Smart-Resume-Interpreter-And-Job-Alert-System

Parth Patel
pvpatel2@ncsu.edu
NC State University
North Carolina, USA

Neel Shah
nshah26@ncu.edu
NC State University
North Carolina, USA

Jay Shah
jshah7@ncsu.edu
NC State University
North Carolina, USA

Harshil Shah
hshah6@ncsu.edu
NC State University
North Carolina, USA

Ameya Tathavadkar
atathav@ncsu.edu
NC State University
North Carolina, USA

ABSTRACT

The idea of this document is to summarize the purpose, functionality and the overall design of the application built as a part of the SE Fall 2021 batch project. S.R.I.J.A.S i.e Smart-Resume-Interpreter-And-Job-Alert-System which provides suggestions to all the relevant jobs opportunities to the user which are available in the market. The technologies used and future scope is summarized in the following sections.

KEYWORDS

Job Tracking,Jobs,Resume,Job Listings, Linkedin,Glassdoor,scrapping

ACM Reference Format:
Parth Patel, Neel Shah, Jay Shah, Harshil Shah, and Ameya Tathavadkar. 2021. Smart-Resume-Interpreter-And-Job-Alert-System. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Applying for jobs can be very frustrating. All jobs have a separate alert system on their portal. Since, there are thousands of jobs,so it becomes impossible to keep the track of new job openings. The main aim of building the application is to provide people with new job openings that are available and apply for the same. The application will provide with the new job openings on Linkedin and Glassdoor that are relevant to the user.

2 APPLICATION OVERVIEW

As noted in the introduction, the application built will notify the user about the new job openings. When the user enters in the interface, he will enter the name, email id, job title to search for and the resume of the user. If the skills of the user matches with the requirements given in the job description, then links of the job postings will be sent to the user via email as scheduled.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Future Scope

The idea is to have more advanced features for the user such as:

(1) Allow more advanced filters
(2) Integrate the web portal with the login service
(3) Create a system to store user profiles and generate insights from it.
(4) Allow users to select previously uploaded resumes to preview
(5) Develop a dashboard to display user analytics

3 KERNEL BEST PRACTICES FOLLOWED

Because of the principles used in its implementation, the Linux Kernel is effective and is dominating other open source project development platforms. Every software project management cycle employs these best practices in order to optimize efficiency and thereby improve usage. The Linux Kernel development best practices are:

(1) Zero Internal Boundaries
(2) Short Release Cycle
(3) Distributed Development Model
(4) Consensus Oriented Model
(5) No Regression Rule

These Linux Kernel best practice are employed in the project SRIJAS following the rubric from project 1 grading rubric .These connections are explained in detailed.

3.1 Zero Internal Boundaries

Different developers work on each project, and each developer should have access to all aspects of the project. Everyone working on the project should have access to the tools and code. However, if a developer is unable to access any section of the project to review or develop code, the distributed model's efficiency suffers. As a result, any restrictions on resource access should be considered in order to achieve a good performance model. Zero Internal Boundaries indicate that all resources, tools, code, and other aspects of the project should be available to everyone working on it, with no restrictions or constraints. In this project, there is no boundary to access any part of the project anywhere to any developer. Each developer can access the code from any part of the project through the cross platform open-source tools. All developers of the project have the same accessibility to all the files in the platform. Since most languages were known by everyone in the team, we were easily able to communicate and also complete our own pieces.

3.2 Short Release Cycle

In a software development project, short release cycles are critical because they allow for the rapid distribution of small features to the user. Short releases indicate that the developers are always working to provide the greatest possible user experience. Long release cycles can put a lot of strain on developers to integrate large quantities of code, which might lead to failure. As a result, short release cycles ensure consistent release. Short releases provide short-term goals by precisely specifying the timeframe and assuring the developer that if a release is missed, the next one will be in a few months rather than a lengthy time. Short Releases allow developers to update the software's flow at any time, as well as add new technologies that may not have been accessible at the time of the original release. Because this project had a month's timeframe, releases were made as numerous push requests to the project.

3.3 Distributed Development Model

Each project must be developed by a team of developers who will review and improve the project's quality. The projects are divided into little pieces and assigned to multiple developers in a distributed architecture. Instead of one developer working on a project, this model allows each developer to work independently and propose system changes. The project will be developed in this manner, with thorough evaluations, debugging, and increased efficiency. The project's overall performance can be improved. The project 1 grading rubric mentions that the workload should be distributed among the group members to be developed. In this project, the work load is equally distributed among all the team developers which can be seen through the number of commits and issues handled by each contributor. The rubric also mentions that group meetings and having a communication channel can help achieve a distributed model.A whatsapp communication channel was created for the discussion of project updates. Weekly inperson meetings were conducted to discuss and assign tasks.Everyone was aware of the statuses and progress across all the divisions. So, a Distributed Development Model is employed for this project.

3.4 Consensus Oriented Model

Consensus Oriented Model means when a change to the project code is made, it is done with the approval of the project's majority contributors. This ensures that no feature or section of the code is compromised when an update is released, as well as assisting developers in verifying the correctness of the code to be introduced. This paradigm guarantees that the software will not fail owing to erroneous coding. During the development of this project, the software developers planned meetings to address open concerns before they were resolved. This discussion guarantees that the issue's solution is accepted by the other developers and that the code does not clash with any previously written code. At any given time, at least two contributors were assigned to double-check any code written for any outstanding issue before it was published to the Git repository. Github looks for discrepancies between several pushes and assigns someone to go over them to provide quality control. Many test cases have been applied to issues in order to ensure that the code is error-free.

3.5 No Regression Rule

Any project may contain bugs, which must be addressed from time to time. As a result, resolving these defects becomes necessary before the addition of more software code or certain internal adjustments. However, these changes should not degrade the software's current quality. The Linux Kernel development best practices' No Regression Rule states that any upgrade to the code or system should not break their system. Any upgrade should not degrade the quality of the product. The settings for any subsequent kernels should be the same as the main kernel. In this project, corresponding to the guidelines in the project 1 grading rubric, all of the version updates are added in the documentation.So, anyone can check the documentation and get to know about the updates. The documentation includes revision control of source control, versions, key features, new releases(if any), etc. From this documentation, it is clear to find out about any update that improves or reduces existing quality. From this, we can see that the this project has employed the No regression Rule and so any user can check details of any updates and its impacts.