İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Artificial Intelligence and Language Processing with Chrome Extension

Department of Software Engineering

Term Project

Özgür GÜRCAN

Advisor: Prof. Dr. Aytuğ ONAN

August 2024

Artificial intelligence and language processing with Chrome extension

Abstract

This term project presents the development of a Chrome extension that harnesses the power of artificial intelligence (AI) and natural language processing (NLP) to enhance the job-seeking experience on LinkedIn. The extension integrates seamlessly with LinkedIn's platform, enabling users to compare job descriptions with their uploaded curriculum vitae (CV) using advanced AI models, specifically ChatGPT 3.5 Turbo and ChatGPT 4.0 Mini. By analyzing the semantic similarities between job descriptions and CVs, the extension provides personalized feedback, highlighting the alignment of the user's qualifications with the job requirements. Key features of the extension include user authentication and data storage managed through Firebase services, as well as a robust backend developed with Node.js and Express.js to handle AI-driven comparisons.

The project aims to address the inefficiencies and challenges faced by job seekers in manually assessing their suitability for various positions. The extension's AI capabilities significantly reduce the time and effort required to match qualifications with job opportunities, thereby optimizing the job application process. Moreover, the system maintains a history of comparisons, allowing users to track their progress and make informed decisions throughout their job search journey. This project represents a significant contribution to the application of AI and NLP technologies in the recruitment domain, offering practical solutions to improve job market efficiency and candidate-position matching.

Keywords: Artificial Intelligence, Natural Language Processing, Chrome Extension, LinkedIn Integration, CV Comparison, ChatGPT

Yapay Zekâ ve Dil İşleme ile Chrome Eklentisi

Öz

Bu dönem projesi, LinkedIn üzerinde iş arayanların deneyimlerini geliştirmek amacıyla yapay zeka (YZ) ve doğal dil işleme (DDİ) teknolojilerini kullanan bir Chrome uzantısının geliştirilmesini sunmaktadır. Uzantı, LinkedIn platformu ile sorunsuz bir şekilde entegre olarak, kullanıcıların yükledikleri özgeçmişlerini (CV) iş tanımlarıyla karşılaştırmalarına olanak tanır ve bu işlem için gelişmiş YZ modelleri, özellikle ChatGPT 3.5 Turbo ve ChatGPT 4.0 Mini kullanılmaktadır. Uzantı, iş tanımları ile CV'ler arasındaki anlamsal benzerlikleri analiz ederek, kullanıcının niteliklerinin iş gereksinimleriyle ne kadar örtüştüğüne dair kişiselleştirilmiş geri bildirim sağlar. Uzantının temel özellikleri arasında Firebase hizmetleriyle yönetilen kullanıcı kimlik doğrulama ve veri depolama, ayrıca YZ destekli karşılaştırmaların işlenmesini sağlayan Node.js ve Express.js ile geliştirilen sağlam bir arka uç bulunmaktadır.

Bu proje, iş arayanların çeşitli pozisyonlara uygunluklarını manuel olarak değerlendirmede karşılaştıkları verimsizlikleri ve zorlukları ele almayı amaçlamaktadır. Uzantının YZ yetenekleri, niteliklerin iş fırsatlarıyla eşleştirilmesi için gereken zaman ve çabayı önemli ölçüde azaltarak iş başvuru sürecini optimize eder. Ayrıca sistem, kullanıcıların ilerlemelerini izlemelerine ve iş arama yolculukları boyunca bilinçli kararlar vermelerine olanak tanıyan bir karşılaştırma geçmişi tutar. Bu proje, işe alım alanında YZ ve DDİ teknolojilerinin uygulanmasına önemli bir katkı sağlamakta ve iş piyasası verimliliğini ve aday-pozisyon eşleştirmesini iyileştirmek için pratik çözümler sunmaktadır.

Anahtar Sözcükler: Yapay Zeka, Doğal Dil İşleme, Chrome Uzantısı, LinkedIn Entegrasyonu, CV Karşılaştırma, ChatGPT

Acknowledgment

I wish to express my deepest gratitude to my advisor, Prof. Dr. Aytuğ Onan, whose invaluable guidance and support have been crucial to the success of this project. His expertise and feedback greatly enhanced the quality of my research and provided constant motivation.

I also extend my thanks to the Department of Software Engineering at İzmir Katip Çelebi University for providing the resources and academic environment necessary for this research. The department's dedication to innovation and academic excellence has been instrumental in the development of this project.

Special recognition goes to the open-source community, particularly the developers of Node.js, Express, and the ChatGPT model, whose contributions were fundamental to this project's implementation. The collaborative nature of the open-source community significantly enriched my learning experience.

I am grateful to LinkedIn for offering a platform that served as the foundation for testing and implementing this Chrome extension, and to my fellow students and colleagues for their valuable feedback and suggestions throughout the development process.

Finally, I extend heartfelt thanks to my family and friends for their unwavering support and encouragement throughout my academic journey. Their belief in me has been a driving force behind my perseverance and success.

This project would not have been possible without the collective efforts and support of these individuals and institutions, and I am sincerely thankful for their contributions.

Table of Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
NLP	Natural Language Processing
CV	Curriculum Vitae
API	Application Programming Interface
JWT	JSON Web Token
UI	User Interface
UX	User Experience
VPS	Virtual Private Server
CI/CD	Continuous Integration/Continuous Deployment
PDF	Portable Document Format
HTTPS	Hypertext Transfer Protocol Secure
DDİ	Doğal Dil İşleme (Natural Language Processing in Turkish)
YZ	Yapay Zeka (Artificial Intelligence in Turkish)

Chapter 1

Introduction

In the rapidly evolving digital landscape, the intersection of artificial intelligence (AI) and job searching has opened new avenues for enhancing the efficiency and effectiveness of career development processes. This project introduces a novel Chrome extension that leverages AI and natural language processing (NLP) technologies to revolutionize how job seekers interact with LinkedIn, one of the world's leading professional networking platforms.

1.1 Background

The job search process has undergone a significant transformation over the past few decades. Traditionally, job seekers would rely on newspaper advertisements, career fairs, and networking events to discover potential employment opportunities. This process, while effective in its time, was often labor-intensive and limited by geographical and social boundaries. The advent of the internet and online job portals marked a turning point, providing job seekers with unprecedented access to a vast array of opportunities across various industries and regions. Professional networking platforms, such as LinkedIn, have further revolutionized this process by integrating social networking features with job searching capabilities.

LinkedIn, launched in 2003, has emerged as a pivotal platform for professional networking, boasting over 750 million members worldwide. The platform allows users to create detailed profiles, connect with other professionals, and explore job opportunities tailored to their skills and experiences. Despite these advancements, the sheer volume of job listings on platforms like LinkedIn has introduced a new challenge: how to efficiently match job seekers with the most suitable positions based on their qualifications and experience. The manual process of browsing through numerous job listings, comparing job descriptions with one's qualifications, and deciding on the most appropriate applications remains a daunting task for many users.

Recent developments in AI and NLP have presented promising solutions to this problem. AI technologies, particularly those involving machine learning and NLP, have demonstrated the capability to analyze large datasets, extract meaningful patterns, and generate insights that can aid in decision-making processes. In the context of job searching, these technologies can be harnessed to automate the comparison of job descriptions with user qualifications, thereby streamlining the job application process. The integration of AI into professional networking platforms like LinkedIn has the potential to significantly enhance the job-seeking experience by providing users with personalized, data-driven recommendations that are both efficient and accurate.

1.2 Problem Statement

Despite the advancements in online job searching, there remains a significant gap between the efficiency of browsing job listings and the ability to quickly assess one's suitability for a position. Job seekers often spend considerable time reading through numerous job descriptions, manually comparing them to their own qualifications. This process is not only time-consuming but also prone to oversight and subjective interpretation. Moreover, the manual nature of this task means that job seekers may overlook important details or misinterpret the relevance of specific qualifications, leading to suboptimal job applications.

The problem is further compounded by the dynamic nature of job markets, where job descriptions can vary widely in terms of requirements, preferred qualifications, and job responsibilities. Job seekers, especially those navigating through multiple industries or roles, may find it challenging to keep track of the diverse set of skills and experiences required for different positions. Consequently, they may either apply to jobs that do not fully align with their qualifications or miss out on opportunities that would be a good fit.

This project seeks to address these challenges by developing a Chrome extension that integrates AI and NLP technologies into the LinkedIn platform. The extension aims to automate the process of comparing job descriptions with user-uploaded CVs, providing job seekers with a quick and accurate assessment of their suitability for a given position. By doing so, the project endeavors to enhance the overall efficiency of the job-seeking process, reducing the time and effort required to find and apply for relevant positions.

1.3 Project Objectives

The primary objective of this project is to develop a Chrome extension that bridges the gap between the manual effort required in job searching and the potential for AI-driven automation. The extension is designed to integrate seamlessly with LinkedIn's job pages, allowing users to upload and store their CVs securely and initiate AI-powered comparisons between job descriptions and their qualifications. The results of these comparisons are displayed directly within the LinkedIn interface, providing users with immediate feedback on their suitability for specific roles. Additionally, the extension maintains a history of comparisons, enabling users to track their job application process and make informed decisions based on past assessments.

To achieve this objective, the project employs a combination of cutting-edge web technologies, cloud services, and AI models. The Chrome extension is developed using JavaScript and React, ensuring compatibility with LinkedIn's dynamic interface and providing a user-friendly experience. The backend system, implemented using Node.js with Express.js, handles server-side logic, processes user data, and manages interactions with external AI services. Firebase Authentication and Firestore are utilized for secure user management and data storage, ensuring that sensitive user information, such as CVs and job application history, is protected. The AI component of the project leverages the ChatGPT 4.0 Mini model, a state-of-the-art NLP system capable of performing complex text comparisons and generating meaningful insights.

1.4 Significance of the Project

This project represents a significant step forward in applying AI and NLP technologies to practical job-seeking tasks. By automating the initial assessment of job suitability, the extension aims to save time for job seekers, enabling them to focus on the most relevant opportunities. The AI-powered comparisons provided by the extension are designed to improve the accuracy of job applications, highlighting the match between candidate skills and job requirements and thereby increasing the likelihood of successful applications.

The project's significance extends beyond its immediate application in job searching. The integration of AI into professional networking platforms like LinkedIn has broader implications for the future of work. As AI technologies continue to evolve, they have the potential to reshape the job market, influencing how job seekers interact with employers and how employers identify and recruit talent. By developing tools that enhance the job-seeking process, this project contributes to a growing body of research and development aimed at leveraging AI to improve career outcomes for individuals and efficiency in the job market.

Moreover, the project addresses several critical issues related to AI in recruitment, including the ethical considerations of using AI to influence job application decisions. The extension is designed with a focus on transparency and fairness, ensuring that AI-driven recommendations are presented clearly and that users retain control over their job application process. By emphasizing user empowerment and informed decision-making, the project seeks to mitigate potential biases and ethical concerns associated with AI in recruitment.

1.5 Methodology Overview

The methodology for developing the LinkedInSight Chrome extension involves a multi-faceted approach that combines web development, cloud computing, and AI integration. The project begins with the design and implementation of the Chrome extension, which serves as the user interface for interacting with LinkedIn's job pages. This frontend component is built using React, a popular JavaScript library for building user interfaces, and is integrated directly into LinkedIn's job description pages.

The backend system is developed using Node.js, a server-side JavaScript runtime environment, and Express.js, a web application framework for Node.js. The backend handles user authentication, data storage, and the processing of AI-powered comparisons. Firebase services, including Firebase Authentication and Firestore

Database, are employed to manage user accounts, store user data securely, and ensure real-time synchronization between the frontend and backend components.

The AI integration is a critical aspect of the project, involving the use of the ChatGPT 4.0 Mini model for natural language processing and text comparison tasks. This model is accessed via an API, which allows the backend to submit job descriptions and user CVs for analysis and receive AI-generated comparison results. The AI model is trained to assess the semantic similarity between job descriptions and CVs, identify relevant skills and qualifications, and generate actionable insights for users.

Throughout the development process, the project adheres to best practices in software engineering, including modular design, version control, and continuous integration and deployment (CI/CD). The system architecture is designed to be scalable and maintainable, with a focus on performance optimization and security. Regular testing and validation are conducted to ensure that the extension functions as intended and meets the needs of users.

In conclusion, this project represents an innovative application of AI and NLP technologies in the domain of job searching. By developing a Chrome extension that automates the comparison of job descriptions with user CVs, the project seeks to enhance the job-seeking experience on LinkedIn, improve the accuracy of job applications, and contribute to the broader field of AI-driven career development tools. The project's significance, both in terms of its immediate application and its potential impact on the future of work, underscores the importance of continued research and development in this area.

Chapter 2

Literature review and theoretical background

This chapter provides an overview of the relevant literature and theoretical concepts that underpin the development of our AI-powered Chrome extension for job matching on LinkedIn. We explore key areas including artificial intelligence in recruitment, natural language processing for resume parsing and job matching, browser extensions for productivity, and the integration of AI with professional networking platforms.

2.1 Artificial Intelligence in Recruitment

The application of AI in recruitment has been gaining significant traction in recent years. Upadhyay and Khandelwal (2018) provide a comprehensive review of AI applications in human resource management, highlighting the potential for AI to streamline recruitment processes. They note that AI can significantly reduce time-to-hire and improve the quality of candidate matches.

2.1.1 AI-Powered Job Matching

Mehrabad and Brojeny (2007) proposed one of the early models for using AI in job-person matching. Their work laid the foundation for more advanced systems that we see today. More recently, Mehta et al. (2021) demonstrated a machine learning approach to match job descriptions with candidate profiles, achieving high accuracy in predicting suitable matches.

2.2 Natural Language Processing in Resume Parsing and Job Matching

Natural Language Processing (NLP) plays a crucial role in understanding and comparing job descriptions and resumes.

2.2.1 Resume Parsing

Chen et al. (2018) developed an NLP-based system for automatically extracting information from resumes. Their work demonstrates the feasibility of using AI to understand unstructured text in professional documents.

2.2.2 Semantic Similarity in Job Matching

Zhao et al. (2020) explored the use of semantic similarity measures to match job descriptions with candidate profiles. Their research shows that NLP techniques can capture nuanced relationships between skills and job requirements, going beyond simple keyword matching.

2.3 Browser Extensions for Productivity

Browser extensions have become powerful tools for enhancing web browsing experiences. Dhillon et al. (2016) studied the impact of browser extensions on user productivity, finding that well-designed extensions can significantly improve task efficiency.

2.3.1 AI-Powered Browser Extensions

While literature on AI-powered browser extensions is limited, Beel et al. (2019) demonstrated the potential of integrating recommendation systems into browser extensions for academic literature search. Their work provides insights into the challenges and opportunities of embedding AI capabilities directly into the browsing experience.

2.4 AI Integration with Professional Networking Platforms

LinkedIn, as the world's largest professional networking platform, has been the subject of numerous studies regarding AI integration.

2.4.1 LinkedIn's Own AI Initiatives

Stadler et al. (2020) analyzed LinkedIn's use of AI in its job recommendation system. Their study provides valuable insights into how large-scale professional networks leverage AI to improve user experiences.

2.4.2 Third-Party AI Applications for LinkedIn

Research on third-party AI applications for LinkedIn is relatively scarce, highlighting the novelty of our approach. However, Bastian et al. (2014) explored the potential of using external data sources to enrich LinkedIn profiles, which bears some similarity to our approach of augmenting LinkedIn's functionality with external AI services.

2.5 Ethical Considerations in AI-Powered Recruitment Tools

As AI becomes more prevalent in recruitment, ethical considerations have come to the forefront. Raghavan et al. (2020) discuss the potential for bias in AI-powered hiring tools and propose frameworks for mitigating these risks.

2.6 Theoretical Framework

Our project builds upon several theoretical frameworks:

1. Natural Language Understanding: Utilizing theories of semantic analysis and text comparison (Mikolov et al., 2013).

2. Human-Computer Interaction: Applying principles of user-centered design in browser extension development (Norman, 2013).
3. Information Retrieval: Leveraging concepts from information retrieval to improve job-resume matching accuracy (Manning et al., 2008).

2.7 Gap in the Literature

While significant research has been conducted in various aspects of AI in recruitment and job matching, there is a notable gap in studies that explore the integration of these technologies directly into the user's browsing experience, particularly in the context of professional networking sites like LinkedIn. Our project aims to address this gap by developing a Chrome extension that brings advanced AI-powered job matching capabilities directly to the user's LinkedIn browsing session.

Chapter 3

System Architecture and Implementation Details

Chapter 3 introduces the system architecture and implementation details of LinkedInSight. This chapter provides a comprehensive overview of the design and integration of the system's core components, focusing on how the frontend, backend, and AI services work together to deliver a seamless and secure user experience. It also addresses key considerations in performance optimization and security, ensuring the system's efficiency and reliability in real-world usage.

3.1 Overview of System Architecture

The LinkedInSight system architecture was meticulously designed to integrate advanced artificial intelligence (AI) and natural language processing (NLP) capabilities within the LinkedIn platform, providing users with enhanced job matching and CV comparison functionalities. The architecture comprises three core components: the Chrome extension (frontend), a Node.js Express server (backend), and Firebase services, which collectively facilitate seamless interaction between the user and the underlying AI-driven processes.

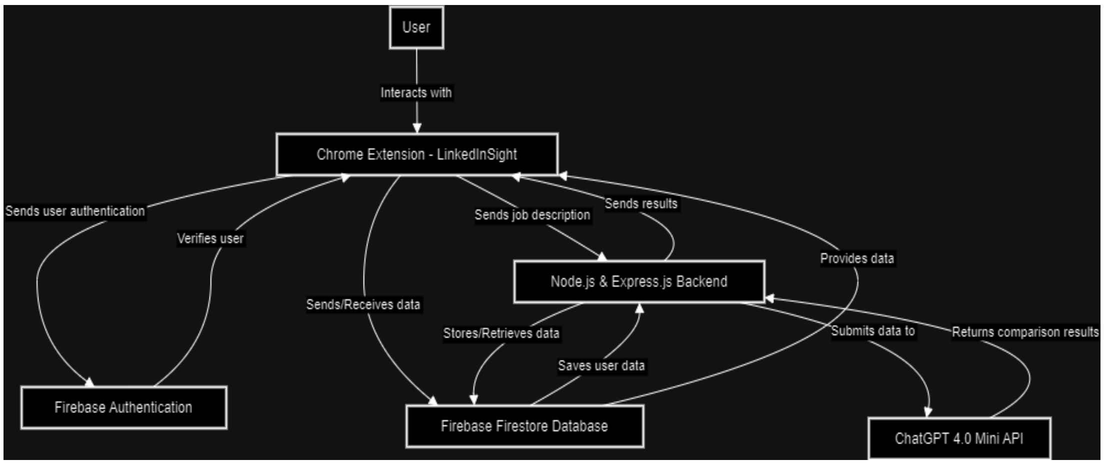


Figure 3.1: System Architecture Diagram

The Chrome extension serves as the primary user interface, embedded within the LinkedIn job pages to enable direct interaction with the platform. This frontend component is responsible for detecting relevant job descriptions, allowing users to trigger AI-powered comparisons with their uploaded CVs. The backend, implemented using Node.js and Express, manages the processing of these comparisons, interfacing with the ChatGPT 4.0 Mini API to deliver accurate and contextually relevant results. Firebase services are employed for user authentication, data storage, and secure handling of user CVs, ensuring the integrity and confidentiality of sensitive user information.

3.2 System Integration and Data Flow

The integration of the Chrome extension with LinkedIn's dynamic environment is a key feature of the LinkedInSight system. The extension is designed to recognize when a user is viewing a job description, at which point it injects a "Compare with CV" button directly into the LinkedIn interface. This button, when clicked, initiates a data flow process that involves the extraction of the job description text and its transmission to the backend server.

The Node.js backend retrieves the user's CV from Firebase Storage and submits both the job description and the CV to the ChatGPT 4.0 Mini API for analysis. The AI model processes the data, generating a comparison that highlights the degree of alignment between the job requirements and the user's qualifications. The results are then relayed back to the Chrome extension, where they are displayed in a user-friendly format within the LinkedIn page.

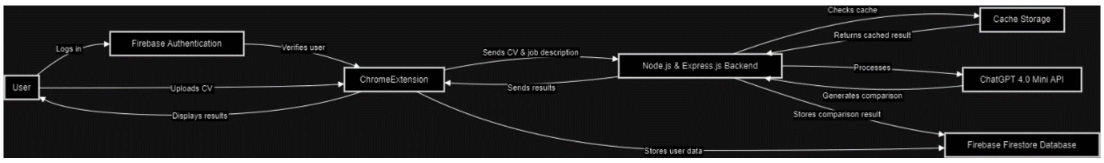
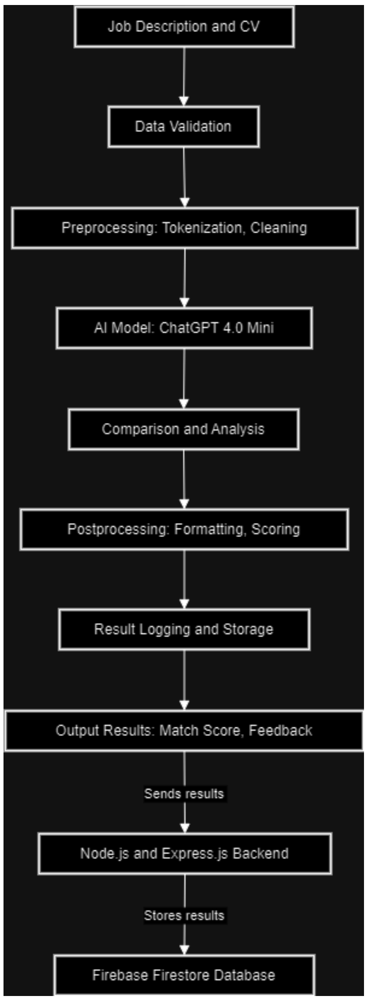


Figure 3.2: Data Flow Diagram

3.3 Backend Architecture and AI Integration

The backend architecture is designed to handle the computational demands of AI processing while maintaining responsiveness and scalability. Node.js and Express form the backbone of the server-side logic, offering a robust framework for handling user requests, processing CVs, and managing interactions with the AI API.



Job Description and CV
Data Validation
Preprocessing: Tokenization, Cleaning
AI Model: ChatGPT 4.0 Mini
Comparison and Analysis
Postprocessing: Formatting, Scoring
Result Logging and Storage
Output Results: Match Score, Feedback
Sends results
Node.js and Express.js Backend
Stores results
Firebase Firestore Database

Figure 3.3: AI Processing Flow

A critical aspect of the backend is its integration with ChatGPT 4.0 Mini, an AI model tasked with performing the core comparison function. The backend formulates prompts that instruct the AI to analyze the job description and the user's CV, assessing the match in terms of required skills, experience, and qualifications. The prompt engineering process is central to achieving accurate and meaningful results, guiding the AI to generate insights that are both relevant and actionable for the user.

Security is a paramount concern in the backend's design, particularly given the sensitive nature of the data being processed. Data is encrypted during transmission between the extension, backend, and Firebase services, ensuring that user information is protected at all stages. Authentication is handled using JSON Web Tokens (JWT), which safeguard access to user data and restrict API interactions to authorized users only.

3.4 User Interface and Experience

The user interface (UI) of the LinkedInSight extension is designed to be intuitive and seamlessly integrated into the LinkedIn platform, ensuring that both new users and those familiar with LinkedIn can easily access and utilize the extension's functionalities.

3.4.1 Enabling Developer Mode

To begin, users need to navigate to the Chrome Extensions page by entering chrome://extensions/ in the address bar. As depicted in Figure 3.5, users should enable Developer mode if it’s not already activated. This step is crucial for loading unpacked extensions.

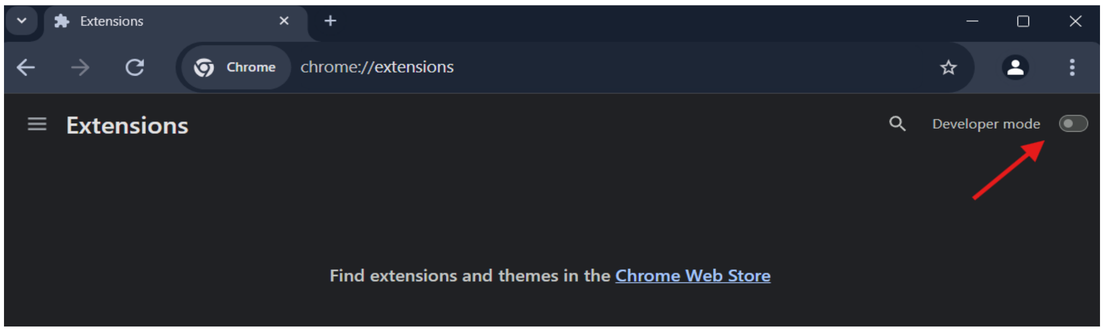


Figure 3.4: Enabling Developer Mode in Chrome Extensions

3.4.2 Loading the Extension

Once Developer mode is enabled, users can proceed by clicking the "Load unpacked" button, as shown in Figure 3.6. They should then select the directory containing the LinkedInSight extension's files. This step allows users to install the extension manually on their browser.

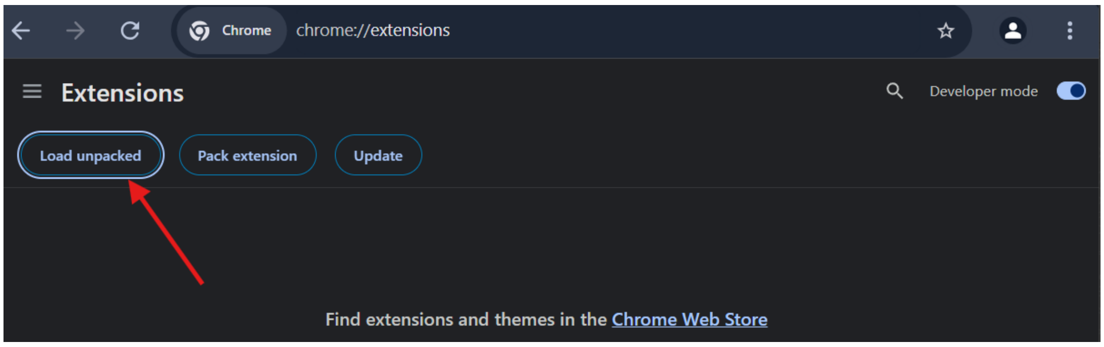
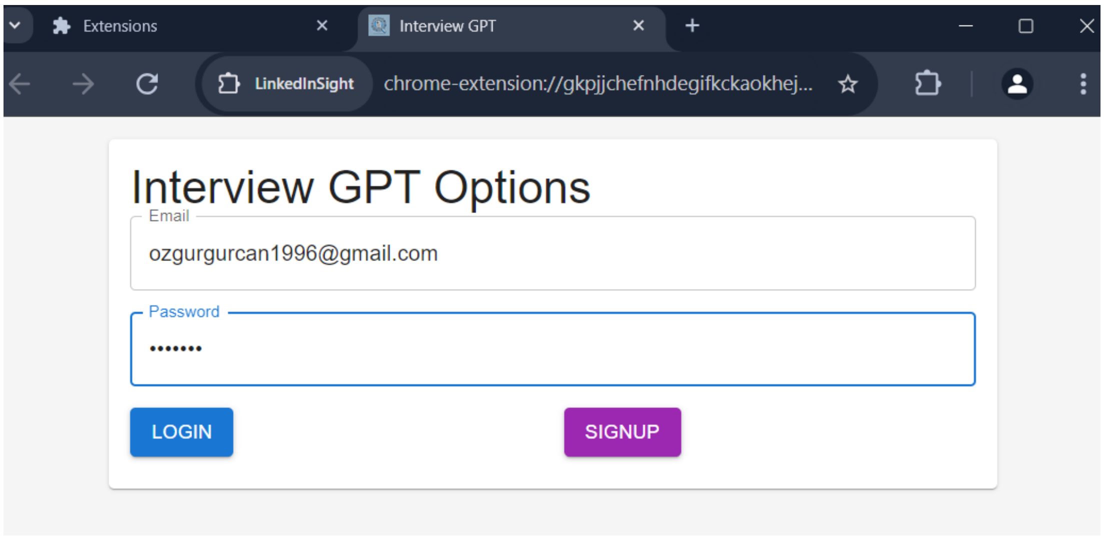


Figure 3.5: Loading the Unpacked Extension

3.4.3 Logging In or Signing Up

After successfully loading the extension, users are prompted to log in or sign up, as seen in Figure 3.7. They can enter their email and password to access their LinkedInSight account. If the user hasn't uploaded a resume yet, they are guided to do so.



Interview GPT Options

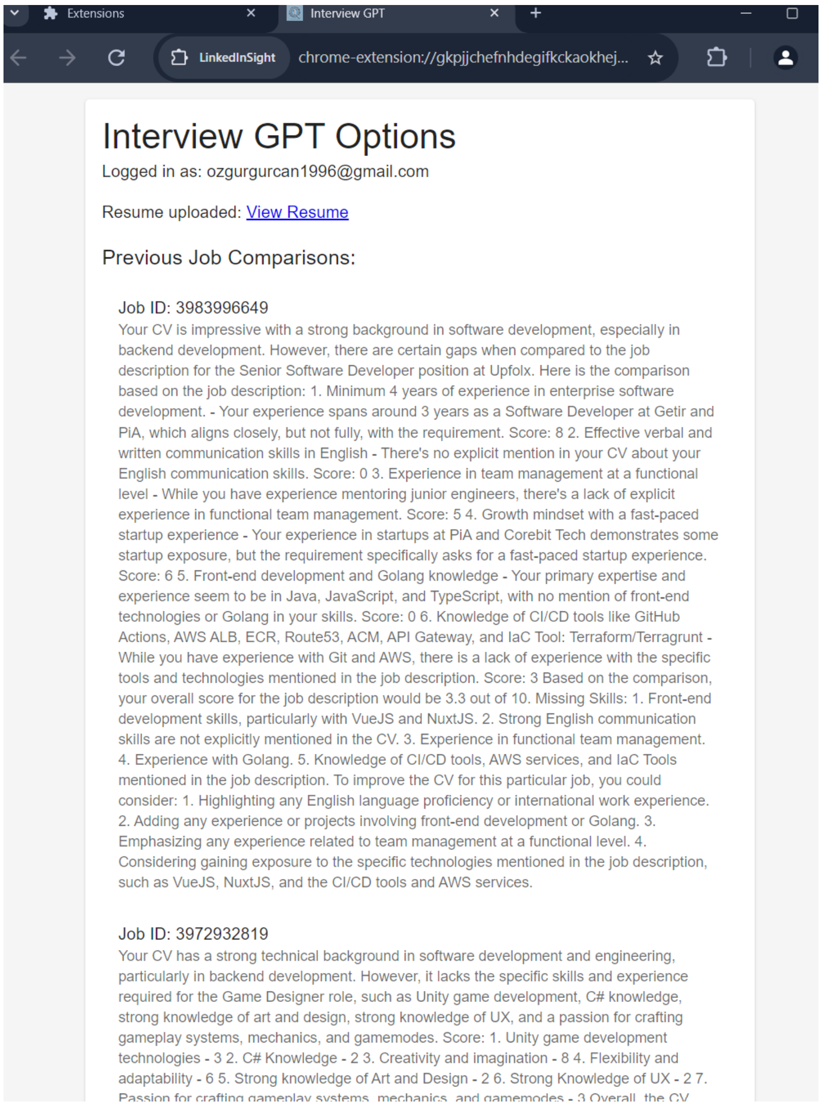
Email: ozgurgurcan1996@gmail.com

Password: •••••••

Figure 3.6: Login or Sign Up Interface

3.4.4 Viewing Previous Comparison Results

Users can view their previously uploaded resume and comparison results within the extension interface, as illustrated in Figure 3.8. This feature allows them to review past job comparisons and make informed decisions about future applications.



Interview GPT Options

Logged in as: ozgurgurcan1996@gmail.com

Resume uploaded: View Resume

Previous Job Comparisons:

Job ID: 3983996649

Your CV is impressive with a strong background in software development, especially in backend development. However, there are certain gaps when compared to the job description for the Senior Software Developer position at Upfolx. Here is the comparison based on the job description: 1. Minimum 4 years of experience in enterprise software development. - Your experience spans around 3 years as a Software Developer at Getir and PiA, which aligns closely, but not fully, with the requirement. Score: 8 2. Effective verbal and written communication skills in English - There's no explicit mention in your CV about your English communication skills. Score: 0 3. Experience in team management at a functional level - While you have experience mentoring junior engineers, there's a lack of explicit experience in functional team management. Score: 5 4. Growth mindset with a fast-paced startup experience - Your experience in startups at PiA and Corebit Tech demonstrates some startup exposure, but the requirement specifically asks for a fast-paced startup experience. Score: 6 5. Front-end development and Golang knowledge - Your primary expertise and experience seem to be in Java, JavaScript, and TypeScript, with no mention of front-end technologies or Golang in your skills. Score: 0 6. Knowledge of CI/CD tools like GitHub Actions, AWS ALB, ECR, Route53, ACM, API Gateway, and IaC Tool: Terraform/Terragrunt - While you have experience with Git and AWS, there is a lack of experience with the specific tools and technologies mentioned in the job description. Score: 3 Based on the comparison, your overall score for the job description would be 3.3 out of 10. Missing Skills: 1. Front-end development skills, particularly with VueJS and NuxtJS. 2. Strong English communication skills are not explicitly mentioned in the CV. 3. Experience in functional team management. 4. Experience with Golang. 5. Knowledge of CI/CD tools, AWS services, and IaC Tools mentioned in the job description. To improve the CV for this particular job, you could consider: 1. Highlighting any English language proficiency or international work experience. 2. Adding any experience or projects involving front-end development or Golang. 3. Emphasizing any experience related to team management at a functional level. 4. Considering gaining exposure to the specific technologies mentioned in the job description, such as VueJS, NuxtJS, and the CI/CD tools and AWS services.

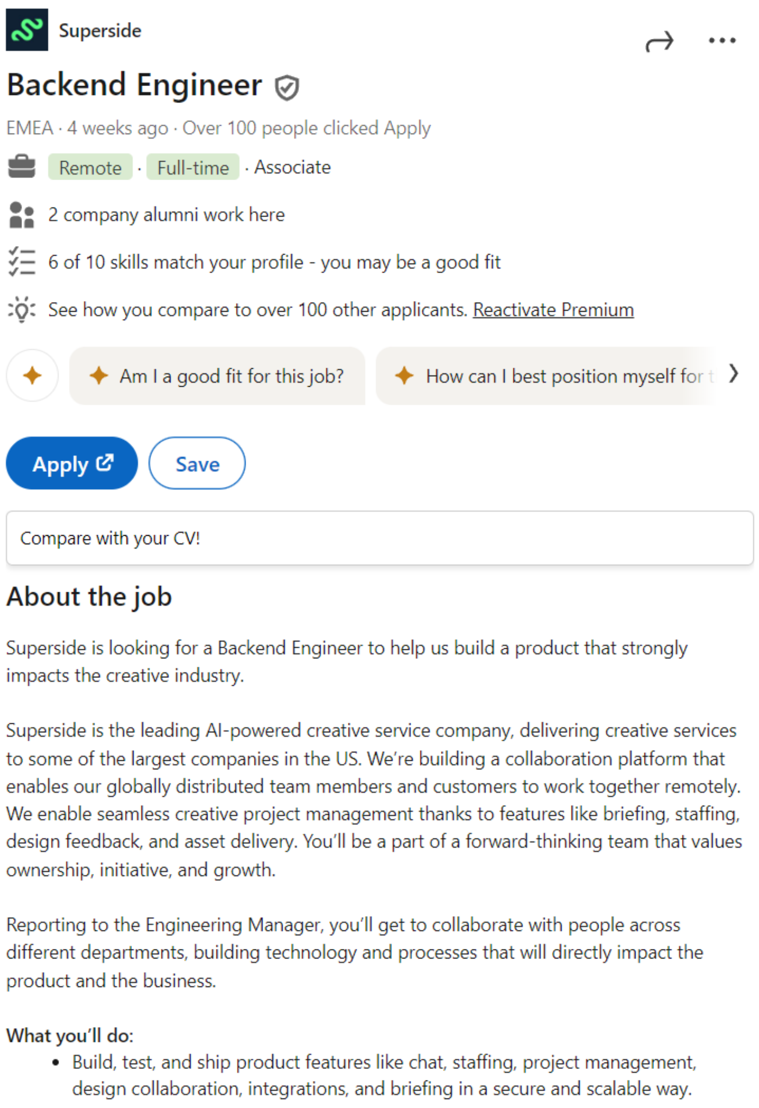
Job ID: 3972932819

Your CV has a strong technical background in software development and engineering, particularly in backend development. However, it lacks the specific skills and experience required for the Game Designer role, such as Unity game development, C# knowledge, strong knowledge of art and design, strong knowledge of UX, and a passion for crafting gameplay systems, mechanics, and gamemodes. Score: 1. Unity game development technologies - 3 2. C# Knowledge - 2 3. Creativity and imagination - 8 4. Flexibility and adaptability - 6 5. Strong knowledge of Art and Design - 2 6. Strong Knowledge of UX - 2 7. Passion for crafting gameplay systems, mechanics, and gamemodes - 3 Overall, the CV

Figure 3.7: Viewing Uploaded Resume and Comparison Results

3.4.5 Using LinkedInSight on LinkedIn Job Pages

When users navigate to a LinkedIn job posting, the LinkedInSight extension automatically integrates a "Compare with your CV!" button directly beneath the "Apply" button, as depicted in Figure 3.9. Upon clicking this button, the extension quickly processes and displays the comparison results, typically within five seconds, directly on the job posting page, as shown in Figure 3.10.


Superside

Backend Engineer

EMEA · 4 weeks ago · Over 100 people clicked Apply

Remote · Full-time · Associate

2 company alumni work here

6 of 10 skills match your profile - you may be a good fit

See how you compare to over 100 other applicants. Reactivate Premium

Am I a good fit for this job?
How can I best position myself for t
Apply
Save
Compare with your CV!

About the job

Superside is looking for a Backend Engineer to help us build a product that strongly impacts the creative industry.

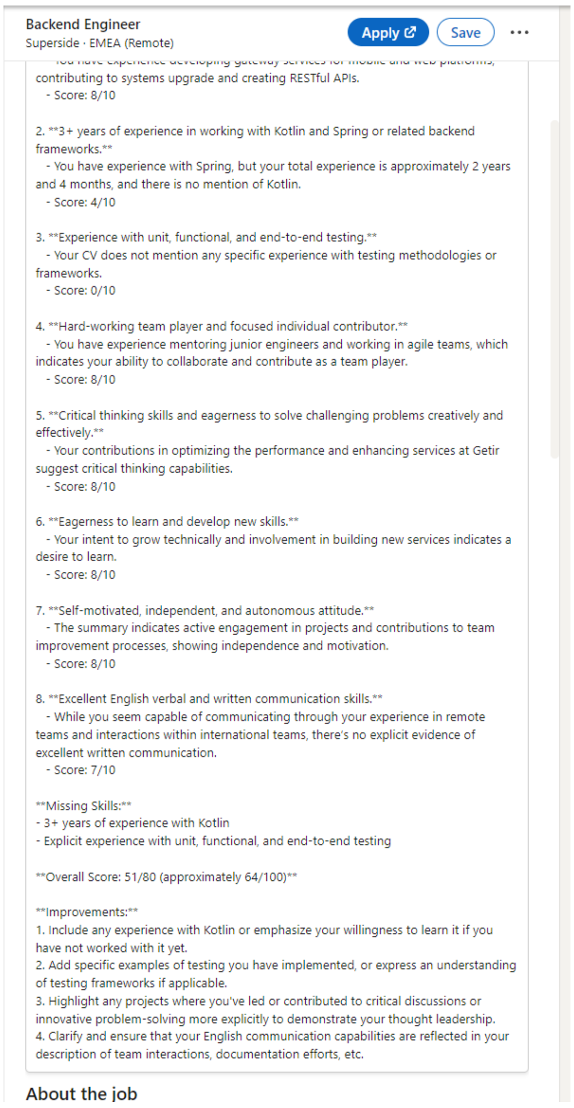
Superside is the leading AI-powered creative service company, delivering creative services to some of the largest companies in the US. We're building a collaboration platform that enables our globally distributed team members and customers to work together remotely. We enable seamless creative project management thanks to features like briefing, staffing, design feedback, and asset delivery. You'll be a part of a forward-thinking team that values ownership, initiative, and growth.

Reporting to the Engineering Manager, you'll get to collaborate with people across different departments, building technology and processes that will directly impact the product and the business.

What you'll do:

- Build, test, and ship product features like chat, staffing, project management, design collaboration, integrations, and briefing in a secure and scalable way.

Figure 3.8: LinkedIn Job Page with 'Compare with Your CV!' Button



Backend Engineer
Superside · EMEA (Remote)

- You have experience developing gateway services for mobile and web platforms, contributing to systems upgrade and creating RESTful APIs.
  - Score: 8/10

2. **3+ years of experience in working with Kotlin and Spring or related backend frameworks.**
  - You have experience with Spring, but your total experience is approximately 2 years and 4 months, and there is no mention of Kotlin.
  - Score: 4/10

3. **Experience with unit, functional, and end-to-end testing.**
  - Your CV does not mention any specific experience with testing methodologies or frameworks.
  - Score: 0/10

4. **Hard-working team player and focused individual contributor.**
  - You have experience mentoring junior engineers and working in agile teams, which indicates your ability to collaborate and contribute as a team player.
  - Score: 8/10

5. **Critical thinking skills and eagerness to solve challenging problems creatively and effectively.**
  - Your contributions in optimizing the performance and enhancing services at Getir suggest critical thinking capabilities.
  - Score: 8/10

6. **Eagerness to learn and develop new skills.**
  - Your intent to grow technically and involvement in building new services indicates a desire to learn.
  - Score: 8/10

7. **Self-motivated, independent, and autonomous attitude.**
  - The summary indicates active engagement in projects and contributions to team improvement processes, showing independence and motivation.
  - Score: 8/10

8. **Excellent English verbal and written communication skills.**
  - While you seem capable of communicating through your experience in remote teams and interactions within international teams, there's no explicit evidence of excellent written communication.
  - Score: 7/10

Missing Skills:
- 3+ years of experience with Kotlin
- Explicit experience with unit, functional, and end-to-end testing

Overall Score: 51/80 (approximately 64/100)

Improvements:
1. Include any experience with Kotlin or emphasize your willingness to learn it if you have not worked with it yet.
2. Add specific examples of testing you have implemented, or express an understanding of testing frameworks if applicable.
3. Highlight any projects where you've led or contributed to critical discussions or innovative problem-solving more explicitly to demonstrate your thought leadership.
4. Clarify and ensure that your English communication capabilities are reflected in your description of team interactions, documentation efforts, etc.

About the job

Figure 3.9: Displaying Job-CV Comparison Results

The interface is crafted with usability in mind, featuring one-click comparisons that significantly streamline the job application process. By minimizing user effort and offering real-time feedback, LinkedInSight enhances the overall job-seeking experience on LinkedIn.

3.5 Performance Optimization and Security Considerations

To ensure a smooth user experience, several performance optimization strategies were implemented. Caching mechanisms reduce the need for repeated AI processing by storing previously computed results, which can be quickly retrieved for subsequent requests. Asynchronous processing further enhances system responsiveness, allowing the extension to handle multiple tasks concurrently without causing delays or disruptions to the user experience.

In addition to performance optimization, security remains a central focus of the LinkedInSight system. All communications are encrypted using HTTPS, and strict access controls are enforced through Firebase security rules and JWT-based authentication. These measures protect user data from unauthorized access and ensure that the system complies with best practices in data security and privacy.

3.6 Deployment and Maintenance

The deployment of the LinkedInSight system is managed through a combination of automated tools and manual oversight, ensuring that the extension and backend are consistently updated and maintained. The backend server is hosted on an Oracle Virtual Private Server (VPS) and managed using Coolify, an open-source platform that supports Continuous Integration/Continuous Deployment (CI/CD) workflows. This setup allows for rapid deployment of updates and scaling of resources as needed, ensuring that the system remains responsive and capable of handling increased user demand.

Monitoring tools, including New Relic, are employed to track the performance and health of the system, enabling proactive identification and resolution of potential issues. Regular updates to both the extension and backend codebases ensure that the system remains secure and up-to-date with the latest advancements in AI and web technologies.

Chapter 4

Results and performance analysis of the extension

Chapter 4 provides a detailed analysis of the LinkedInSight extension's performance and evaluation results. This chapter explores the functional accuracy of the AI-driven job-CV comparison, the system's responsiveness, and user experience feedback. Additionally, it addresses security, privacy, and ethical considerations, as well as the challenges and limitations encountered during testing, outlining the implications for future development.

4.1 Evaluation Methodology and Overview

The LinkedInSight extension was developed with the primary aim of enhancing the job-seeking experience on LinkedIn by leveraging AI-powered comparisons between job descriptions and user CVs. To assess the effectiveness and efficiency of this extension, a comprehensive evaluation methodology was employed. The evaluation focused on several key aspects, including the functional accuracy of the job-CV matching process, the responsiveness of the system under varying conditions, and the overall user experience, despite the extension not yet being publicly released.

The extension was tested in a controlled environment to simulate real-world usage scenarios. This involved functional testing across various LinkedIn job postings, as well as usability testing to gauge the user experience and interface intuitiveness. Additionally, the backend system's performance was scrutinized to ensure reliable processing of AI-driven comparisons, secure user data management, and prompt responses to user interactions.

4.2 Functional Accuracy and System Performance

The primary functionality of the LinkedInSight extension centers on accurately comparing job descriptions with user-uploaded CVs, using advanced natural language processing techniques facilitated by the ChatGPT 4.0 Mini API. The accuracy of these comparisons was evaluated through manual verification processes, where the AI-generated results were cross-referenced with human judgments. While comprehensive field testing was not possible due to the extension's unpublished status, initial findings indicate that the AI model demonstrates a satisfactory level of accuracy in matching relevant skills and qualifications.

System performance, particularly response times, was also a key focus of the evaluation. Despite the limited scope of testing, it was observed that the backend system, built on Node.js and Express, handled processing tasks efficiently. The integration of Firebase services for user authentication, data storage, and real-time updates further contributed to the smooth operation of the extension. The use of asynchronous processing and caching strategies were instrumental in maintaining a responsive user interface, minimizing delays during AI processing tasks.

4.3 User Experience and Interface Usability

User experience (UX) is a critical factor in the success of any browser extension, and the LinkedInSight extension was designed with a focus on simplicity and ease of use. The interface, built with React and Material-UI, was subjected to usability testing to ensure it met the needs of potential users. Key aspects of the interface, such as the integration of the "Compare with CV" button directly into LinkedIn job pages and the clear presentation of comparison results, were highlighted as particularly effective in streamlining the job application process.

Feedback gathered during the testing phase indicated that users found the extension intuitive and easy to navigate, with the one-click CV comparison feature being especially well-received. However, there were suggestions for further improvements, such as enhancing the speed at which comparison results are delivered and refining the AI's recommendations to be more contextually relevant.

4.4 Security, Privacy, and Ethical Considerations

Given the sensitive nature of the data involved, particularly user CVs and personal information, stringent security and privacy measures were implemented. Data encryption was utilized for all communications between the extension, backend, and Firebase services, ensuring that user information remained protected throughout the process. Additionally, the use of JSON Web Tokens (JWT) for authentication provided a robust mechanism for securing access to user data and ensuring that only authorized users could perform comparisons.

Ethical considerations were also a central aspect of the project, particularly in relation to the AI's role in influencing job application decisions. Efforts were made to ensure that the AI-driven recommendations were transparent and fair, minimizing potential biases that could disadvantage certain users. The extension’s design emphasized user control and informed decision-making, allowing users to view and assess AI-generated insights before acting upon them.

4.5 Challenges and Limitations

Despite the promising results observed during testing, the LinkedInSight extension faces several challenges and limitations that need to be addressed in future development phases. The unpublished status of the extension limited the scope of testing, restricting the ability to gather comprehensive data on real-world performance. Additionally, the reliance on LinkedIn’s existing infrastructure posed challenges in maintaining compatibility with dynamic changes to the platform.

The accuracy of the AI model, while generally satisfactory, could benefit from further refinement, particularly in understanding and comparing complex job requirements with user qualifications. Moreover, the need for ongoing updates to the AI model and backend systems is evident to keep pace with advancements in AI and changes to LinkedIn’s platform.

4.6 Summary and Implications for Future Development

In summary, the LinkedInSight extension demonstrates significant potential to enhance the job-seeking experience on LinkedIn through AI-powered job-CV comparisons. Initial testing indicates that the extension is capable of delivering accurate and timely insights, with a user-friendly interface that integrates seamlessly into LinkedIn's ecosystem. However, the unpublished status of the extension and the limitations of the current testing phase highlight the need for further development and testing to fully realize its potential.

Future work will focus on expanding the scope of testing, refining the AI algorithms, and addressing the challenges identified during the initial evaluation. By continuing to enhance the system's accuracy, performance, and user experience, LinkedInSight can become a valuable tool for job seekers, improving the efficiency and effectiveness of the job application process.

Chapter 5

Implications, Limitations, and Future Work

Chapter 5 explores the broader implications, limitations, and future directions for the LinkedInSight project. This chapter summarizes the key findings and significance of the project, discusses the ethical considerations and challenges associated with AI-driven job matching, and outlines the limitations of the current implementation. It also proposes future work, including expanding the extension's capabilities, addressing its limitations, and exploring research opportunities through interdisciplinary collaboration.

5.1 Summary of Key Findings and Project Significance

The LinkedInSight project represents a significant advancement in the application of AI and NLP technologies to enhance the job-seeking process. By integrating these technologies into a Chrome extension tailored for LinkedIn, this project has laid the groundwork for more efficient, accurate, and user-friendly job application processes. The extension's ability to perform AI-driven comparisons between job descriptions and user CVs directly within the LinkedIn platform addresses a critical need for streamlined job matching, potentially transforming how users interact with professional networking sites.

The findings from the initial testing phase indicate that the extension is capable of delivering valuable insights that can improve job-seeking outcomes. Despite the limitations posed by the extension's unpublished status, the controlled testing environment provided useful data on the system's functionality, user interface, and overall performance. The project's significance extends beyond its immediate application, offering insights into the broader implications of integrating AI into professional networking platforms.

5.2 Implications for Users, Recruiters, and the Job Market

The introduction of AI-driven tools like LinkedInSight has profound implications for job seekers, recruiters, and the broader job market. For job seekers, the extension offers a powerful tool that can save time and reduce the cognitive load associated with evaluating job opportunities. By automating the comparison of CVs with job descriptions, the extension enables users to focus on the most relevant opportunities, thereby improving the quality and relevance of job applications.

For recruiters and employers, LinkedInSight can enhance the recruitment process by promoting better alignment between job requirements and candidate qualifications. This can lead to higher quality applicant pools and more efficient hiring processes, ultimately benefiting organizations by reducing the time and resources spent on recruitment.

At the macro level, the widespread adoption of AI-powered job matching tools could have significant implications for the job market as a whole. By improving the efficiency of job-skill alignment, such tools could contribute to reducing unemployment rates and increasing job satisfaction among employees, as individuals find positions that better match their skills and career aspirations.

5.3 Ethical Considerations and Challenges

The integration of AI in job matching raises important ethical considerations, particularly concerning privacy, bias, and transparency. The LinkedInSight extension handles sensitive personal data, such as user CVs and job histories, necessitating robust data protection measures. Encryption and secure data storage practices have been implemented to safeguard user information, but ongoing vigilance is required to ensure these measures remain effective against evolving cybersecurity threats.

AI bias is another critical concern. The algorithms used in LinkedInSight must be continually assessed to ensure they do not inadvertently disadvantage certain groups of users. Bias in AI-driven job matching could exacerbate existing inequalities in the job market, making it essential to implement fairness checks and regular audits of the AI’s decision-making processes.

Transparency is also paramount in maintaining user trust. Users need to understand how the AI operates, the factors it considers when making comparisons, and how they can influence or override AI-generated recommendations. Clear communication about these aspects is essential to foster informed decision-making and prevent over-reliance on AI.

5.4 Limitations of the Study and Implementation

The LinkedInSight project, while promising, has several limitations that must be acknowledged. First and foremost, the extension has not yet been published, which restricted the scope of testing and limited the ability to gather real-world usage data. This lack of public deployment means that the findings are based on simulated environments and may not fully capture the challenges or variations encountered in broader, real-world applications.

The current implementation is also constrained by its reliance on LinkedIn’s platform, which may change its structure or functionalities over time. These changes could affect the extension’s compatibility and necessitate ongoing updates to maintain functionality. Furthermore, the AI model used in LinkedInSight, while effective, is dependent on the quality and comprehensiveness of the data it processes. Any limitations in the data (e.g., incomplete CVs or job descriptions) could impact the accuracy of the AI’s recommendations.

5.5 Future Work and Development Directions

Future work on LinkedInSight should focus on addressing the limitations identified and expanding the extension’s capabilities. Immediate priorities include the public release of the extension, which would allow for comprehensive real-world testing and provide more robust data on user engagement, system performance, and overall effectiveness. This will be crucial for refining the extension’s features and improving user satisfaction.

In the medium term, there is potential to expand the functionality of LinkedInSight to support additional platforms beyond LinkedIn. This could involve integrating with other professional networking sites or job portals, thereby broadening the extension’s user base and applicability. Additionally, enhancements to the AI model, such as incorporating more advanced NLP techniques or personalizing the AI based on individual user data, could further improve the accuracy and relevance of the job matching process.

In the long term, the vision for LinkedInSight could extend beyond its current form as a browser extension. Exploring the development of a standalone platform or mobile application that offers comprehensive career development tools, including job searching, networking, and skills assessment, could position LinkedInSight as a leader in the HR tech industry.

5.6 Research Opportunities and Interdisciplinary Collaboration

The LinkedInSight project opens up several avenues for future research. Longitudinal studies on the impact of AI-assisted job searching on career outcomes would provide valuable insights into the long-term benefits and potential drawbacks of such technologies. Comparative studies between LinkedInSight and other job matching tools could also offer important benchmarks and identify areas for improvement.

Interdisciplinary collaboration with fields such as labor economics, organizational psychology, and human-computer interaction could further enrich the development of LinkedInSight. Understanding the societal impacts of widespread AI adoption in job searching, including its effects on employment patterns and worker mobility, is critical for ensuring that these technologies are used responsibly and equitably.

5.7 Conclusion and Final Thoughts

The LinkedInSight project represents a pioneering effort to harness the power of AI and NLP in enhancing the job-seeking experience on LinkedIn. While there are challenges and limitations to be addressed, the project has demonstrated the potential of AI-driven tools to transform how job seekers interact with professional networking platforms. As AI continues to evolve, there is a tremendous opportunity to further integrate these technologies into the recruitment process, ultimately contributing to more efficient, fair, and satisfying job markets.

Looking ahead, the ongoing development and refinement of LinkedInSight will be essential to realizing its full potential. By continuing to prioritize user needs, ethical considerations, and technological advancements, LinkedInSight can become a valuable asset for job seekers and recruiters alike, shaping the future of work in meaningful ways.

References

Bastian, M., Hayes, M., Vaughan, W., Shah, S., Skomoroch, P., Kim, H., Uryasev, S., & Lloyd, C. (2014). LinkedIn skills: Large-scale topic extraction and inference. Proceedings of the 8th ACM Conference on Recommender Systems, 1-8. https://doi.org/10.1145/2645710.2645728

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. International Journal on Digital Libraries, 17(4), 305-338. https://doi.org/10.1007/s00799-015-0156-0

Chen, J., Gao, L., & Tang, Z. (2018). Information extraction from resume documents in PDF format. 2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA), 268-273. https://doi.org/10.1109/ICBDA.2018.8367707

Dhillon, S., & Mahmoud, Q. H. (2016). An evaluation framework for cross-platform mobile application development tools. Software: Practice and Experience, 46(10), 1331-1357. https://doi.org/10.1002/spe.2376

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

Mehta, S., Paunwala, C. N., & Vaidya, B. (2021). Recruitment recommendation system using machine learning approaches. 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 1361-1367. https://doi.org/10.1109/ICACCS51430.2021.9441867

Mehrabad, M. S., & Brojeny, M. F. (2007). The development of an expert system for effective selection and appointment of the jobs applicants in human resource management. Computers & Industrial Engineering, 53(2), 306-312. https://doi.org/10.1016/j.cie.2007.06.023

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 26, 3111-3119.

Norman, D. A. (2013). The design of everyday things: Revised and expanded edition. Basic Books.

Raghavan, M., Barocas, S., Kleinberg, J., & Levy, K. (2020). Mitigating bias in algorithmic hiring: Evaluating claims and practices. Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 469-481. https://doi.org/10.1145/3351095.3372828

Stadler, G., Hautz, J., Matzler, K., & Von Den Eichen, S. F. (2020). Distributed knowledge across organizational boundaries: The role of online professional networks in sharing expertise. In Engineering a Better Future (pp. 107-124). Springer. https://doi.org/10.1007/978-3-030-20656-7_7

Zhao, M., Javed, F., Jacob, F., & McNair, M. (2020). SKILL: A system for skill identification and normalization. Proceedings of the AAAI Conference on Artificial Intelligence, 34(08), 9529-9536. https://doi.org/10.1609/aaai.v34i08.7037

Appendices

Appendix A: Chrome Extension Code

~~~
# webpack.prod.js

```js
const { merge } = require('webpack-merge');
const common = require('./webpack.common.js');

module.exports = merge(common, { mode: 'production' });

```

# webpack.dev.js

```js
const { merge } = require('webpack-merge');
const common = require('./webpack.common.js');

module.exports = merge(common, { mode: 'development', devtool: 'cheap-module-source-map' });

```

# webpack.common.js

```js
const path = require('path');
const CopyPlugin = require('copy-webpack-plugin');
const HtmlPlugin = require('html-webpack-plugin');

module.exports = {
  entry: {
    popup: path.resolve('src/popup/popup.tsx'),
    options: path.resolve('src/options/options.tsx'),
    background: path.resolve('src/background/background.ts'),
    contentScript: path.resolve('src/contentScript/contentScript.tsx'),
  },
  module: {
    rules: [
      { test: /\.tsx?$/, use: 'ts-loader', exclude: /node_modules/ },
      { test: /\.css$/i, use: ['style-loader', 'css-loader'] },
      { test: /\.(jpg|jpeg|png|woff|woff2|eot|ttf|svg)$/, type: 'asset/resource' }
    ]
  },
  resolve: { extensions: ['.tsx', '.ts', '.js'] },
  plugins: [
    new CopyPlugin({ patterns: [{ from: 'src/static', to: 'dist' }] }),
    ...['popup', 'options'].map(chunk => new HtmlPlugin({ filename: `${chunk}.html`, chunks:
[chunk] }))
  ],
  output: { filename: '[name].js', path: path.resolve('dist') },
  optimization: { splitChunks: { chunks: chunk => chunk.name !== 'contentScript' } }
};

```

# tsconfig.json
~~~

~~~
```json
{
  "compilerOptions": {
    "jsx": "react",
    "module": "es6",
    "target": "es6",
    "moduleResolution": "node",
    "esModuleInterop": true,
    "lib": [
      "dom",
      "esnext"
    ]
  }
}
```

# package.json

```json
{
  "name": "LinkedInSight",
  "version": "1.0.0",
  "description": "Compare LinkedIn job postings with user-uploaded resumes and provide feedback.",
  "scripts": {
    "start": "webpack --watch --progress --config webpack.dev.js",
    "build": "webpack --watch --progress --config webpack.prod.js"
  },
  "author": "Ozgur Gurcan",
  "devDependencies": {
    "@emotion/react": "^11.11.4",
    "@emotion/styled": "^11.11.5",
    "@mui/icons-material": "^5.15.16",
    "@mui/material": "^5.15.16",
    "@types/chrome": "^0.0.267",
    "@types/react": "^18.3.0",
    "@types/react-dom": "^18.3.0",
    "clean-webpack-plugin": "^3.0.0",
    "copy-webpack-plugin": "^7.0.0",
    "css-loader": "^5.2.4",
    "fontsource-roboto": "^4.0.0",
    "html-webpack-plugin": "^4.5.2",
    "react": "^18.3.0",
    "react-dom": "^18.3.0",
    "style-loader": "^2.0.0",
    "terser-webpack-plugin": "^5.1.1",
    "ts-loader": "^8.2.0",
    "typescript": "^5.4.5",
    "webpack": "^5.35.1",
    "webpack-cli": "^4.6.0",
    "webpack-merge": "^5.7.3"
  }
}
```

# .gitignore

```
dist
node_modules

```

# src\constants.tsx

```tsx
~~~

~~~
export const ELEMENT_IDS = {
    JOB_DETAILS: 'job-details',
    COMPARISON_CARD: 'job-compare-card',
};

```


# src\utils\storage.ts

```ts
export const setStorageUser = (user) => chrome.storage.local.set({ user });
export const getStorageUser = () => new Promise(resolve => chrome.storage.local.get(['user'], res
=> resolve(res.user || null)));
export const clearStorageUser = () => chrome.storage.local.remove(['user']);

export const setStorageToken = (token) => chrome.storage.local.set({ authToken: token });
export const getStorageToken = () => new Promise(resolve => chrome.storage.local.get(['authToken'],
res => resolve(res.authToken || null)));
export const clearStorageToken = () => chrome.storage.local.remove(['authToken']);

```


# src\utils\messages.ts

```ts
export enum Messages {
    // Content script to background script
    TOGGLE_OVERLAY,

    // Background script to content script
}
```


# src\utils\jobUtils.tsx

```tsx
export function getJobDescription() {
    const jobDescriptionElement = document.getElementById('job-details');
    if (!jobDescriptionElement) return 'Job description not found';

    let jobDescription = '';
    jobDescriptionElement.querySelectorAll('h2, p, ul, li').forEach(section => {
        if (section.tagName === 'H2') jobDescription += `\n\n${section.textContent}\n`;
        else if (section.tagName === 'UL') section.querySelectorAll('li').forEach(li =>
jobDescription += ` - ${li.textContent}\n`);
        else jobDescription += `${section.textContent}\n`;
    });

    return jobDescription.trim();
}

export function getLinkedinJobId() {
    return new URLSearchParams(window.location.search).get('currentJobId') || '';
}

```


# src\utils\chromeHelper.ts

```ts
export function getCurrentTab(): Promise<chrome.tabs.Tab> {
    return new Promise((resolve, reject) => {
        chrome.tabs.query({ active: true, currentWindow: true }, (tabs) => {
            if (tabs.length === 0) {
                reject('No active tab');
            }
~~~

~~~~
            resolve(tabs[0]);
        });
    });
}
```


# src\utils\api.ts

```ts
import { clearStorageToken, clearStorageUser, getStorageToken, setStorageToken, setStorageUser }
from "./storage";

const BASE_URL = 'https://linkedinsight.ozgurgurcan.dev/api/v1';
// const BASE_URL = 'https://linkedinsight.vercel.app/api/v1';
//const BASE_URL = 'http://127.0.0.1:3000/api/v1';

export async function login(email, password) {
    const response = await fetch(`${BASE_URL}/auth/login`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, password }),
    });
    const user = await response.json();
    await setStorageToken(user.token);
    await setStorageUser(user);
    return user;
}

export async function signup(email, password) {
    const response = await fetch(`${BASE_URL}/auth/signup`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ email, password }),
    });
    const user = await response.json();
    await setStorageUser(user);
    await setStorageToken(user.token);
    return user;
}

export const logout = async () => {
    await clearStorageToken();
    await clearStorageUser();
};

export const getUser = async () => {
    const response = await authFetch(`${BASE_URL}/user`);
    const user = await response.json();
    await setStorageUser(user);
    return user;
};

export async function uploadResume(file, uid) {
    const formData = new FormData();
    formData.append("resume", file);
    const response = await authFetch(`${BASE_URL}/upload-resume?uid=${uid}`, {
        method: 'POST',
        body: formData
    });
    return response.json();
}

export async function compareJobWithCv(jobDescription, linkedinJobId, uid) {
    const response = await authFetch(`${BASE_URL}/job/compare`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
~~~~

~~~
        body: JSON.stringify({ jobDescription, linkedinJobId, uid })
    });
    return response.json();
}

async function authFetch(url, options = {}) {
    const token = await getStorageToken();
    return fetch(url, {
        ...options,
        headers: { ...options.headers, 'Authorization': `Bearer ${token}` },
    });
}

```

# src\static\manifest.json

```json
{
  "name": "LinkedInSight",
  "description": "Compare LinkedIn job postings with user-uploaded resumes and provide feedback.",
  "version": "1.0.6",
  "manifest_version": 3,
  "icons": {
    "16": "icon2.png",
    "48": "icon2.png",
    "128": "icon2.png"
  },
  "action": {
    "default_title": "LinkedInSight",
    "default_icon": "icon2.png"
  },
  "permissions": [
    "storage",
    "offscreen",
    "activeTab"
  ],
  "options_page": "options.html",
  "background": {
    "service_worker": "background.js"
  },
  "content_scripts": [
    {
      "matches": [
        "https://www.linkedin.com/jobs/*"
      ],
      "js": [
        "contentScript.js"
      ]
    }
  ]
}
```

# src\static\icon2.png

This is a binary file of the type: Image

# src\static\icon.png

This is a binary file of the type: Image

# src\contentScript\contentScript.tsx

```tsx
import React, { useEffect, useState } from 'react';
~~~

~~~
import ReactDOM from 'react-dom';
import { compareJobWithCv } from '../utils/api';
import { getJobDescription, getLinkedinJobId } from '../utils/jobUtils';
import { ELEMENT_IDS } from '../constants';
import ComparisonCard from '../components/ComparisonCard';

const App: React.FC = () => {
    const [user, setUser] = useState(null);
    const [comparisonResult, setComparisonResult] = useState('');

    const updateCard = (result) => {
        const jobDescriptionElement = document.getElementById(ELEMENT_IDS.JOB_DETAILS);
        if (!jobDescriptionElement) return;

        const existingCard = document.getElementById(ELEMENT_IDS.COMPARISON_CARD);
        if (existingCard) existingCard.remove();

        const card = document.createElement('div');
        card.id = ELEMENT_IDS.COMPARISON_CARD;
        jobDescriptionElement.prepend(card);

        ReactDOM.render(<ComparisonCard user={user} comparisonResult={result}
onCompare={handleCompare} />, card);
    };

    const handleCompare = async () => {
        const jobDescription = getJobDescription();
        const linkedinJobId = getLinkedinJobId();
        const { comparisonResult } = await compareJobWithCv(jobDescription, linkedinJobId,
user.uid);
        setComparisonResult(comparisonResult);
        updateCard(comparisonResult);
    };

    useEffect(() => {
        chrome.storage.local.get(['user'], (result) => {
            const user = result.user || null;
            setUser(user);
            if (user) updateCard('');
        });
    }, []);

    return null;
};

ReactDOM.render(<App />, document.createElement('div'));

```

# src\contentScript\contentScript.css

```css
.overlayCard {
    position: fixed;
    left: 5%;
    top: 15%;
    max-width: 240px;
    max-height: 240px;
    background-color: white !important;
    z-index: 1000;
}
```

# src\components\ErrorBoundary.tsx

```tsx
~~~

~~~
import React from 'react';

class ErrorBoundary extends React.Component {
    state = { hasError: false };

    static getDerivedStateFromError() { return { hasError: true }; }

    componentDidCatch(error, errorInfo) { console.error('Uncaught error:', error, errorInfo); }

    render() { return this.state.hasError ? <h1>Something went wrong.</h1> : this.props.children; }
}

export default ErrorBoundary;

```

# src\components\ComparisonCard.tsx

```tsx
import React from 'react';
import { User } from '../utils/api';

const ComparisonCard: React.FC<{ user: User, comparisonResult: string, onCompare: () => void }> =
({ comparisonResult, onCompare }) => (
    <div style={{ border: '1px solid #ccc', borderRadius: '5px', padding: '10px', marginBottom:
'10px', boxShadow: '0 2px 4px rgba(0,0,0,0.1)' }}>
        <button onClick={onCompare}>Compare with your CV!</button>
        {comparisonResult && <div style={{ marginTop: '10px', whiteSpace: 'pre-wrap'
}}><strong>Comparison Result:</strong><br />{comparisonResult}</div>}
    </div>
);

export default ComparisonCard;

```

# src\options\options.tsx

```tsx
import React, { useEffect, useState } from 'react';
import ReactDOM from 'react-dom';
import { Box, Button, Card, CardContent, Grid, TextField, Typography } from '@mui/material';
import { getUser, login, logout, signup, uploadResume } from '../utils/api';
import { getStorageUser } from '../utils/storage';

const App: React.FC = () => {
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');
    const [user, setUser] = useState(null);
    const [file, setFile] = useState(null);

    useEffect(() => {
        getStorageUser().then(setUser);
    }, []);

    const handleAuth = async (authFunc) => {
        try {
            setUser(await authFunc(email, password));
            setError('');
        } catch (err) {
            setError(`Auth failed: ${err.message}`);
        }
    };

    const handleFileChange = (e) => setFile(e.target.files[0]);
~~~

```
    const handleUploadResume = async () => {
        if (!file || !user) return;
        try {
            await uploadResume(file, user.uid);
            setFile(null);
        } catch (err) {
            setError('Failed to upload resume');
        }
    };

    return (
        <Box mx="10%" my="2%">
            <Card>
                <CardContent>
                    <Typography variant="h4">Interview GPT Options</Typography>
                    {error && <Typography color="error">{error}</Typography>}
                    {!user ? (
                        <Grid container spacing={2}>
                            <Grid item xs={12}>
                                <TextField fullWidth label="Email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
                            </Grid>
                            <Grid item xs={12}>
                                <TextField fullWidth label="Password" type="password"
value={password} onChange={(e) => setPassword(e.target.value)} />
                            </Grid>
                            <Grid item xs={6}>
                                <Button variant="contained" color="primary" onClick={() =>
handleAuth(login)}>Login</Button>
                            </Grid>
                            <Grid item xs={6}>
                                <Button variant="contained" color="secondary" onClick={() =>
handleAuth(signup)}>Signup</Button>
                            </Grid>
                        </Grid>
                    ) : (
                        <Grid container spacing={2}>
                            <Grid item xs={12}>
                                <Typography>Logged in as: {user.email}</Typography>
                            </Grid>
                            {user.cv?.url ? (
                                <Grid item xs={12}>
                                    <Typography>Resume uploaded: <a href={user.cv.url}
target="_blank">View Resume</a></Typography>
                                </Grid>
                            ) : (
                                <Grid item xs={12}>
                                    <input accept=".pdf,.doc,.docx" style={{ display: 'none' }}
id="raised-button-file" type="file" onChange={handleFileChange} />
                                    <label htmlFor="raised-button-file">
                                        <Button variant="contained" component="span">Select
Resume</Button>
                                    </label>
                                    {file && (
                                        <Button onClick={handleUploadResume} variant="contained"
color="primary" style={{ marginLeft: '10px' }}>
                                            Upload Resume
                                        </Button>
                                    )}
                                </Grid>
                            )}
                            <Grid item xs={12}>
                                <Button variant="contained" color="secondary"
onClick={logout}>Logout</Button>
                            </Grid>
```

~~~
                        </Grid>
                    )}
                </CardContent>
            </Card>
        </Box>
    );
};

ReactDOM.render(<App />, document.createElement('div'));

```

# src\options\options.css

```css
body {
  background-color: #f5f5f5;
}

```

# src\background\background.ts

```ts
chrome.action.onClicked.addListener(() => chrome.runtime.openOptionsPage());

chrome.runtime.onInstalled.addListener(() => chrome.runtime.openOptionsPage());

```
~~~

Appendix B: Node.js Server Code

~~~
# package.json

```json
{
  "name": "linkedinsight-server",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "nodemon": "nodemon src/index.js",
    "start": "node src/index.js",
    "digest": "npx ai-digest"
  },
  "license": "ISC",
  "dependencies": {
    "ai-digest": "^1.0.5",
    "bcrypt": "^5.1.1",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "cross-fetch": "^4.0.0",
    "dotenv": "^16.4.0",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "firebase-admin": "^12.1.0",
    "jsonwebtoken": "^9.0.2",
    "multer": "^1.4.5-lts.1",
    "openai": "^4.25.0",
    "pdf-parse": "^1.1.1"
  },
  "devDependencies": {
    "nodemon": "^3.0.3"
  },
  "helpful": [
    "https://www.sitepoint.com/speech-to-text-whisper-react-node/",
    "https://github.com/vercel/examples/blob/main/solutions/express/vercel.json"
  ]
}
```

# .gitignore

```
node_modules
.env
/uploads/
./idea/
.vercel
*.key
*.zip
donem odevi.txt

```

# src\pdfHandler.js
~~~

~~~
```js
const pdf = require('pdf-parse');
const { completions } = require('./openai');

const extractTextFromPdf = async (dataBuffer) => (await pdf(dataBuffer)).text;

const transcribeCvText = async (cvText) => completions("you are a recruiter, here is my cv,
sumirize it for me please.", cvText);

module.exports = { extractTextFromPdf, transcribeCvText };

```

# src\openai.js

```js
const OpenAI = require("openai");
require('dotenv').config();

const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

const completions = async (systemContent, userContent) => {
    const { choices } = await openai.chat.completions.create({
        messages: [{ role: "system", content: systemContent }, { role: 'user', content: userContent
}],
        model: "gpt-4o-mini"
    });
    return choices[0].message.content;
};

module.exports = { completions };

```

# src\jobHandler.js

```js
const { completions } = require('./openai');

async function compareJobDescriptionWithMyCv(jobDescription, cvText) {
  return await completions(
    "Your job is to compare a job description with my cv. When you compare you should give score
from 0 to 10 for 10 perfect match, 0 no match according to job description. Give importance to
Mandatory skills. Give overall score. Specify all missing skills. Also please provide what should
be improved in the cv." +
    " For example you can give response like: " +
    "1. Minimum 3 years Java development experience.\n" +
    "    - You have approximately 2 years and 4 months of experience, which is close.\n" +
    "    - Score: 8/10\n" +
    "\n" +
    "2. Computer engineering degree.\n" +
    "    - You have a Bachelor of Engineering in Computer Engineering.\n" +
    "    - Score: 10/10" +
    "**Missing Skills:**" +
    "**Overall score: 67/100.**" +
    "**Improvements:**",
    "Here is my cv: " + cvText + "\n\nHere is the job description: " + jobDescription
  )
}

module.exports = { compareJobDescriptionWithMyCv };

```

# src\index.js
~~~

~~~
```js
const app = require("./app");
app.listen(process.env.PORT, () => console.log(`App listening on port ${process.env.PORT}`));

```

# src\firebase.js

```js
const { initializeApp, cert } = require('firebase-admin/app');
const { getFirestore, Timestamp, FieldValue } = require('firebase-admin/firestore');
const { getStorage, getDownloadURL } = require('firebase-admin/storage');
const { getAuth } = require('firebase-admin/auth');
const serviceAccount = require("./config/firebase.json");

initializeApp({ credential: cert(serviceAccount), storageBucket: 'interview-gpt-cc505.appspot.com'
});

const DB = getFirestore();
const BUCKET = getStorage().bucket();
const AUTH = getAuth();

const loginWithEmail = (email) => AUTH.getUserByEmail(email).then(user => updateUser(user.uid, {
lastLogin: FieldValue.serverTimestamp() }).then(() => user));

const getUserDO = (uid) => DB.collection('users').doc(uid).get().then(doc => doc.exists ?
doc.data() : null);

const uploadCv = (uid, file) => {
  const fileName = `resume/${uid}/${file.originalname}`;
  const fileUpload = BUCKET.file(fileName);
  return new Promise((resolve, reject) => {
    fileUpload.createWriteStream({ contentType: file.mimetype })
      .on('error', reject)
      .on('finish', () => resolve(getDownloadURL(fileUpload)))
      .end(file.buffer);
  });
};

const updateUser = (uid, data) => DB.collection('users').doc(uid).set(data, { merge: true });

const verifyIdToken = async (idToken) => AUTH.verifyIdToken(idToken).then(decoded =>
decoded.uid).catch(() => null);

module.exports = { loginWithEmail, uploadCv, updateUser, getUserDO, verifyIdToken, DB, AUTH, BUCKET
};

```

# src\app.js

```js
require("dotenv").config();
const express = require("express");
const cors = require("cors");
const { authenticateToken } = require("./middleware/auth");

const app = express();
app.use(cors());
app.use(express.json());

app.get("/", (req, res) => res.send("Express on Coolify"));

app.use("/api/v1", authenticateToken, require("./api"));
~~~

~~~
module.exports = app;

```

# api\index.js

```js
const app = require('../src/app');

module.exports = app;
```

# src\middleware\auth.js

```js
const jwt = require('jsonwebtoken');
const JWT_SECRET = process.env.JWT_SECRET;

const authenticateToken = (req, res, next) => {
    const token = req.headers['authorization']?.split(' ')[1];
    if (!token) return res.sendStatus(401);

    jwt.verify(token, JWT_SECRET, (err, user) => {
        if (err) return res.sendStatus(403);
        req.user = user;
        next();
    });
};

module.exports = { authenticateToken };

```

# src\config\firebase.json

```json
{
  "type": "service_account",
  "project_id": "interview-gpt-cc505",
  "private_key_id": "******",
  "private_key": "*****",
  "client_id": "******",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": *****",
  "universe_domain": "googleapis.com"
}
```

# src\api\index.js

```js
const router = require('express').Router();
const multer = require('multer');
const upload = multer({ storage: multer.memoryStorage() });

const { loginWithEmail, uploadCv, updateUser, getUserDO } = require('../firebase');
const { compareJobDescriptionWithMyCv } = require('../jobHandler');
const { extractTextFromPdf, transcribeCvText } = require('../pdfHandler');

router.post('/upload-resume', upload.single('resume'), async (req, res) => {
    const { uid } = req.query;
    const file = req.file;
    if (!file) return res.status(400).json({ message: 'No resume file provided' });
~~~

~~~
    try {
        const publicUrl = await uploadCv(uid, file);
        const cvText = await extractTextFromPdf(file.buffer);
        const cvSummary = await transcribeCvText(cvText);
        await updateUser(uid, { cvText, cvSummary, cvUrl: publicUrl });
        res.json({ message: 'Resume uploaded successfully', cvUrl: publicUrl });
    } catch (error) {
        res.status(500).json({ message: 'Failed to upload resume', error: error.message });
    }
});

router.get('/cv', async (req, res) => {
    const user = await getUserDO(req.query.uid);
    user ? res.json(user) : res.status(404).json({ message: 'User not found' });
});

router.post('/job/compare', async (req, res) => {
    const { uid, jobDescription, linkedinJobId } = req.body;
    const user = await getUserDO(uid);
    if (!user) return res.status(404).json({ message: 'User not found' });

    const jobs = user.jobs || [];
    const existingJob = jobs.find(job => job.linkedinJobId === linkedinJobId);
    if (existingJob) return res.json({ comparisonResult: existingJob.comparisonResult });

    const comparisonResult = await compareJobDescriptionWithMyCv(jobDescription, user.cvText);
    jobs.push({ jobDescription, linkedinJobId, comparisonResult });
    await updateUser(uid, { jobs });
    res.json({ comparisonResult });
});

router.get('/job', async (req, res) => {
    const user = await getUserDO(req.query.uid);
    user ? res.json(user.jobs || []) : res.status(404).json({ message: 'User not found' });
});

router.delete('/job', async (req, res) => {
    const { uid, linkedinJobId } = req.body;
    const user = await getUserDO(uid);
    if (!user) return res.status(404).json({ message: 'User not found' });

    const updatedJobs = (user.jobs || []).filter(job => job.linkedinJobId !== linkedinJobId);
    await updateUser(uid, { jobs: updatedJobs });
    res.json({ message: 'Job removed' });
});

module.exports = router;

```
~~~