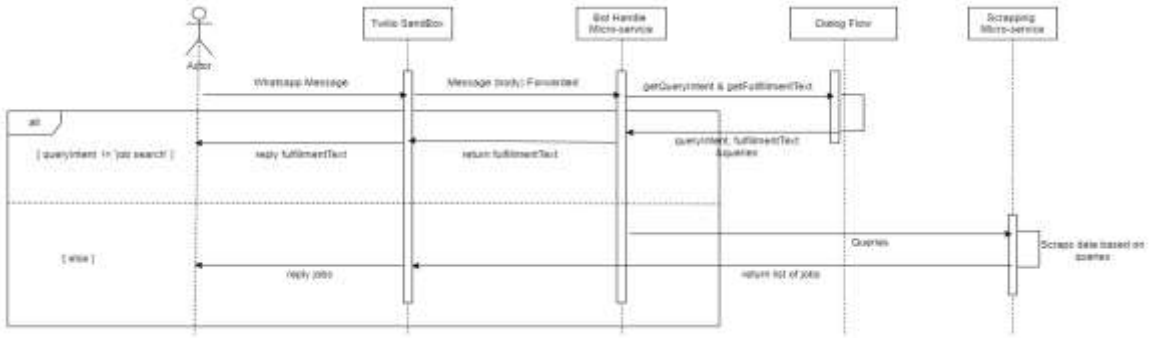

Fig. 2 Sequence Diagram of Chatbot Request

The Figure 3 depicts the system architecture of our portal which has users, and a database as major elements. It shows the workflow of our website which starts with a user requesting for a search. These requests which are sent by users will be handled by API calls.

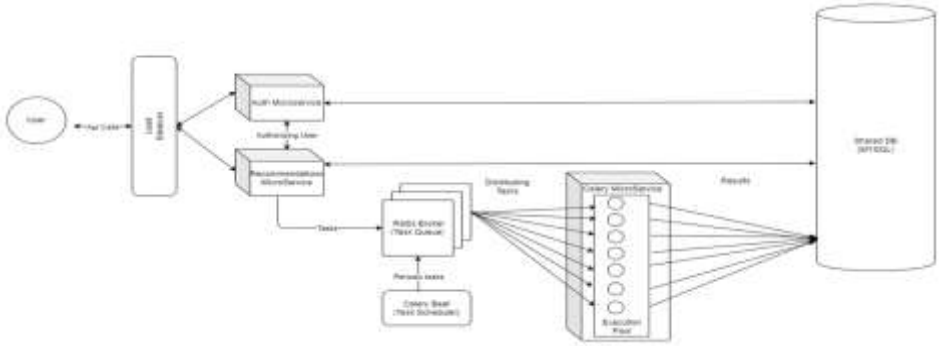

Fig.3 High level system architecture

The output from API calls is then redirected to the load balancers. Here load balancers act as a midway by distributing the clients/users request to servers based on their availability so that every request can be fulfilled with less waiting time.

This architecture is mostly microservice architecture where each task works parallelly ensuring a fast response for a user request. These microservices are broadly classified as Auth-microservices and Recommendations-microservices.

Auth-microservices fulfills the task of verifying the user identity and credentials across the database so that the user can have an isolated and secure account. It can also take details from users and register them by inserting the data into the database, which can be used for future reference and allows users to create an account.

Once the user's auth-microservice executes successfully then the recommendations-microservice gets activated. It's a complex microservice, where the tasks execute parallelly and multithreading comes into picture. For a portal, there can be many users who can request for a search at a time, from different locations, hence it becomes difficult for a single server to process each request without delay. To solve this problem this project has used celery microservice execution pool where each task executes simultaneously, it enables multithreading and the user requests from api calls are stored inside a redis queue where it manages the tasks to execute without letting the task die with starvation and preventing the deadlock. This even reduces the wait time of each request and response time of a server. Redis here acts as a broker between recommendations microservice and celery microservices by distributing the tasks.

For notification alerts for a user regarding updates, this project updates users every day at specific time intervals, to do this the tasks have to be executed. Manually creating these is a hectic task and hard to accomplish, so this project used Celery beat which creates a periodic task for a period of time and places it into the redis queue for execution. This can be achieved dynamically reducing manual power.

After every successful execution the results are stored into the database from which recommendations microservice access the data, so if a duplicate request is requested, instead of executing it from start it checks in the database first and fetches the data if present, else it sends the request to the redis and executes.

V. RESULTS AND DISCUSSION

Every website has three main divisions (frontend, backend, database). For frontend implementation, this project used ReactJs as tech stack, backend implementation is one through Django, selenium, celery microservice, celery beat, redis queue and for database, MySQL is used. Chatbots are implemented using Dialog Flow, Twilio Sandbox.

Django cannot handle more than one request at a time [8]. As the project is based on web scraping which is a time-consuming task and there can be many requests at a time. For the smooth functioning of the application which facilitates better response time and delay in propagation of requests celery is used as an assistance to Django. As the Project is solely based on web scraping which collects information from various job portals it is necessary to use a tool like celery which enhances user experience and helps with fast and accurate results. The project is not only confined to web application but also provides NLP bots which doesn't require any continuous monitoring in search of jobs. The bots can answer the user whenever he/she prompts about the job of his/her interest. Celery-beat is another important package which is used in this project to get email notifications, the main application of this package is to run periodically based on a time limit and get notified with new job notifications if there are any.

The obtained results of this project contain many modules which are structured as

1. *Sign-up and Login page*
2. *Preferences*
3. *Recommendations*
4. *Saved Job Profiles*
5. *Mail Notification Alerts*
6. *WhatsApp Chatbot*

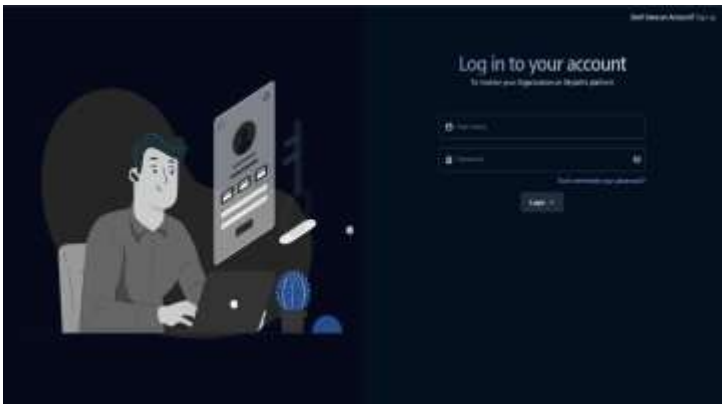

Fig.4 Login Page

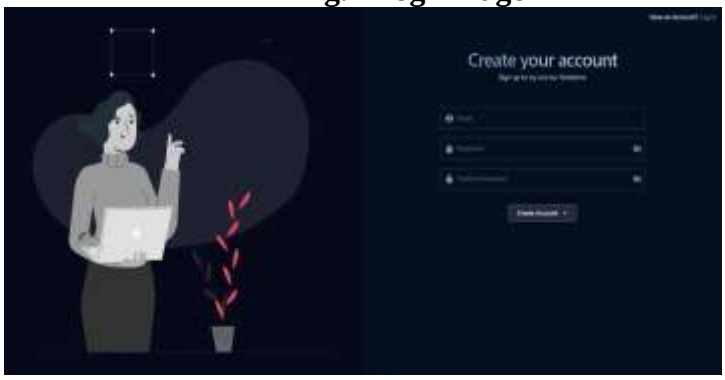

Fig.5 Signup page

Fig.6 Preferences

Fig.7 Recommendations

Fig.8 Saved Jobs

Fig.9 Notification Alerts

Fig.10 WhatsApp Chatbot

VI. CONCLUSION

So finally, to conclude this, our website reduces the jobseeker time in search for jobs and also alerts them through mails, provides chatbots to communicate and can get updates based on preferences set by the user. As the chatbot uses dialog flow, users' queries will never remain unanswered. All the updates that users require can be answered through WhatsApp if linked with chatbot or sent through emails if required. It's more like all in one for job posting websites. It also has multiprocessing, so users' wait time is never that long.

Our portal also allows users to save their jobs for future referencing which enables them to access whenever required without checking out for the company website every time about the status of the job. Based on the user's set, he/she can see the recommended jobs in the module, so users can cherry pick their job.

ACKNOWLEDGMENT

Our Asst.Professor Ch. Anusha has helped us with the research paper and guided us through our project. We are grateful to have you on our back when needed, it's like a continuous push through our tough times and your guidance brought us flying colors.

REFERENCES

[1] B. Zhao, "Web Scraping," in *Encyclopedia of Big Data*, Springer International Publishing, 2017, pp. 1–3. doi: 10.1007/978-3-319-32001-4_483-1.
[2] M. B. S and S. G. S, "Survey on Web scraping technology".
[3] S. B. Gulik and J. L. Choudhary, "Scraping of Job Portal," *International Journal of Advance Research, Ideas and Innovations in Technology*.
[4] "The Selenium Browser Automation Project | Selenium." https://www.selenium.dev/documentation/ (accessed May 27, 2022).
[5] "Using Redis — Celery 3.1.25 documentation." https://docs.celeryq.dev/en/3.1/getting-started/brokers/redis.html (accessed May 27, 2022).
[6] N. A. Khan and J. Albatein, "COVIBOT-An intelligent WhatsApp based advising bot for Covid-19," in *Proceedings of 2nd IEEE International Conference on Computational Intelligence and Knowledge Economy, ICCIKE 2021*, Mar. 2021, pp. 418–422. doi: 10.1109/ICCIKE51210.2021.9410801.
[7] "Dialogflow Documentation | Google Cloud." https://cloud.google.com/dialogflow/docs (accessed May 27, 2022).
[8] "Django documentation | Django documentation | Django." https://docs.djangoproject.com/en/4.0/ (accessed May 27, 2022).