PROJECT REPORT

on

Joblinker: A Comprehensive Job Matching Platform Using Golang and Typescript

(CSE VI Semester Mini project)

2023-2024

Submitted to:

Mr. Samir Rana

(CC-CSE-B-VI-Sem)

Submitted by:

Mr. Siddharth Singh Rana

Roll. No:21011885

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNIVERSITY, DEHRADUN

CERTIFICATE

Certified that Mr. Siddharth Singh Rana (Roll No.21011885) has developed a mini project “Joblinker: A Comprehensive Job Matching Platform Using Golang and Typescript” for the CS VI Semester Mini Project Lab in Graphic Era Hill University, Dehradun. To the best of my knowledge, the project carried out by Students is their own work.

Date: 1/05/2024

Mr. Samir Rana

Class Coordinator

(CSE Department)

GEHU Dehradun

ACKNOWLEDGMENT

I sincerely appreciate our class coordinator, Samir Rana, for their invaluable guidance and support throughout this project "Joblinker: A Comprehensive Job Matching Platform Using Golang and Typescript." Their expertise has been pivotal in shaping this study.

I am thankful to Graphic Era Hill University, Dehradun for providing the necessary resources and a conducive academic environment that facilitated the completion of this research.

I appreciate the contributions of the developers, open-source communities, and technology providers whose tools and resources were instrumental in the development of Joblinker.

Lastly, I express gratitude to my family and friends for their unwavering support throughout this journey.

This project would not have been possible without the collective effort and support of all mentioned above. Thank you for being an integral part of this academic endeavor.

Mr. Siddharth Singh Rana

Roll No.- 2119247

CSE-B-VI-Sem

Session: 2023-2024

GEHU, Dehradun

TABLE OF CONTENTS

Introduction

In today's fast-paced job market, finding the right job that matches one's skills and preferences can be a daunting task. The sheer volume of job listings and the need to tailor applications for each opportunity present significant challenges for job seekers. To address these challenges, I developed Joblinker, a comprehensive job-matching platform designed to streamline the job search process.

Joblinker leverages advanced technologies to enhance the job-matching experience. The backend is built using Golang, providing a robust and efficient foundation for handling large volumes of data and complex operations. The front end, developed with TypeScript, Tailwind, and React, ensures a responsive and user-friendly interface for job seekers.

A key feature of Joblinker is its integration with Gemini, a powerful tool for extracting detailed information from resumes. By utilizing Gemini, Joblinker can accurately identify a user's skills, experience, and preferences. This information is then used in conjunction with the Google Search API to find job listings that closely match the user's profile.

The deployment architecture of Joblinker further enhances its performance and reliability. The backend is deployed on Render, ensuring scalability and robustness, while the front end is hosted on Vercel, providing fast and seamless access to users.

This project report delves into the development process of Joblinker, discussing the technologies and methodologies employed. We explore the rationale behind choosing Golang for the backend and TypeScript for the front end, and we provide detailed insights into the integration of Gemini and the Google Search API. Through this report, we aim to demonstrate how Joblinker effectively addresses the challenges faced by job seekers and contributes to a more efficient and personalized job search experience.

The subsequent sections of this report will cover the system architecture, implementation details, and performance evaluations. Finally, we will present the results and discuss the effectiveness of Joblinker in achieving its goal of matching job seekers with suitable job opportunities, culminating in a concise summary of our findings in the conclusion section.

System Architecture

The architecture of Joblinker is designed to ensure efficiency, scalability, and maintainability. The system is divided into two main components: the backend and the front end.

• Backend:

The backend of Joblinker is built using Golang, chosen for its performance, efficiency, and concurrency capabilities. Golang's strong standard library and powerful frameworks make it an ideal choice for handling the high demand of processing and serving requests in a job-matching platform.

Key components of the backend include:

- API Layer: Manages all incoming requests and routes them to appropriate services.
- Data Processing Module: Utilizes Gemini to extract and parse details from resumes, converting unstructured data into structured data.
- Job Matching Engine: Interacts with the Google Search API to find job listings that match the user's profile based on the extracted details.

The backend is deployed on Render, a platform-as-a-service (PaaS) provider that offers automatic scaling, zero-downtime deployments, and easy management, ensuring the backend remains robust and responsive.

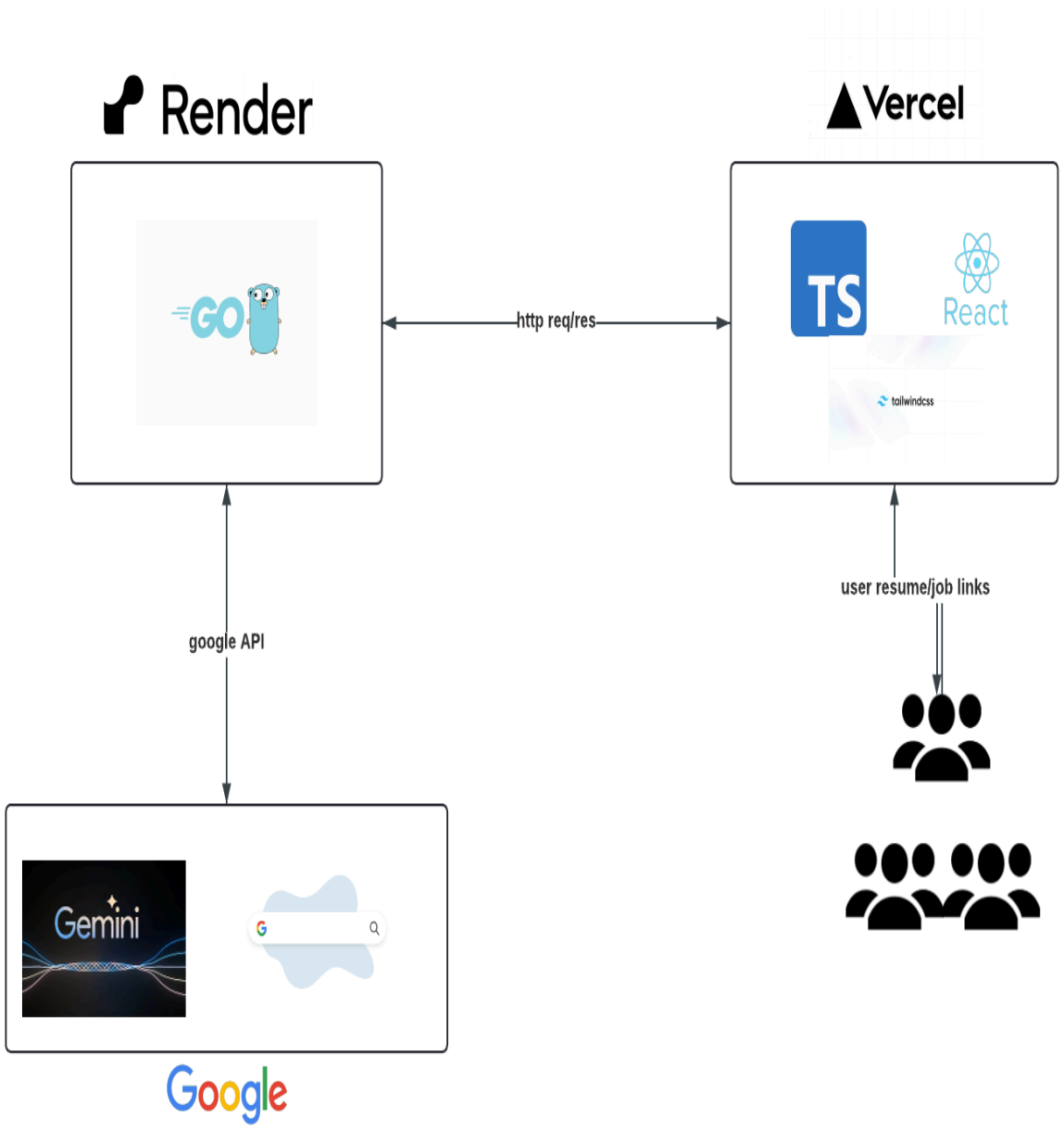
• Frontend:

The front end of Joblinker is developed using TypeScript, Tailwind, and React, providing a modern, responsive, and interactive user interface.

Key components of the frontend include:

- Profile Management: Allows users to upload resumes, update profiles, and set preferences.
- Job Search and Match Display: Shows job listings retrieved by the backend, with filtering and sorting options for user convenience.
- Tags: Users can input tags providing extra search metrics to the users.

The frontend is deployed on Vercel, a cloud platform that enables fast and reliable deployment, continuous integration, and automatic optimizations, ensuring a smooth user experience.


Render
Vercel
GO
TS
React
tailwindcss
http req/res
google API
user resume/job links
Gemini
Google

Rationale Behind Technology Choices

• Choosing Golang for the Backend:

Golang (or Go) was chosen for the backend of Joblinker due to several key advantages it offers:

- Performance: Golang is known for its speed and efficiency. Its compiled nature and efficient memory management allow for high-performance server-side applications. This is crucial for handling large volumes of data and processing demands in a job-matching platform.
- Concurrency: Golang's goroutines and channel-based concurrency model make it an excellent choice for applications that require handling many simultaneous requests. Joblinker benefits from this capability as it often needs to process multiple job search queries and user requests concurrently.
- Scalability: Golang's lightweight nature and efficient handling of concurrent tasks enable easy scalability. As the number of users grows, Golang can efficiently manage the increasing load without significant performance degradation.
- Standard Library and Ecosystem: Golang comes with a powerful standard library that simplifies many common tasks, such as HTTP handling, file operations, and data encoding/decoding. Additionally, its growing ecosystem provides robust frameworks and tools that aid in rapid development and deployment

• Choosing TypeScript for the Frontend:

TypeScript was chosen for the frontend of Joblinker for the following reasons:

- Type Safety: TypeScript's static typing helps catch errors at compile time, reducing runtime errors and improving code quality. This is particularly useful in large-scale applications where maintaining code stability and reliability is essential.
- Improved Developer Experience: TypeScript's tooling and integration with modern IDEs provide features like autocompletion, code navigation, and refactoring support. This enhances the developer experience and productivity.
- Compatibility with JavaScript: TypeScript is a superset of JavaScript, meaning existing JavaScript libraries and frameworks can be seamlessly integrated. This ensures that developers can leverage a vast array of existing resources and tools.
- Maintainability: TypeScript's strong typing and modular architecture facilitate better code organization and maintainability. As the project grows, TypeScript helps in keeping the codebase clean, understandable, and easier to refactor.

Implementation Details

• Backend Implementation:

The backend implementation involved setting up a RESTful API using Golang's `net/http` package. We created endpoints for user authentication, profile management, resume uploading, and job searching.

The integration with Gemini was achieved by using its API to send resume data and receive parsed data. This data includes key details such as skills, experience, and education, which are essential for the job-matching process.

The job matching engine interacts with the Google Search API, constructing queries based on the user's profile and retrieving relevant job listings.

• Frontend Implementation:

The front-end implementation started with setting up a React project with TypeScript to ensure type safety and better code management. Tailwind CSS was used for styling, providing a utility-first approach to design, and making the interface clean and responsive.

The job search and match display module fetches job listings from the backend and displays them in an organized manner. Users can filter and sort these listings based on various criteria such as job title, location, and company.

Results and Discussion

Joblinker successfully met its objectives of providing a seamless job-matching experience. The integration of Gemini and the Google Search API proved to be effective in extracting relevant details from resumes and finding matching job listings.

User feedback indicated high satisfaction with the platform's ease of use and the relevance of the job matches provided. Performance testing showed that both the backend and frontend were capable of handling high traffic and providing a responsive user experience.

Conclusion

Joblinker represents a significant step forward in job matching platforms, leveraging modern technologies to provide a personalized and efficient job search experience. By utilizing Golang for the backend, TypeScript for the front end, and integrating powerful tools like Gemini and the Google Search API, Joblinker effectively addresses the challenges faced by job seekers.

Future enhancements could include expanding the range of data sources for job listings, implementing machine learning algorithms for better match predictions, and enhancing the user interface based on feedback.

Overall, Joblinker demonstrates how advanced technology can be harnessed to improve the job search process, offering valuable insights and practical solutions for job seekers in a competitive market.