THESIS

Building a chatbot to match jobs and resumes of software engineers.

PANAGIOTIS FOTIADIS

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

May **2024**

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

Building a chatbot to match jobs and resumes of software engineers.

Panagiotis Fotiadis

Supervisor

Dr. Georgia Kapitsaki

Acknowledgments

I would like to extend my deepest gratitude to Dr. Georgia Kapitsaki, whose guidance and expertise have been invaluable throughout the course of this research. Her mentorship was crucial in shaping both the direction and execution of this thesis, and I am deeply appreciative of her support.

I am also grateful to the University of Cyprus for providing the resources and environment necessary to conduct this research.

I must express my profound thanks to my family for their unwavering support and encouragement. Their belief in my abilities and their constant encouragement kept me motivated throughout my academic journey.

Thank you.

Abstract

This thesis aims to create (design, implement, test, and evaluate) a chatbot that aims to match the abilities of a software engineer based on their CV and their GitHub activity with available job advertisements in online sites. The job listings are collected from publication websites for jobs (ergodotisi.com for Cyprus and glassdor.com for USA listings). The matching of CV and job listings is based on the user's previous years of experience, soft skills, hard skills, education, and other personal preferences such as location and company. The creation of a user interface (UI) was essential to ensure accessibility for all users and to facilitate the evaluation process through surveys and user testing.

Contents

CHAPTER 1 Introduction 7
  1.1 Context and purpose of the thesis 7
  1.2 Structure of the text 8
CHAPTER 2 Background 9
  2.1 Evolution of Artificial Intelligence 9
  2.2 Introduction to Chatbots 10
  2.3 Challenges of Employment Searching 11
  2.4 Limitations of Current Job Matching Systems 12
  2.5 Importance of Personalized Job Recommendations 13
  2.6 Principles of Human-Computer Interaction (HCI) and their Importance 13
  2.7 Research Motivation 14
CHAPTER 3 Technologies 16
  3.1 Data Collection 16
    3.1.1 Selenium 16
    3.1.2 BeatifulSoup4 16
    3.1.3 Lightcast 16
  3.2 Backend 17
    3.2.1 MongoDB 17
    3.2.2 Docker 18
    3.2.3 Flask API 19
    3.2.4 JWT 20
    3.2.5 RASA 21
  3.3 Frontend 22
    3.3.1 React 22
  3.4 Hosting 23
    3.4.1 Vercel 23
    3.4.2 Ngrok 24
CHAPTER 4 Design and Implementation 25
  4.1 Introduction 25
  4.2 Implementation of Data Collection 26
  4.3 Implementation of Backend 27
    4.3.1 Docker and MongoDB 27

4.3.2 API 28
4.3.3 RASA...30
4.4 Implementation and Functions of Frontend with React.......................................32
4.4.1 Login/Sign up Page...32
4.4.2 Page Layout...34
4.4.3 Chat Interface 35
4.4.4 Chat History...36
4.4.5 CV analysis 37
4.4.6 Change User Credentials 38
4.4.7 HCI Principles in UI...38
4.5 Example of Usage 41
4.6 Recommended Usage Path for Best Results...43
CHAPTER 5 Evaluation 45
5.1 Survey 45
5.2 Results 46
CHAPTER 6 Future and Conclusion..51
6.1 Summary...51
6.2 Future Work...52
6.3 Conclusions 53
BIBLIOGRAPHY...55
APPENDIX...56
Survey..60

CHAPTER 1 Introduction

1.1 Context and purpose of the thesis

The rise of artificial intelligence (AI) and machine learning (ML) has transformed many sectors, notably the job market. Within this wave of innovation, AI-driven chatbots have become a significant tool for automating and tailoring the job search experience. This thesis focuses on designing, implementing, testing, and evaluating an AI-driven chatbot specifically tailored for software engineers, aiming to bridge the gap between job seekers and potential employers by matching resumes and job listings more efficiently and effectively (Sridevi G.M et al. (2022)).

The idea of this project comes from the observation of the challenges faced by job seekers in the technology sector. Job seekers often find it difficult to sift through numerous job listings to find ones that match their unique skills, experiences, and preferences (Connie R. Wanberg et al. (2020)).

On the recruitment side, companies are increasingly turning to AI-driven tools to optimize the hiring process. AI technologies are applied in numerous stages of recruiting, including writing job ads, screening applicant resumes, and conducting initial interviews via automated systems. Some of the notable AI recruiting tools mentioned include Workable, which offers AI-powered candidate screening and job advertising, and YouTeam, which uses AI for matching candidates to roles based on their skills and experiences. (V.R Uma et al (2023) Derek S. Chapman and Jane Webster (2003))

This thesis needs to design the chatbot's architecture, developing its conversation flow and natural language processing (NLP). In addition, the thesis needs to perform the collection of up-to-date job listings and analyze them to extract specific details. The evaluation of the chatbot involves user testing to assess its usability, effectiveness in matching jobs and resumes of software engineers, and overall user satisfaction. (Chowdhary, K.R. (2020)

The broader significance of this thesis lies in its contribution to enhancing the efficiency of the job search and recruitment process in the technology sector by demonstrating how AI-driven chatbots can be utilized to personalize job searches. Furthermore, the insights gained from this research could inspire further innovations in AI and chatbot technologies, potentially extending their applications beyond job matching to other areas of career development and human resources management.

In conclusion, this thesis represents a step forward in using automation and personalization technologies to address real-world challenges in the job market. By focusing on the specific needs of software engineers and the unique aspects of the technology job market, this project showcases the potential of AI-driven chatbots to transform the way job seekers and employers connect, making the process more efficient, effective, and personalized.

1.2 Structure of the text

This thesis is structured into several chapters, each dedicated to a different aspect of the project, from conception to realization and evaluation.

- Chapter 1 introduces the thesis, outlining the project's objectives and the significance of enhancing the job matching process through AI technologies.
- Chapter 2 delves into the background information necessary for understanding the project, including the evolution of artificial intelligence, an overview of chatbots, the challenges associated with employment searching, the limitations of current job matching systems, the importance of personalized job recommendations, and the motivation behind this research.
- Chapter 3 focuses on the technologies utilized in this project, detailing the methods of data collection (Selenium, BeautifulSoup4, LightCast), the backend technologies (MongoDB, Docker, Flask API, JWT, RASA), and the frontend development with React. This section provides a foundation for understanding the technical aspects of the chatbot's development.
- Chapter 4, Design, and Implementation, describes the process of creating the chatbot, from data collection and backend setup to the frontend interface. It includes sub-sections on the implementation of data collection, backend architecture, API functionality, and the frontend's design and features, such as the login/sign-up page, page layout, chat interface, chat history, CV analysis, and user credential management.
- Chapter 5, Evaluation, discusses the methods used to assess the chatbot's effectiveness and user satisfaction, including a survey and analysis of the results, providing insights into the chatbot's performance and areas for improvement.
- Finally, Chapter 6 concludes the thesis with a summary of the work done, future directions for this research, and overall conclusions drawn from the project. The Bibliography and Appendix provide references and additional materials supporting the thesis.

CHAPTER 2 Background

2.1 Evolution of Artificial Intelligence

The journey of artificial intelligence is a testament to human ingenuity and the quest to create machines that can mimic and surpass human intelligence in certain tasks. This journey, spanning several decades, has reached the highest development in the creation of technologies capable of transforming industries.

The dream of creating intelligent machines began in the 1950s, an era of optimism where the foundational goals and visions of AI were established. Early AI research was driven by the goal of understanding and emulating human cognitive processes. This period saw the development of algorithms and models for problem-solving and symbolic processing, laying the groundwork for future advancements in AI. These foundational years set the stage for AI's application in complex problem-solving tasks. (Vivek Kaul MD et al. (2020))

The subsequent era of AI was dominated by the development of symbolic AI (focuses on the processing and manipulation of symbols or concepts) and expert systems, which attempted to encode expert knowledge into computer systems to make decisions or solve problems within specific domains. This period highlighted the potential of AI to carry out tasks requiring human-like expertise. However, limitations in knowledge representation and reasoning capabilities eventually led to reduced funding and interest in AI research. (Vivek Kaul MD et al. (2020))

The most recent phase in AI's evolution has been marked by significant advancements in machine learning and NLP, fueled by the availability of large datasets and powerful computational resources. This shift from a rule-based to a data-driven approach has unlocked unprecedented capabilities in pattern recognition, decision-making, and natural language processing. This era has witnessed the emergence of chatbots and intelligent assistants capable of understanding and processing human language, making them invaluable tools in various applications. Today, AI-powered chatbots represent the convergence of decades of AI research and development. In the context of this thesis, the chatbot developed to match software engineers with job opportunities is a typical example of AI's potential to revolutionize recruitment. (Hamed Taherdoost et al. (2023))

In conclusion, the evolution of AI from its conceptual beginnings to its current state has directly contributed to the development of sophisticated technologies like recruitment chatbots. These intelligent systems stand as a testament to the progress made in AI and offer a glimpse into the future, where AI's potential to transform industries continues to be realized.

2.2 Introduction to Chatbots

Chatbots represent a breakthrough in artificial intelligence, fundamentally changing how businesses engage with customers. These computer programs simulate human conversation, primarily online, by understanding and responding in natural language. This capability allows chatbots to function effectively across various fields, from customer support to personal assistance, making them a crucial asset in numerous applications. (Tatwadarshi P. Nagarhalli et al. (2020))

The evolution of chatbots dates to the 1960s, with the creation of ELIZA, a primitive chatbot developed at MIT that could mimic the language patterns of a psychotherapist. Since then, technology has grown exponentially, powered by advancements in NLP and machine learning. Modern chatbots can learn from interactions, improving their understanding over time to provide more accurate responses.

Chatbots serve multiple purposes across different industries. In customer service, they offer 24/7 assistance, handling inquiries and resolving issues promptly (for example chatbot.com). In e-commerce, chatbots enhance the shopping experience by offering personalized recommendations and support (for example maisieai.com). In healthcare, they can triage symptoms and provide health-related information (for example https://www.ibm.com/products/watsonx-assistant/healthcare). The applications are virtually limitless, underscoring the versatility of chatbots. (Tatwadarshi P. Nagarhalli et al. (2020))

Implementing chatbots offers numerous benefits. They can significantly reduce operational costs by automating repetitive tasks and handling multiple inquiries simultaneously. Chatbots also improve customer satisfaction through immediate responses and round-the-clock availability. Moreover, they gather valuable data from interactions, providing insights into customer behavior and preferences. (Tatwadarshi P. Nagarhalli et al. (2020))

However, developing effective chatbots requires careful consideration of several factors. The chatbot's design should focus on its purpose and the needs of its users, ensuring seamless and intuitive interaction. It must also possess a robust understanding of natural language to accurately interpret and respond to queries. Privacy and security are paramount, as chatbots often handle sensitive information.

In conclusion, chatbots represent a leap forward in human-computer interaction, offering efficient, scalable, and personalized communication. As AI technology continues to advance, chatbots are expected to become even more sophisticated, further transforming our digital landscape and the way we interact with machines.

2.3 Challenges of Employment Searching

The landscape of job searching has evolved, becoming a complex task filled with an overflow of information included in each job advert that can confuse even the most determined applicants. The challenges during the job search process affect both job seekers and employers.

The digital age has made the job market more accessible, but this convenience comes with a downside, information overload. Job boards, company websites, and social media channels are filled with listings, making it increasingly difficult for seekers to go through and identify opportunities that match their skills. This huge amount of information often leads to decisional problems. (Peter Gordon Roetzel (2019))

A persistent challenge in the job market is the mismatch between the skills job seekers possess and those demanded by employers. Rapid advancements in technology and changing industry trends have exacerbated this gap. Many find themselves caught in a loop, where they cannot get the job without the skills, but they cannot acquire the skills without access to the job. This mismatch not only hinders individual career progression but also impacts businesses struggling to find talent with the right competencies.

For certain positions, especially in high-demanding industries or roles, the competition is overwhelming. The volume of applications for a single opening can be in the hundreds or thousands, making it a daunting task for any one individual to stand out. This high level of competition necessitates not just qualifications and experience but also a unique set of skills in personal branding and networking to gain an edge.

Beyond the logistical and strategic aspects of job searching, the emotional impact cannot be overlooked. The process can be a rollercoaster of hope and despair, with each rejection or ignored application chipping away at one’s self-esteem. The psychological toll of prolonged job searching can lead to stress, anxiety, and depression, affecting overall well-being.

In response to these challenges, it's crucial for job seekers to adopt an adaptive approach to job searching, leveraging technology, expanding networks, and continuously upskilling. For employers, recognizing these problems and implementing more supportive recruitment processes can help bridge the gap between talent and opportunity. As the job market continues to evolve, understanding and addressing these challenges is vital for creating a more efficient and equitable employment landscape.

This thesis introduces a chatbot designed to bridge the gap between software engineers and relevant job opportunities, directly addressing several challenges in employment searching. It streamlines the job search process by automating the matching of candidate profiles with job listings, saving job seekers from the time-consuming task of sifting through countless irrelevant job posts. This targeted approach also benefits employers by connecting them with candidates who are well-suited for their open positions, potentially reducing the time and resources spent on the recruitment process.

2.4 Limitations of Current Job Matching Systems

The recent escalation in the development of artificial intelligence made it possible to create AI tools that help, personalize, and boost the process of finding the best suiting jobs for the seekers. Sush tools are “Simplify” (https://simplify.jobs/), “Sonara.ai” (https://www.sonara.ai/), “Careerflow.ai” (https://www.careerflow.ai/) and “Dream.jobs” (https://dream.jobs/).

	Sonara.ai	Simplify.ai	Careerflow.ai	Dream.jobs	Thesis Imp.
Chatbot Functionality	No	Yes	No	No	Yes
Analyze Resume/CV	Yes	No	Yes	Yes	Yes
Analyze GitHub	No	No	No	No	Yes
Huge variety of jobs in database	No	Yes	No	No	Yes
Takes in consideration hard skills	Yes	Yes	Yes	Yes	Yes
Takes in consideration soft skills	No	No	No	Yes	Yes

Sonara.ai and Careerflow.ai work by analyzing user’s CV for years of experience and hard skills and based on that information filters the job listings that are posted on the side of the employer. Simplify.ai uses a chatbot to collect information such as hard skills, location, salary, and preferred company. This system has a lot of companies uploading their job listings directly to them. Dream.jobs also has a small number of job listings, but the systems CV analyzer takes into consideration hard skills and soft skills.

This thesis implementation of job finder analyzes the users CV and GitHub for hard skills and soft skills and then with the use of chatbot takes users information for desired location, company, type of employment (full time, part time), their years of experience and any extra hard skills and soft skills the user want to add. Then using the collected information filters the job listings that are collected and analyzed from Glassdoor.com and Ergodotisi.com.

2.5 Importance of Personalized Job Recommendations

The essence of personalized job recommendations lies in their ability to significantly increase employment chances and simplify the job-seeking process. By delivering tailored job suggestions that align closely with an individual's skills, experience, and career aspirations, these systems transform the often-daunting task of job hunting into a more manageable and targeted endeavor.

Personalized recommendations understand the exact needs of job seekers, from preferred job locations and desired company cultures to specific role requirements. This understanding ensures that candidates are matched with opportunities that not just fit their qualifications but also their personal preferences and career goals. Consequently, job seekers find themselves applying to positions where they have a higher likelihood of being hired and satisfied in the long term.

For employers, the precision of personalized job recommendations means attracting candidates who are not only qualified but also genuinely interested in what the company has to offer. This alignment between a candidate's aspirations and the company's needs leads to more successful hires and, ultimately, a more productive and engaged workforce.

In essence, personalized job recommendations serve as a bridge between the job market's complexity and an individual's unique career path. They filter out the noise of irrelevant job postings, allowing job seekers to focus their efforts on viable opportunities. The result is a job search process that is not only less time-consuming but also more effective, increasing the overall chances of employment and making the path to the next career step clearer and more accessible. (Qing Zhou et al. (2019))

2.6 Principles of Human-Computer Interaction (HCI) and their Importance

Human-Computer Interaction (HCI) principles are essential to creating interfaces that bridge the gap between complex computational systems and their human users. These principles focus on optimizing the design of interactive systems to ensure they are accessible, intuitive, and efficient. This emphasis on user-centered design is crucial because it enhances the usability of technology, making it more adaptable to varying human needs and preferences. As technology becomes increasingly integrated into everyday life, from smartphones to complex enterprise systems, the importance of HCI principles cannot be overstated.

Adhering to HCI principles can significantly improve user satisfaction and productivity. By designing systems that are easy to navigate and understand, users can complete tasks more quickly and with less frustration. This is particularly important in environments where efficiency and accuracy are paramount, such as in medical or financial software. HCI-driven designs help minimize errors by aligning with natural human tendencies and cognitive processes, reducing the learning curve, and helping prevent costly mistakes that could arise from poorly designed interfaces.

Furthermore, HCI principles promote inclusivity by ensuring that digital products are usable by people from diverse backgrounds, including those with disabilities. This inclusiveness extends the reach of technology, opening opportunities for more people to participate in digital and knowledge-based activities. For example, designing websites that comply with accessibility guidelines means that users with visual impairments can use screen readers to navigate these sites effectively.

In the context of global software deployment, HCI principles are instrumental in accommodating cultural differences that influence how users interact with technology. For instance, color schemes, symbols, and layout preferences can vary significantly between cultures; hence, understanding and integrating these elements into the design process can lead to higher acceptance rates in different regions.

In summary, Human-Computer Interaction principles are not just guidelines but foundational elements that ensure technology serves its primary purpose: to augment human abilities and enhance day-to-day life. As we continue to push the boundaries of what technology can achieve, grounding innovation in HCI principles ensures that these advancements remain humane, considerate, and ultimately more effective. This focus on the human aspect of technology interaction is what will continue to drive user engagement and satisfaction in an increasingly digital future. (Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., & Elmqvist, N. (2016))

2.7 Research Motivation

The motivation behind this research stems from the recognition of several key challenges within the current job market, particularly for software engineers and IT professionals. The rapidly evolving tech industry demands a unique approach to job matching that goes beyond traditional methods. This thesis is driven by a desire to address these challenges through innovative solutions, aiming to enhance the job search and recruitment process in several ways:

There's a noticeable disconnect in the current job market, where employers struggle to find candidates with the right mix of skills and experiences, while job seekers often find it challenging to locate opportunities that match their specific skill set and career aspirations.

This research aims to create a platform that efficiently matches job seekers with relevant opportunities based on a detailed analysis of their skills and preferences.

The utilization of AI and machine learning technologies presents an opportunity to significantly improve the accuracy and relevance of job recommendations. By analyzing vast amounts of data, including CVs, GitHub activities, and job listings, this thesis seeks to develop a system that can make highly personalized job recommendations, thereby increasing the chances of successful employment.

Recognizing the importance of user experience in job search platforms, this research introduces a chatbot as a friendly and accessible interface. The chatbot aims to gather user information in a conversational manner, making the process of inputting preferences and skills more engaging and less tedious.

Ultimately, this thesis is motivated by the vision of a more efficient and effective job market, where the time and effort required to match the right candidates with the right opportunities are minimized. By improving the job matching process, the research aims to contribute to higher employment rates, job satisfaction, and overall productivity in the tech industry.

Through this research, the goal is to explore and validate the potential of AI-driven platforms to revolutionize the way job matching is conducted, offering tangible benefits to both job seekers and employers in the tech sector.

CHAPTER 3 Technologies

3.1 Data Collection

3.1.1 Selenium

Selenium is an open-source tool primarily utilized for automating web browser tasks. It provides a powerful interface for developing automated tests by simulating user interactions in web browsers. In the context of this thesis, Selenium's role was to automate the browsing process to collect job listing data from various online platforms. By using Selenium with Python, made it possible to navigate through the job advertising websites and detect, collect, and analyze the data collected. (https://www.selenium.dev/)

This technique was preferred as the access to API, which is offered by the job advertising websites have many limitations or are non-existent. In addition, the widespread use of security services and firewalls that are integrated into the websites will block any CRUD request sent to them, making the data collection impossible.

3.1.2 BeatifulSoup4

BeautifulSoup4 is a Python library designed to simplify the process of parsing HTML and XML documents. It provides iterating, searching, and modifying the parse tree, making it ideal for web scraping tasks. In the context of this thesis, BeautifulSoup4 was used alongside Selenium to extract specific content from web pages after they were loaded and interacted with by Selenium. While Selenium automated the navigation and interaction with job listing websites, BeautifulSoup4 handled the parsing and extraction of structured data from the HTML content of the pages. This combination allowed for efficient data extraction from complex web pages, where job details are embedded within nested HTML elements. BeautifulSoup4's ability to navigate the DOM tree with ease and extract attributes, text, and other data expediently played a critical role in constructing a comprehensive dataset for the job matching system. This capability enhanced the accuracy and efficiency of data collection, enabling the thesis project to analyze and match job listings more effectively. (https://pypi.org/project/beautifulsoup4/)

3.1.3 Lightcast

Lightcast plays a pivotal role in providing labor market analytics, offering detailed insights that are crucial for understanding workforce trends and employment dynamics. For the thesis project, utilizing Lightcast was instrumental in gathering comprehensive data on job postings, industry trends, salary benchmarks, and skill demands across various regions and sectors. This extensive dataset enabled the chatbot developed in the thesis to offer more accurate and contextually relevant job recommendations.

Integrating Lightcast's analytics into the chatbot's functionality allowed for a dynamic adjustment of the job matching process according to real-time labor market conditions. By leveraging updated information on employer needs and job market fluctuations, the system could better tailor its recommendations to match the qualifications and preferences of job seekers with the specific requirements of available positions. This not only enhanced the precision of the job matches but also improved user satisfaction by connecting them with opportunities that closely aligned with their career goals and skills.

Furthermore, the use of Lightcast in the thesis underscores the importance of data-driven decision-making in the development of employment technologies. The ability to access detailed and actionable labor market data ensured that the job recommendations were not only based on theoretical criteria but were grounded in the practical realities of the job market. Lightcast services were preferred over manual skill taxonomy, as Lightcast updates their data constantly, ensuring that any new hard skill or technology that emerges, is in the database. (https://lightcast.io/)

3.2 Backend

3.2.1 MongoDB

MongoDB is a robust NoSQL database renowned for its excellent performance, high availability, and effortless scalability. It uses a document-oriented data model, which is a flexible, JSON-like format, allowing for varied data structures and rapid development. In the context of the thesis project, MongoDB serves as the backbone for data storage, managing both structured and unstructured data efficiently. This database is particularly well-suited for projects like the job matching chatbot, where diverse datasets, such as user profiles, job listings, and interaction logs, need to be handled seamlessly.

Using MongoDB in the thesis allowed for dynamic queries and real-time data updates, which are essential for maintaining the responsiveness and accuracy of the chatbot. Its schema-less nature enabled quick iterations during development phases, as changes in the data model could be made without significant downtime or restructuring. Moreover, MongoDB's robust indexing capabilities ensure that searches across large datasets are fast and efficient, enhancing the chatbot’s ability to match job seekers with relevant opportunities swiftly. Another significant advantage of MongoDB is its scalability, which is vital for handling the potentially large and growing datasets involved in job matching systems.

Overall, MongoDB's flexible and feature-rich platform supported a more agile development process and provided the performance necessary to handle complex queries and large volumes of data, crucial for the effective functioning of the AI-driven job matching chatbot in the thesis.

A NoSQL database was preferred from a SQL database because the optimization for quick data retrieval that a NoSQL database offers is crucial to eliminate long waiting times for the user until they get their results back. Also, the better scalability options that are available make it possible to handle more users and data volume. (https://www.mongodb.com/)

3.2.2 Docker

Docker is a powerful platform that simplifies the process of developing, shipping, and running applications by using containerization technology. Containers enable developers to encapsulate an application along with all its necessary dependencies into a standardized unit. This ensures consistent functionality across different computing environments. This is particularly beneficial in software development where differences in operating environments can lead to bugs and conflicts, thereby reducing these "it works on my machine" scenarios.

Docker containers are efficient and lightweight because they share the host operating system's kernel and do not need a separate operating system for each application. This leads to greater server efficiency and lowers both server and licensing costs. For developers, Docker offers an easy and predictable way to create and manage isolated environments for their applications. Containers are very portable, which means they can run consistently across any desktop, traditional IT, or cloud infrastructure.

In the context of deploying a software system, such as the thesis project, Docker can be instrumental. It allows developers to create a container with the project's application and all necessary configurations and dependencies. This setup simplifies the process of moving the application between different stages of development and production environments and ensures that all team members are working in the same environment. This consistency eliminates time spent setting up environments and troubleshooting environment-specific issues, allowing the team to focus more on development.

Moreover, Docker's ability to quickly start and stop containers makes it ideal for a development environment that requires setting up and tearing down applications repeatedly. Docker's version control system for containers, with the Docker Hub, enables easy version updates and quick rollbacks to earlier versions if necessary. This capability is critical for continuous integration and continuous deployment (CI/CD) practices, making Docker a valuable tool for developers aiming to improve productivity and streamline operations in their software development lifecycle. For the thesis project, using Docker means a more streamlined deployment and more efficient handling of application development and testing, ultimately leading to a more robust and reliable final product. (https://www.docker.com/)

3.2.3 Flask API

Flask is a lightweight yet robust Python web framework, well-known for its ease of use and flexible design. It allows developers to build scalable web applications quickly and with minimal setup. At its core, Flask handles web requests and responses, forming the backbone of any web application. It provides the necessary tools to create routes, which are Python functions that map to specific HTTP requests. By design, Flask includes only the essentials to get a web application running, offering extensions for additional functionalities like form validation, user authentication, and database integration.

For developers working on web APIs, Flask becomes particularly useful. Flask simplifies the creation of RESTful APIs—services that allow different software systems to communicate over the internet by using standard HTTP methods such as GET, POST, PUT, and DELETE. Flask's ability to seamlessly handle these methods makes it an ideal choice for backend development, where creating efficient, secure, and scalable APIs is crucial. Developers can leverage Flask to serialize data, handle requests, and format responses effectively, ensuring that the frontend of an application can communicate with the backend without issues.

Cross-Origin Resource Sharing (CORS) is a crucial security feature that enables or restricts web pages from making requests to a domain other than the one that served the initial web page. This mechanism is essential in modern web development, particularly when building APIs with Flask, to prevent unauthorized interactions between web resources and services.

In the context of a Flask API, implementing CORS is vital, especially if your API is intended to be accessed by client-side applications hosted on different domains. Without CORS, browsers block frontend JavaScript code from making requests to a different domain than the one that served the web application, adhering to the same-origin policy. This policy is a critical security measure that prevents malicious scripts on one page from obtaining access to sensitive data on another web page through that page's API.

To handle CORS in Flask, developers typically use the flask-cors library, which provides a simple mechanism to allow CORS requests. This library can be easily integrated into any Flask application. By using decorators or global settings, developers can specify which domains are allowed to access the API, which HTTP methods are permitted, and whether credentials, such as cookies or authentication headers, are allowed to be shared across origins.

Moreover, proper configuration of CORS in a Flask API is not just about security but also about accessibility. By enabling specific CORS policies, developers can ensure their APIs are accessible to other services and applications as intended without exposing them to security risks. This functionality is particularly important for APIs intended for wide consumption across various services, enabling a controlled and secure method of resource interaction across the internet.

In summary, CORS is an indispensable feature for securing Flask APIs and managing how resources are accessed and shared across different domains. Its implementation is a best practice that aligns with the overarching goals of modern web security and interoperability, ensuring that APIs deliver their intended functionality safely and effectively to a diverse set of client applications.

In the context of a thesis project like building a job matching system, Flask can manage the server-side logic and database interactions efficiently. It can serve as the interface for handling requests from the front end, processing those requests (e.g., fetching job listings, managing user profiles), and sending appropriate responses back to the client side. Flask's compatibility with various database systems and its lightweight nature ensure that the application remains responsive and efficient, even as it scales. Moreover, Flask supports the development of secure APIs with extensions that can help manage user authentication and data security, essential for maintaining user trust and protecting sensitive information in a job matching platform. Thus, using Flask in such a thesis not only facilitates rapid development and testing but also ensures that the application is robust and ready for real-world deployment. (https://flask.palletsprojects.com/en/latest/api/)

3.2.4 JWT

JSON Web Tokens (JWT) are defined by the open standard RFC 7519 as a compact and self-contained method for securely transferring information between parties in the form of a JSON object. This data is trustworthy and verifiable due to its digital signature. JWTs can be secured using a secret with the HMAC algorithm, or through a public/private key pair employing RSA or ECDSA. They are particularly useful in authentication and authorization processes.

The structure of a JWT makes it an ideal choice for web applications. It consists of three parts: the header, the payload, and the signature. The header typically contains the type of token (JWT) and the signing algorithm. The payload contains the claims, which are statements about an entity (typically, the user) and additional metadata. These claims can include data such as user ID, roles, and other attributes relevant to the user session. The signature ensures that the token hasn't been altered after it was issued and can be verified using the signing key.

In the context of web development and APIs, JWTs facilitate the creation of stateless, scalable authentication mechanisms. When a user logs in, the server generates a token and sends it to the client, which then includes the token in the HTTP header of subsequent web requests. This simplifies the server design because it doesn't need to maintain a session store; each request is self-contained and contains all the necessary data to be authenticated and authorized. This is particularly beneficial in distributed systems were maintaining session state across servers can be challenging.

Using JWT in a thesis project that involves building a job matching system allows for secure handling of user sessions across different components of the system. It supports the scalability of the application by enabling it to handle requests efficiently across multiple servers without session synchronization issues. JWT also adds a layer of security by ensuring that user credentials are not repeatedly sent over the network, reducing the risk of interception. Overall, incorporating JWT helps in building a secure, efficient, and modern web application that aligns with current best practices in software development. (https://jwt.io/)

3.2.5 RASA

RASA is an open-source framework designed specifically for building conversational AI applications, such as chatbots and voice assistants. It provides tools and components to handle both the understanding of user input (Natural Language Understanding, NLU) and the decisions of what to do next (Dialogue Management, DM). What sets RASA apart from other conversational AI platforms is its flexibility and capability to be customized extensively to fit complex, specific, or unusual requirements. It supports sophisticated conversational capabilities using machine learning models that can be trained on your data, allowing for highly personalized interactions.

RASA consists of two main components: RASA NLU and RASA Core. RASA NLU is responsible for understanding user messages, including intent recognition and entity extraction. RASA Core, on the other hand, manages the flow of the conversation, deciding on the next action based on the current state of the dialogue and its training data. The combination of these two components enables developers to create more dynamic and context-aware conversational agents that can maintain meaningful and coherent interactions over extended dialogues.

The framework supports advanced features such as Custom Actions (where the bot can execute code to perform operations like querying a database or calling an API), slot filling (for gathering information from the user across multiple turns of conversation), and form actions (for managing complex data entry tasks). Moreover, RASA is highly scalable and can be deployed in any environment, from a small-scale local setup to a large-scale cloud-based architecture, making it suitable for both development and production use in real-world applications.

Incorporating RASA into a thesis project that focuses on building a job matching chatbot allows for the exploitation of these advanced conversational features. The chatbot can not only respond to user queries but also drive the conversation towards obtaining necessary information for job matching, such as skills, preferences, and previous experiences, using an engaging and interactive dialogue system. RASA's ability to learn from each interaction and adapt to users' individual communication styles and preferences enhances the effectiveness and user satisfaction of the chatbot, making it a powerful tool in the deployment of interactive AI systems. (https://rasa.com/)

3.3 Frontend

3.3.1 React

React is a robust JavaScript library that excels in creating user interfaces, especially renowned for effectively building single-page applications (SPAs). It was developed by Meta and has quickly become a favorite among front-end developers due to its component-based architecture. This design allows developers to create encapsulated components that handle their own state, which can then be combined to form complex interfaces. Each component maintains its state and properties, simplifying data management across the application.

One of the key features of React is its virtual DOM. Instead of directly manipulating the browser's DOM, React constructs a virtual DOM in memory. Whenever a component’s state changes, React updates this virtual DOM first. It then employs a diffing algorithm to calculate the minimal updates needed for the real DOM, reducing direct DOM manipulation, and enhancing both performance and user experience.

React also benefits from substantial community support and a strong ecosystem, offering numerous tools and extensions like React Router for navigating SPAs. Additionally, React's compatibility with various backend technologies and frameworks makes it a versatile option for both front-end and full-stack development.

Using React in a thesis project, especially one that involves building complex, interactive user interfaces like a job finder chatbot, provides numerous benefits. React’s modular structure ensures that the user interface components are reusable and maintainable, which is crucial for scaling the project. Moreover, its efficient update mechanism improves the responsiveness and fluidity of the application, enhancing user engagement and satisfaction. This aligns perfectly with the needs of modern web applications that demand high interactivity and dynamic content management without sacrificing performance. (https://react.dev/)

3.4 Hosting

3.4.1 Vercel

Vercel is a sophisticated cloud platform engineered to enhance the development, deployment, and scaling of contemporary web projects, with a focus on frameworks like Next.js, React, and Angular. It streamlines the workflow for developers from the initial stages of development through to production, providing a frictionless experience.

The platform automates the build and deployment processes each time new code is pushed to a linked Git repository, supporting platforms like GitHub, GitLab, and Bitbucket. This feature facilitates easier collaboration on projects. Vercel ensures operational reliability with features such as instant rollbacks to previous deployments, which allow for swift recovery from deployment issues.

A key feature of Vercel is its global Content Delivery Network (CDN), which delivers static assets and dynamic content from locations near the user, drastically improving load times. The platform enhances security with automatic SSL certificates, which secure connections seamlessly.

Moreover, Vercel includes advanced logging capabilities, which are crucial for debugging and monitoring the health of applications. Developers can access detailed logs that track everything from server errors to API response times, helping to identify and resolve issues quickly.

Integrated directly into the dashboard, Vercel's analytics and performance monitoring tools provide developers with critical insights into their applications’ performance. This helps in making data-driven optimizations to improve efficiency and user experience. These comprehensive features make Vercel an attractive option for developers aiming to rapidly deploy and efficiently manage high-performance web applications. (https://vercel.com/)

3.4.2 Ngrok

Ngrok is a versatile tool that allows developers to expose local servers behind NATs and firewalls to the public internet over secure tunnels. This is particularly useful for testing and developing web applications that need to interact with external services and APIs.

Using Ngrok, developers can easily create a secure tunnel to a local web server by running it on the command line. This assigns a public URL to your local server, which can be accessed from anywhere over the internet. The process involves forwarding traffic from an Ngrok domain to your local development environment. For example, if you are running a local API on port 5000, Ngrok can generate a public HTTPS URL that forwards to this local server, allowing remote clients and services to interact with your API as if it were hosted on a live server.

This capability is incredibly useful for API development and testing. Developers can share the URL generated by Ngrok with external stakeholders (like frontend teams or mobile app developers), who can then interact with the backend API without any deployment. It also facilitates real-time debugging and testing of webhooks and other services that require public exposure.

Additionally, Ngrok provides detailed introspection capabilities. Developers can inspect all HTTP traffic passing through the tunnel, view the request and response data, replay requests for testing, and debug the process. This makes it an invaluable tool for API development workflows, where understanding and logging the HTTP traffic is crucial for troubleshooting and optimization.

Overall, Ngrok is a powerful tool that simplifies the process of connecting local development servers to the public internet, providing a practical, real-time solution for testing, and developing APIs in a real-world environment without the need for deployment. (https://ngrok.com/)

CHAPTER 4 Design and Implementation

4.1 Introduction

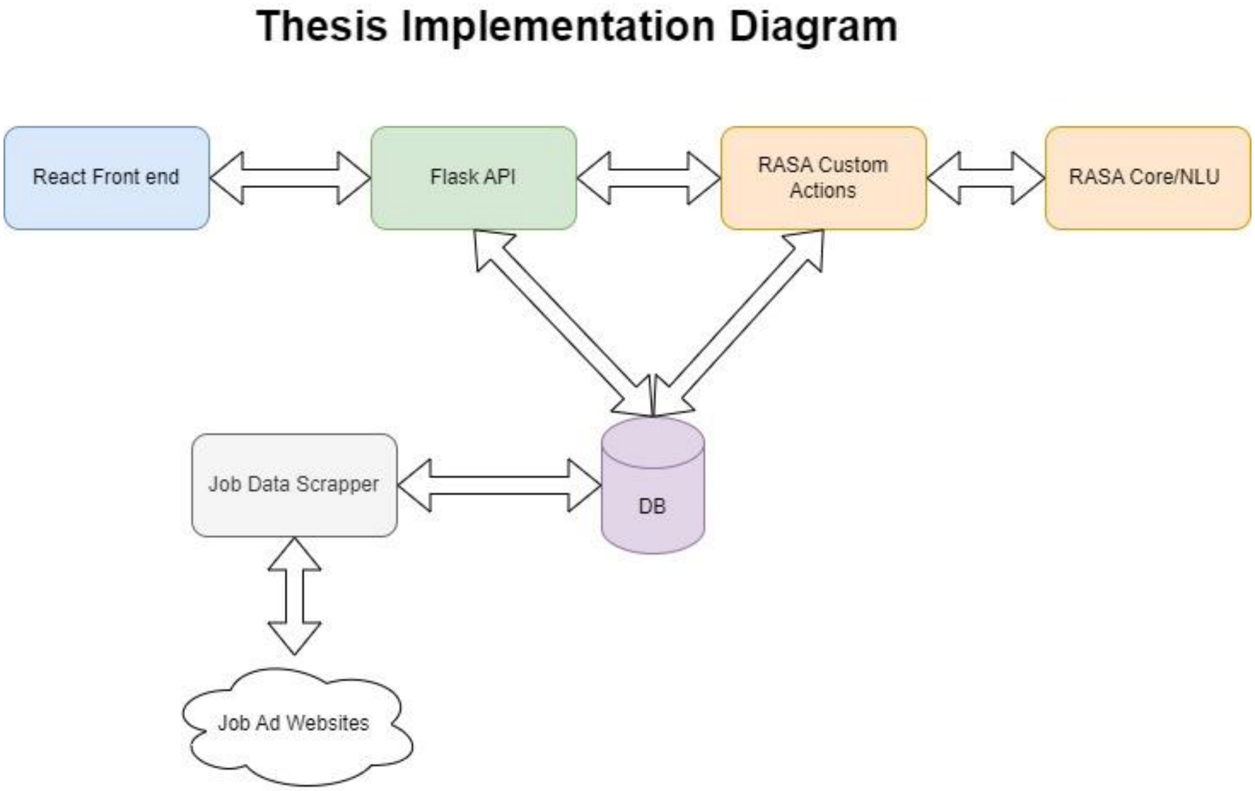


Figure 1

Thesis Implementation Diagram

This thesis encompasses four interconnected components essential for the operation of a job-finding application. The "React Front End" serves as the graphical user interface, facilitating user interaction. The "Flask API" functions as a CRUD (Create, Read, Update, Delete) API, ensuring smooth data operations. The "Rasa Custom Actions" and "Rasa Core/NLU" segments form the backbone of the Rasa chatbot model, enabling sophisticated dialogue management and natural language understanding. The "Job Data Scraper" is tasked with scraping and analyzing job advertisements from websites like "Glassdoor.com" and "Ergodotisi.com", and it stores this data in the "DB" MongoDB database.

To maintain stateless operations, both the Flask API and Rasa Custom Actions are directly connected to the database for necessary user data storage. The front end interacts with both Rasa and the database through the Flask API, which is secured with JWT authentication. To ensure the database always has up-to-date job listings, the Job Data Scraper has the potential to automatically run weekly, collecting, analyzing, and storing the latest job advertisement data for access by the other components.

4.2 Implementation of Data Collection

The Data Collection component of this project utilizes Python, Selenium, BeautifulSoup4, and the Lightcast API to effectively scrape job listings from websites. To initiate the scraper, run python data_extractor_manager.py from the command prompt. This command triggers the import of two specific modules, glassdor_scrapper and ergodotisi_scrapper, which are tailored for scraping Glassdoor.com and Ergodotisi.com respectively.

The selection of Glassdoor and Ergodotisi as primary sources for the data collection was influenced by their structural similarities and extensive listings. Both platforms present job advertisements in a concise list format, with each listing linked to a separate URL detailing the full job description. This consistent structure not only facilitated the reuse of code but also expedited the development process. Glassdoor, being one of the leading job advertisement websites in the United States, and Ergodotisi, holding a similar standing in Cyprus, were ideal for compiling a comprehensive and inclusive dataset.

Other platforms such as LinkedIn and Monster were considered; however, they presented specific challenges that influenced their exclusion. LinkedIn features a sophisticated search engine and complex website structure, which posed significant challenges in scraper development. Conversely, Monster was found to have a limited number of listings for software engineering positions, which would not contribute to a robust dataset. Thus, the choice of Glassdoor and Ergodotisi was deemed most suitable for the objectives of this study.

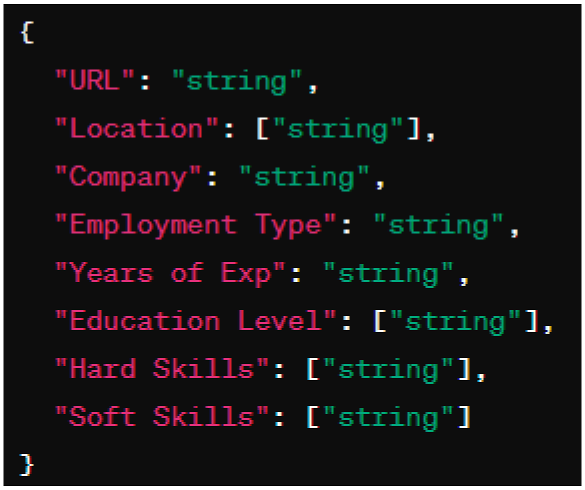
Each scraper is designed with three main parts: url_scrapper, description_scrapper, and lightcast_skill_extractor. Both target websites feature a similar structure, displaying job advertisement previews on the main page along with URLs. When a user clicks on a preview, it directs them to a page containing comprehensive job details including the company, location, and job description.

The url_scrapper is multithreaded to enhance performance. Each thread corresponds to a specific state, navigating through the website's main page using Selenium to apply location filters. BeautifulSoup4 is then used to extract the URLs of job listings, which are subsequently stored in a CSV file alongside their respective locations.

Following this, the description_scrapper—also multithreaded—operates by launching a Selenium WebDriver for each URL gathered previously. It uses BeautifulSoup4 to collect details such as the company, employment type, years of experience, education level, and job description. These details are then saved in another CSV file.

Finally, the lightcast_skill_extractor utilizes the Lightcast API by sending the collected job descriptions to https://emsiservices.com/skills/versions/latest/extract. This API returns the relevant soft and hard skills found within each description. All the scraped data is compiled into a JSON file according to a predefined schema (Figure 2) and then stored in a MongoDB database.

This streamlined process ensures efficient and organized data collection, suitable for subsequent analysis and application use.



```
{
  "URL": "string",
  "Location": ["string"],
  "Company": "string",
  "Employment Type": "string",
  "Years of Exp": "string",
  "Education Level": ["string"],
  "Hard Skills": ["string"],
  "Soft Skills": ["string"]
}
```

Figure 2

Job listings JSON schema

4.3 Implementation of Backend

4.3.1 Docker and MongoDB

Docker is utilized to manage two MongoDB databases: one dedicated to job data and the other for user data. To establish these databases, containers are configured through separate docker-compose.yml files, situated in the docker/jobs and docker/users directories respectively. Each container is set to operate on unique ports to ensure they are accessible independently; the job data container uses port 27017, while the user data container uses port 27018. Furthermore, each database is associated with distinct Docker volumes, which isolates their stored data and prevents any overlap. This setup not only enhances data management by segregating user and job information but also optimizes performance by allowing the containers to run concurrently without interference. To initiate the containers, the command docker-compose up -d must be executed within each respective directory, effectively bringing up the MongoDB instances as configured. This approach provides a robust and scalable infrastructure for managing distinct datasets efficiently in a controlled environment. The docker-compose.yml for each database can be seen at Appendix "Job Docker" and "User Docker".

4.3.2 API

The system's API, developed using the Flask framework in Python, is located within the /mongodb-api/mongodb_api.py file. The API is organized into endpoints that serve two main purposes: interfacing with the frontend and facilitating backend interactions with the RASA chatbot.

For frontend interactions, the API provides several key endpoints:

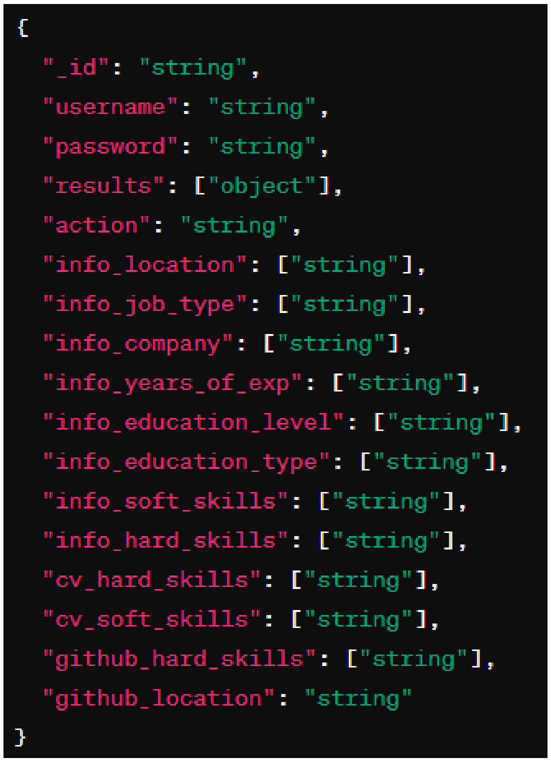
- /login: Manages user authentication and generates JWT tokens.
- /register: Handles user registration and records user data in the database using a predefined JSON schema.
- /api/messages: Enables message transmission to the chatbot.
- /get-username: Retrieves the user's username using a provided JWT token.
- /save-chat: Archives chat sessions upon their completion.
- /api/chat/<chatId>: Retrieves specific chat details by chat ID.
- /api/change-password and /api/change-username: Allow users to update their password and username, respectively.
- /info_github and /get_github: These endpoints interact with the user's GitHub account to fetch programming languages and analyze data.
- /reset_rasa: Resets user data related to a specific chat session.
- /analyze-text: Analyzes a user's CV to extract soft and hard skills.
- /get-cv-soft-skills and /get-cv-hard-skills: Return the soft and hard skills identified in the user's CV, respectively.
- /get-github-location and /set_github_location: Manage the extraction and storage of the user's location based on their GitHub profile.

For Rasa interactions, the API provides:

- /actions/<user_id>/<action>: Stores user's next action path in chatbot
- /info_location/<user_id>: Stores user's preferred location that provided to chatbot

- /info_job_type/<user_id>: Stores user's preferred job type that provided to chatbot
- /info_company/<user_id>: Stores user's preferred job type that provided to chatbot
- /info_years_of_exp /<user_id>: Stores user's years of experience that provided to chatbot
- /info_education/<user_id>: Stores user's education that provided to chatbot
- /info_soft_skills/<user_id>: Stores user's soft skills that provided to chatbot
- /info_hard_skills/<user_id>: Stores user's hard skills that provided to chatbot
- /get_rasa/<user_id>: Returns user’s information that collected from chatbot
- /get_action/<user_id>: Returns user's next action path in chatbot
- /get_results: Returns job listings that have close similarity to the user’s information given in chatbot. (Appendix)

These endpoints are essential for the dynamic functionality of the system, ensuring seamless user interactions and effective data management. They support a variety of operations from user management to detailed personal data handling, making the system robust and user-friendly.



```
{
  "_id": "string",
  "username": "string",
  "password": "string",
  "results": ["object"],
  "action": "string",
  "info_location": ["string"],
  "info_job_type": ["string"],
  "info_company": ["string"],
  "info_years_of_exp": ["string"],
  "info_education_level": ["string"],
  "info_education_type": ["string"],
  "info_soft_skills": ["string"],
  "info_hard_skills": ["string"],
  "cv_hard_skills": ["string"],
  "cv_soft_skills": ["string"],
  "github_hard_skills": ["string"],
  "github_location": "string"
}
```

Figure 3

User’s JSON schema

4.3.3 RASA

To implement RASA Core/NLU, adjustments were made to several key configuration files: endpoint.yml, config.yml, domain.yml, nlu.yml, and stories.yml. Each of these files serves a specific purpose in the setup and functionality of the RASA framework. The endpoint.yml must include the custom actions endpoint of Rasa (Appendix Rasa Action Endpoint). The config.yml (Appendix Rasa Config) which contains pipeline with a WhitespaceTokenizer, splits text into tokens based on whitespace, setting the stage for further text processing. This is followed by the LexicalSyntacticFeaturizer, which enriches tokens with lexical and syntactic features that are helpful for understanding context and grammatical relationships.

The CountVectorsFeaturizer appears twice: the first instance operates at the word level, while the second is configured to analyze character n-grams from 1 to 4 characters long within word boundaries (char_wb), enhancing the model's ability to recognize patterns within words.

The DIETClassifier is a crucial component, designed for dual intent and entity recognition with the ability to classify both simultaneously. It's configured for a substantial 200 epochs, indicating a deep learning phase to accurately predict intents and entities in user messages.

The EntitySynonymMapper helps in mapping different phrasings of the same entity to a single canonical form, thereby standardizing entity values. The RegexEntityExtractor utilizes regular expressions to identify entity values from text, benefiting from lookup tables and being case insensitive to enhance flexibility.

The ResponseSelector is tailored for selecting appropriate responses based on the learned patterns, trained for 100 epochs to optimize decision-making processes.

Under the policies section, the MemoizationPolicy ensures the model remembers specific stories during training, while the TEDPolicy (Transformer Embedding Dialogue) drives the decision-making in dialogues, set with a maximum history of 15 and trained over 200 epochs to capture complex conversational contexts.

This structured approach, combining tokenization, feature extraction, and sophisticated classification and decision-making mechanisms, empowers the application to handle intricate user interactions effectively, recognizing intents and extracting entities with high precision.

The nlu.yml file contains lookup tables with expected words that the user can give and examples of answers that the user might give using the lookup tables words. This file is used to create natural language model which the chatbot is based on recognizing the user’s answers and decide the chatbot story flow.

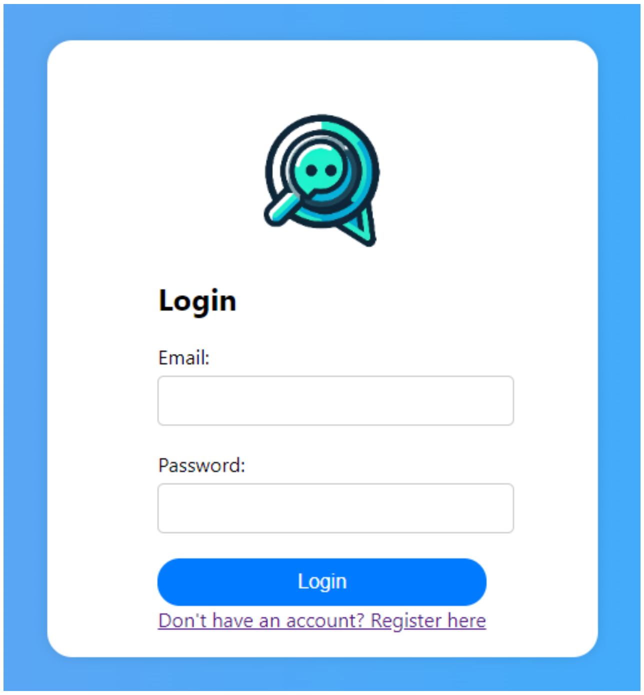
The domain.yml contains intents and entities names that exist in nlu.yml, the responses that the chatbot can give on each user’s input (multiple responses can be added and the chatbot will choose randomly one each time) and the actions names that exist in custom actions.

Stories.yml is a diagram of the chat flow and how the chatbot must proceed after each question is answered and what information should expect from the user to give. Also, custom actions can be accessed from stories.yml. The information collected from chatbot are location, job type, preferred company, years of experience, education, hard skills, and soft skills.

To effectively implement Rasa Custom Actions, the actions.py file was refined to act as a stateless API, a critical feature that supports multiple concurrent user interactions. This file is instrumental in managing the conversation flow to ensure that user inputs do not disrupt the progression of the dialogue. It functions by preserving the current phase of a user's interaction within the story flow. During each interaction, actions.py evaluates the entities detected by Rasa Core to determine if they adequately address the conversational prompts. If the entities align with the required responses, the script permits the conversation to proceed; otherwise, it prompts the user to provide the necessary information again, repeating this process until the correct data is gathered. Once the required information is successfully collected, it is temporarily stored in the user database. This data is held only until the conclusion of the story, at which point the results are compiled and sent to the user, ensuring a seamless and efficient user experience.

4.4 Implementation and Functions of Frontend with React

4.4.1 Login/Sign up Page.



Login

Email:

Password:

Figure 4

Login Page

For the user to authenticate themselves and generate a new JWT token to use the chatbot, they must fill in the email and password details that they apply during the registration process. If the user does not have a registered account, they can navigate to the Registration page by clicking on “Don’t have an account? Register here”.

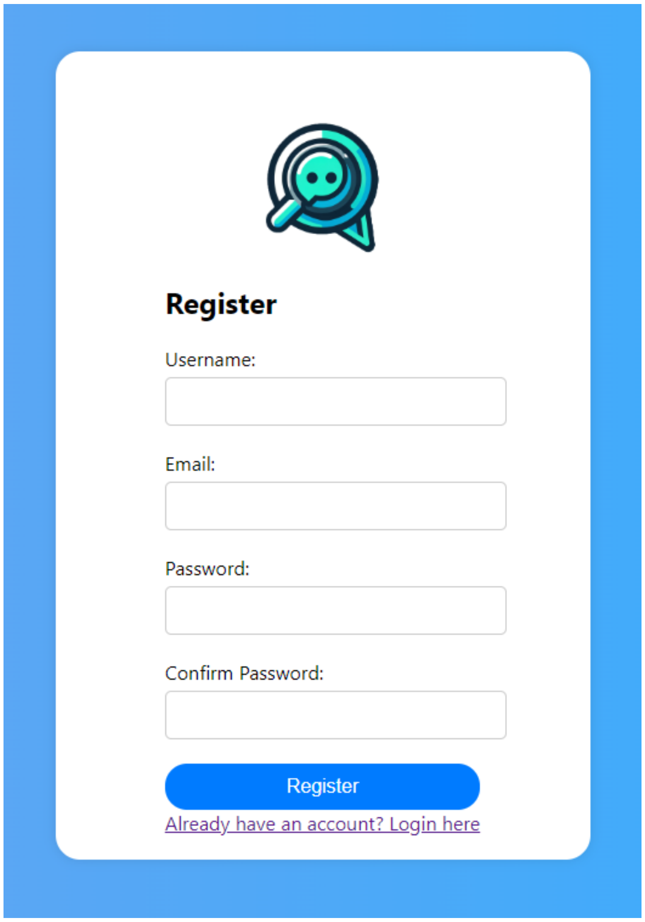


Figure 5

Sign Up Page

The user can create a new account by filling in the form on the Registration page. When data is provided correctly, a notification will appear informing the user of the successful registration and then it will automatically redirect to the login page. The backend will create a new user in the database with data like in Figure 3.

4.4.2 Page Layout

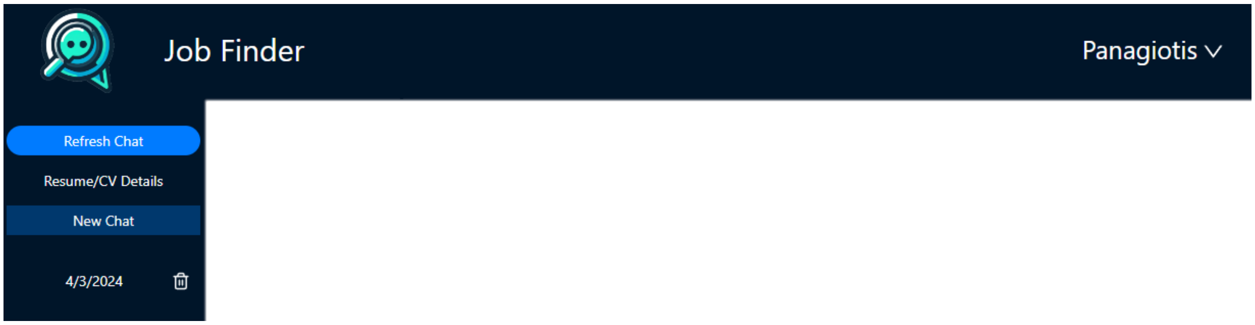


Figure 6

Page Layout

When user is logged in the system the page layout will allow the user to Refresh the current chat by clicking “Refresh Chat”, create a new chat by clicking “New Chat”, upload their CV and GitHub at “Resume/CV Details” and navigate to their old chat that are displayed on the sidebar with their date of creation. The ability to delete old chats is possible by clicking the Trash can icon next to them. User’s temporary data that are stored during a chatbot use are deleted every time the user logs in or clicks on Refresh chat. Ensuring that there are no old or irrelevant temporary data in the database guarantees accurate results from any new chats the user makes.

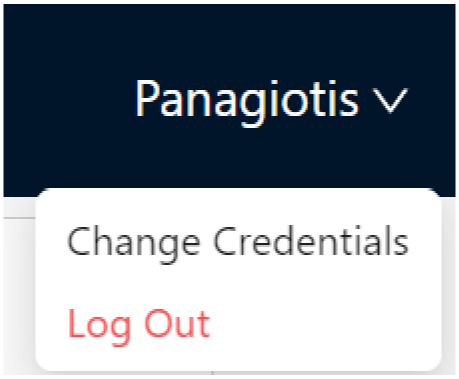
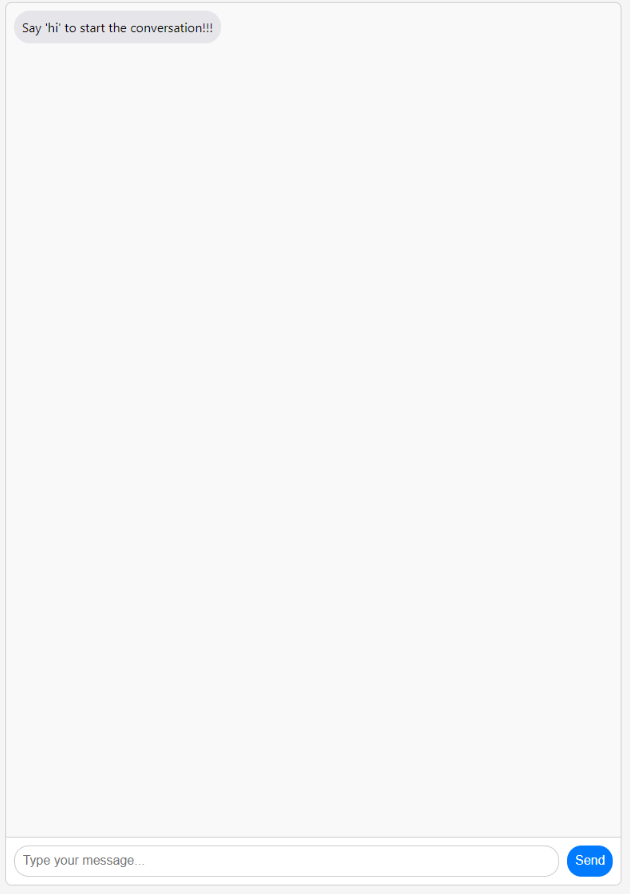


Figure 7

Header Options

By clicking on the User’s name that appears on the Header of the page, a dropdown will appear displaying the options of changing user’s credentials and logging out. Log out option deleted the JWT token from the browser and navigate to Login Page.

4.4.3 Chat Interface



Say 'hi' to start the conversation!!!

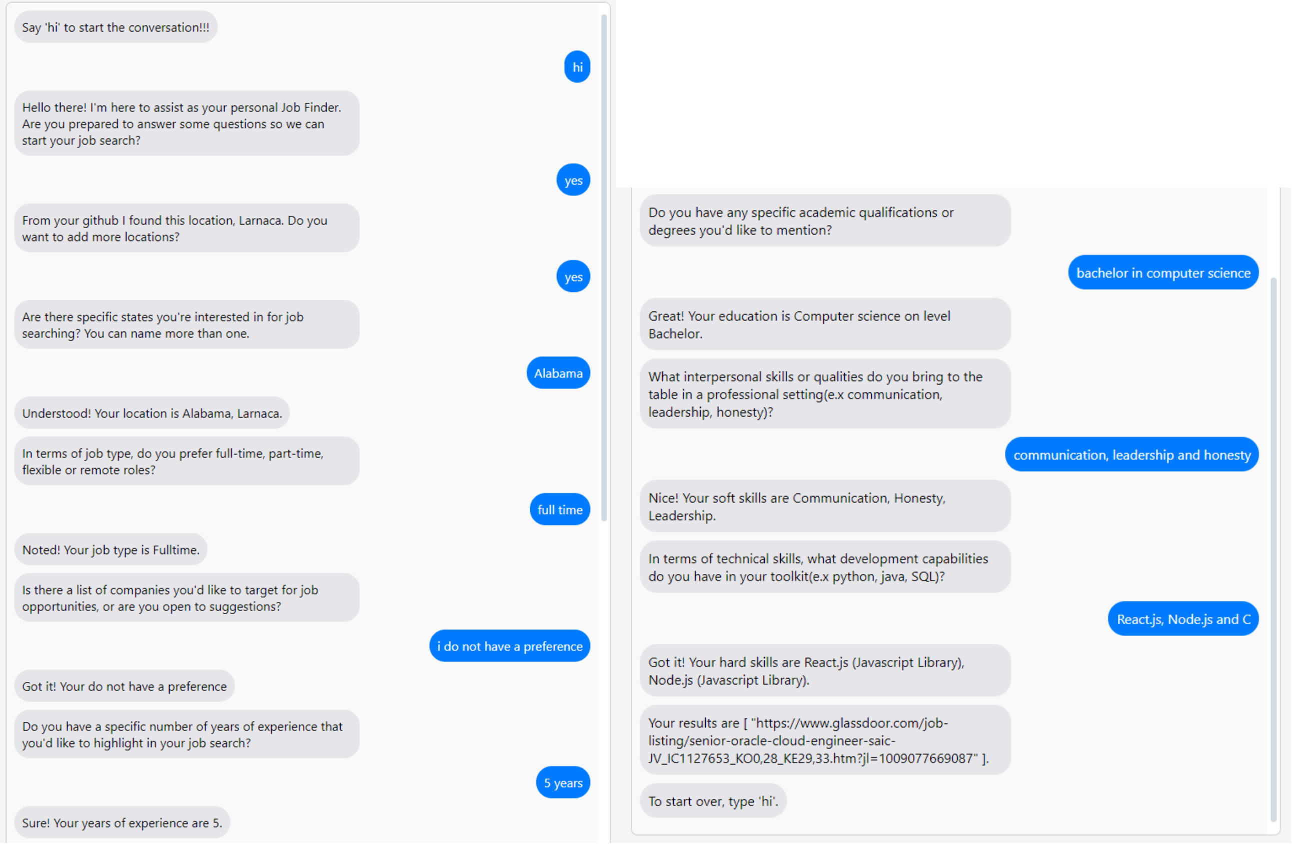
Figure 8

Chat Interface

Users can interact with chatbot through the interface in Figure 8. By sending a greeting message the chatbot initiates the story flow from stories.yml included in RASA. The user can respond to chatbot’s questions by typing the answer in “Type your message...” and clicking “Send”. When “Send” is clicked the button will get disabled until the chatbot respond. Disabling the button will verify that no more than one message is sent before an answer from chatbot. The messages are not sent directly to RASA, but they pass through the Flask API to authenticate the JWT token and then send to RASA with the user’s username. By providing RASA with a unique username, it will be able to handle multiple messages from multiple users and handle them concurrently.

4.4.4 Chat History

When a user opens an old chat, they can view the conversation, but they cannot send another message. Figure 9 is sample of an old chat and how the chat UI looks like.



Say 'hi' to start the conversation!!!

hi

Hello there! I'm here to assist as your personal Job Finder. Are you prepared to answer some questions so we can start your job search?

yes

From your github I found this location, Larnaca. Do you want to add more locations?

yes

Are there specific states you're interested in for job searching? You can name more than one.

Alabama

Understood! Your location is Alabama, Larnaca.

In terms of job type, do you prefer full-time, part-time, flexible or remote roles?

full time

Noted! Your job type is Fulltime.

Is there a list of companies you'd like to target for job opportunities, or are you open to suggestions?

i do not have a preference

Got it! Your do not have a preference

Do you have a specific number of years of experience that you'd like to highlight in your job search?

5 years

Sure! Your years of experience are 5.

Do you have any specific academic qualifications or degrees you'd like to mention?

bachelor in computer science

Great! Your education is Computer science on level Bachelor.

What interpersonal skills or qualities do you bring to the table in a professional setting(e.x communication, leadership, honesty)?

communication, leadership and honesty

Nice! Your soft skills are Communication, Honesty, Leadership.

In terms of technical skills, what development capabilities do you have in your toolkit(e.x python, java, SQL)?

React.js, Node.js and C

Got it! Your hard skills are React.js (Javascript Library), Node.js (Javascript Library).

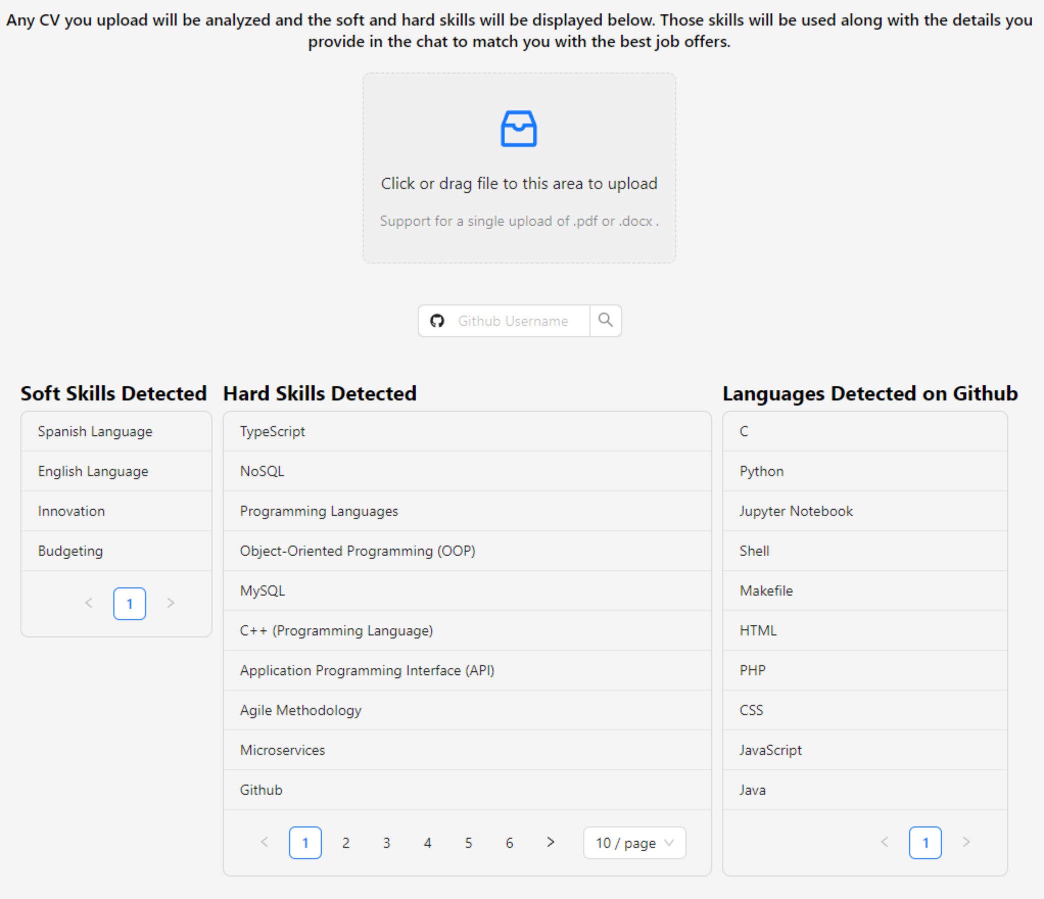
Your results are ["https://www.glassdoor.com/job-listing/senior-oracle-cloud-engineer-saic-JV_IC1127653_KO0,28_KE29,33.htm?jl=1009077669087"].

To start over, type 'hi'.

Figure 9

Chat Sample

4.4.5 CV analysis



Any CV you upload will be analyzed and the soft and hard skills will be displayed below. Those skills will be used along with the details you provide in the chat to match you with the best job offers.

Click or drag file to this area to upload

Support for a single upload of .pdf or .docx .

Soft Skills Detected
Spanish Language
English Language
Innovation
Budgeting

Hard Skills Detected
TypeScript
NoSQL
Programming Languages
Object-Oriented Programming (OOP)
MySQL
C++ (Programming Language)
Application Programming Interface (API)
Agile Methodology
Microservices
Github

Languages Detected on Github
C
Python
Jupyter Notebook
Shell
Makefile
HTML
PHP
CSS
JavaScript
Java

Figure 10

CV analysis

When the user uploads their CV/Resume the backend analyzes the document using Lightcast and returns the detected Soft and Hard skills. By giving their GitHub username, the backend uses GitHub's API to detect all programming languages that are in the repositories and the user's location if exists. The collected data appears in tables as shown in Figure 10. Those collected data is used in addition to those collected through the chatbot conversation to find the most fitting results.

4.4.6 Change User Credentials

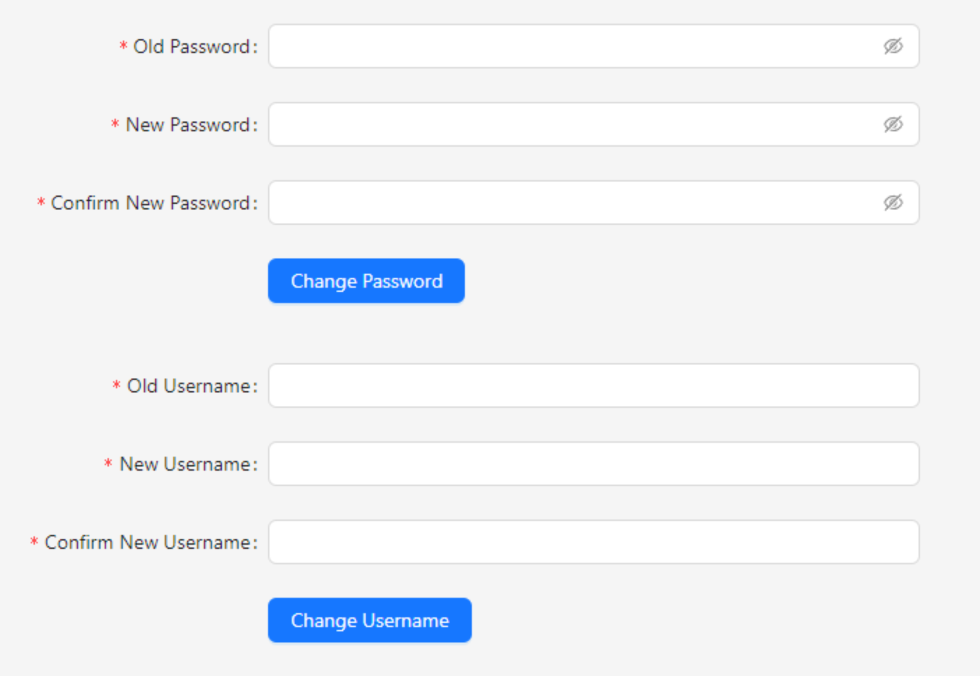


Figure 11

Change User Credentials Page

The user can change their username or password by providing their old one, give their new one and confirm it. Then by clicking the appropriate button if new data is confirmed the backend will change the old value to the new value in the database and the appropriate notification will appear on user’s screen.

4.4.7 HCI Principles in UI

In the design of the "Job Finder Chatbot," special attention was given to human-computer interaction (HCI) principles to ensure an optimal user experience. This chapter explores how the chatbot's user interface (UI) adheres to established HCI guidelines, enhancing usability, accessibility, and user satisfaction.

Application of HCI Principles in the Chatbot Interface

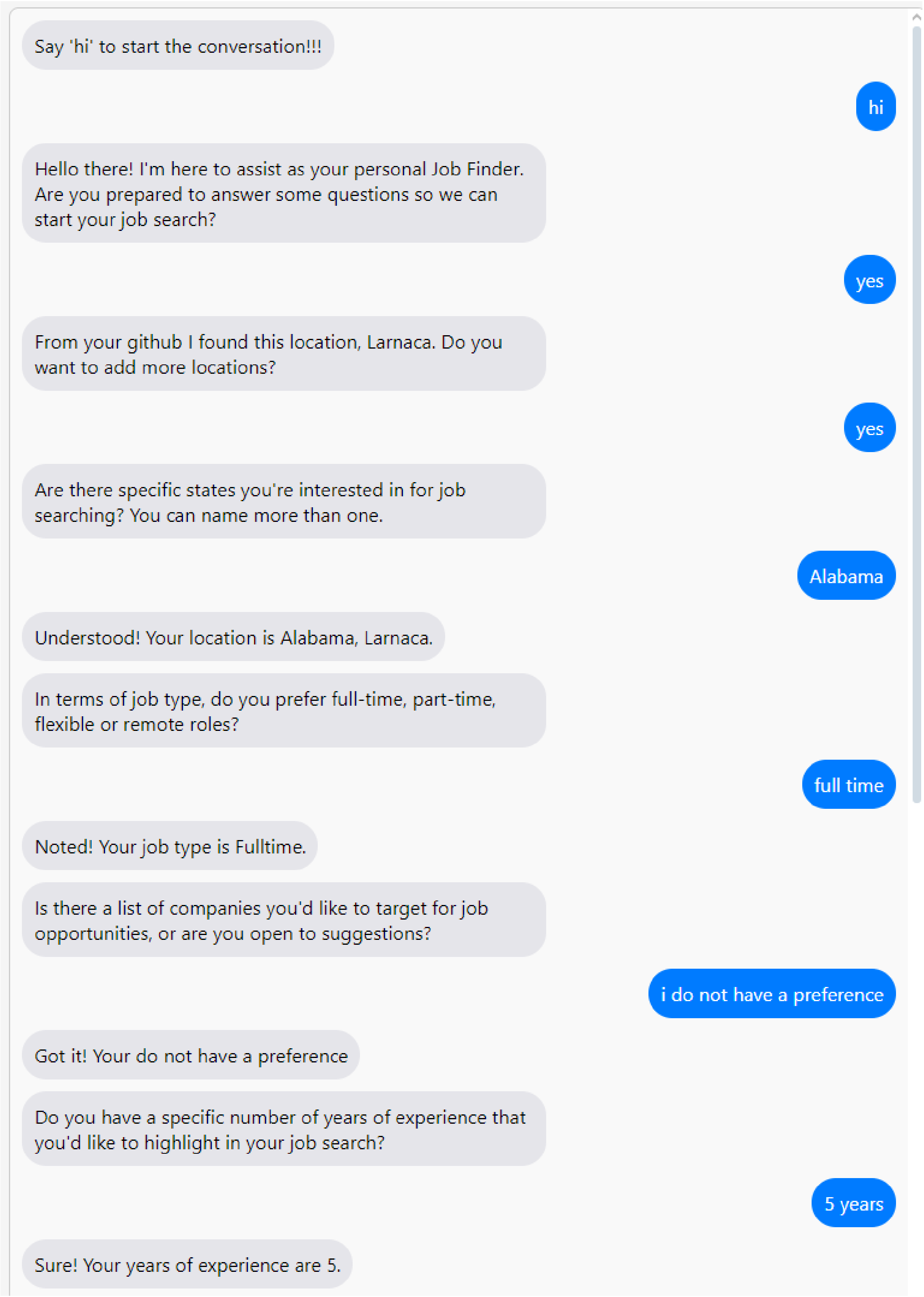
1. Visibility of System Status
    - Feedback: The chatbot provides immediate feedback following user actions, ensuring users are always informed about what is happening. For example, when a job search is initiated, the user sees a loading indicator along with a textual update on the progress.
    - Status Updates: During longer processes, such as analyzing a CV, periodic updates keep the user informed, preventing confusion and frustration.
2. Match Between System and the Real World
    - Language and Icons: The chatbot uses familiar language and intuitive icons that mimic real-world conventions, making the system understandable and relatable. For instance, job categories are represented with recognizable icons like a briefcase for professional jobs or a wrench for technical jobs.
3. User Control and Freedom
    - Exit: At any point, users can exit out of the current interaction pathway or reset the chat, giving them freedom and control over the use of the application.
4. Consistency and Standards
    - UI Consistency: The chatbot maintains a consistent UI across all platforms, whether accessed via mobile or desktop, ensuring that users have a seamless experience regardless of the device used.
    - Interaction Consistency: Interaction elements such as buttons, links, and menu structures are consistent in appearance and behavior across different parts of the application.
5. Error Prevention
    - Input Validation: The system minimizes potential errors by validating user inputs in real-time. For example, if a user enters an invalid email address, the system immediately prompts correction before proceeding.
    - Guided Actions: The chatbot guides users through the process with clear, step-by-step instructions, reducing the risk of errors.
6. Aesthetic and Minimalist Design
    - Clean Layout: The UI is uncluttered, focusing the user’s attention on essential elements by avoiding irrelevant information.

   - Aesthetic Elements: The interface employs a pleasing color scheme and typography that enhance readability and user engagement without sacrificing functionality.

7. Help Users Recognize, Diagnose, and Recover from Errors
   - Error Messaging: Error messages are expressed in plain language, precisely indicating the problem.

The "Job Finder Chatbot" demonstrates a strong adherence to HCI principles, ensuring that the user interface is intuitive, efficient, and user-friendly. This commitment to excellent user experience design not only enhances user satisfaction but also contributes to the overall effectiveness of the job search process. As technology and user expectations evolve, continual evaluation and adaptation of these HCI principles will remain essential to maintaining and improving the chatbot's usability and appeal.

4.5 Example of Usage



Say 'hi' to start the conversation!!!

hi

Hello there! I'm here to assist as your personal Job Finder. Are you prepared to answer some questions so we can start your job search?

yes

From your github I found this location, Larnaca. Do you want to add more locations?

yes

Are there specific states you're interested in for job searching? You can name more than one.

Alabama

Understood! Your location is Alabama, Larnaca.

In terms of job type, do you prefer full-time, part-time, flexible or remote roles?

full time

Noted! Your job type is Fulltime.

Is there a list of companies you'd like to target for job opportunities, or are you open to suggestions?

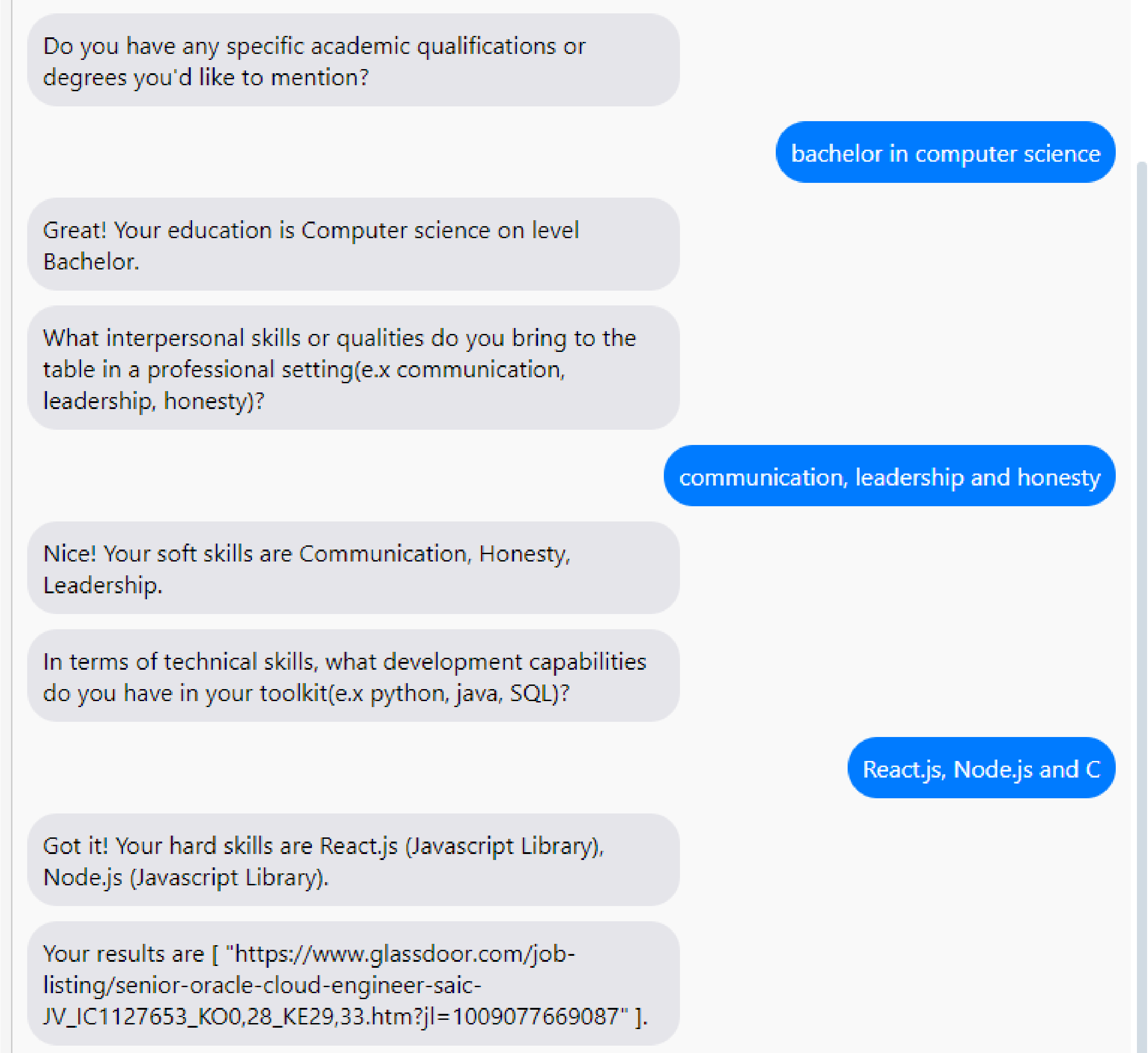
i do not have a preference

Got it! Your do not have a preference

Do you have a specific number of years of experience that you'd like to highlight in your job search?

5 years

Sure! Your years of experience are 5.


Do you have any specific academic qualifications or degrees you'd like to mention?

bachelor in computer science

Great! Your education is Computer science on level Bachelor.

What interpersonal skills or qualities do you bring to the table in a professional setting(e.x communication, leadership, honesty)?

communication, leadership and honesty

Nice! Your soft skills are Communication, Honesty, Leadership.

In terms of technical skills, what development capabilities do you have in your toolkit(e.x python, java, SQL)?

React.js, Node.js and C

Got it! Your hard skills are React.js (Javascript Library), Node.js (Javascript Library).

Your results are ["https://www.glassdoor.com/job-listing/senior-oracle-cloud-engineer-saic-JV_IC1127653_KO0,28_KE29,33.htm?jl=1009077669087"].

Figure 12

Chat Example

User Input Collection: The chatbot efficiently collects detailed user inputs across several categories critical to job matching, including geographic location, job type preference (full-time), years of experience, educational background, soft skills, and technical skills. This demonstrates the chatbot's capability to engage users in a comprehensive dialogue, ensuring that all relevant information is gathered to support personalized job recommendations.

Natural Language Processing (NLP) Performance: The chatbot successfully interprets and processes user responses, such as interpreting "Alabama" as a location and recognizing "5 years" as the amount of experience. This indicates that the NLP component is effectively tuned to understand and extract meaningful information from casual user inputs.

Data Integration and Application: The chatbot demonstrates a well-integrated system that takes user data (e.g., skills and experience) and maps it directly to job postings, as shown by the link to a job listing on Glassdoor tailored to the user's profile. This suggests that the backend systems are effectively leveraging the collected data to fetch relevant job opportunities from external job boards.

User Feedback Mechanism: Each stage of the interaction allows for user confirmation or correction, which is crucial for maintaining the accuracy of user data. This mechanism also enhances user trust and engagement, as they can see and verify the information being processed.

4.6 Recommended Usage Path for Best Results

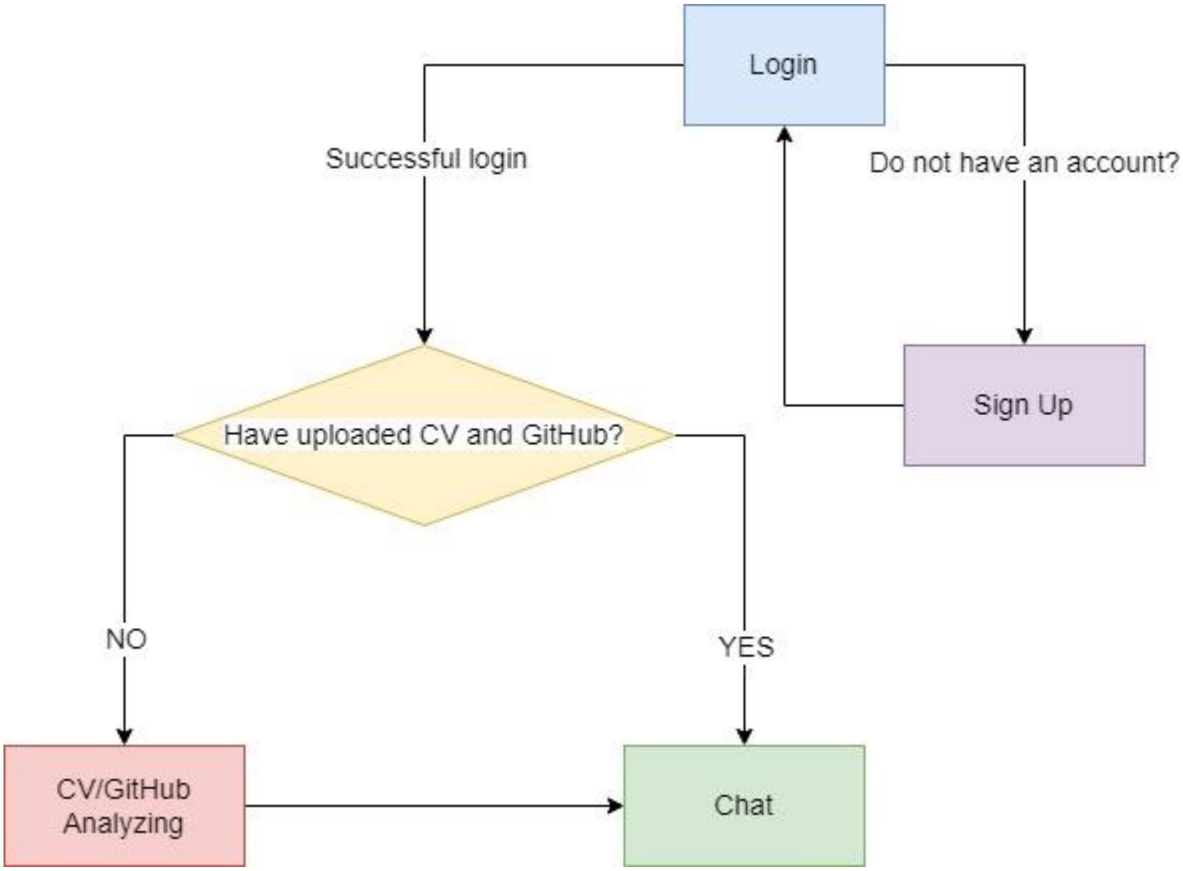


Figure 13

User Diagram

The diagram provided outlines the user flow for a job finder application, illustrating a critical path where users are prompted to upload their CV and GitHub data following a successful login or sign-up. This step is pivotal in personalizing the job search process, as it leverages the detailed information contained within a user's CV and GitHub profiles to match them with the most relevant job opportunities.

Uploading a CV allows the system to analyze comprehensive details about a user's educational background, work experience, skills, and accomplishments. This data provides a rich, structured insight into what the user has achieved and their professional capabilities. Similarly, integrating GitHub data enables the system to assess a user's coding projects, contributions to public repositories, and technical skills, which are often not fully captured in a traditional CV.

By combining these two data sources, the system can perform a nuanced analysis of both the user's formal qualifications and their practical, demonstrable skills. This dual-layered approach allows for a more accurate and personalized matching process compared to systems that rely solely on keyword matching or less dynamic user profiles. For instance, the system can identify not only the programming languages that the user knows but also their recent activity and expertise level in using those languages in real-world projects.

Consequently, this integration leads to highly personalized job recommendations that are closely aligned with the user's proven skills and professional preferences. This tailored approach significantly increases the likelihood of a successful job match, enhancing user satisfaction and improving the efficiency of the job search process, as depicted in the workflow diagram. This method not only streamlines the path to finding relevant job opportunities but also ensures that the recommendations are finely tuned to each user's unique professional profile.

The accuracy of the results produced by the chatbot heavily depends on the comprehensiveness and reliability of the information it receives from the users. If the chatbot relies solely on conversational data provided during interactions and does not have access to additional detailed documents like CVs, there may be gaps in the data it uses to match job listings.

CHAPTER 5 Evaluation

5.1 Survey

The primary goal of the survey is to gather feedback on the usability, effectiveness, and overall user experience of the chatbot. It aims to identify areas for improvement and assess the chatbot's performance in aiding users in their job search.

The survey followed the below structure:

- Demographic Information: Questions regarding gender, age range, and current employment status help contextualize the responses based on different user backgrounds.
- Job Search Behavior and Preferences: These questions explore the amount of time users dedicate to job searches and what factors they consider most important in a job (e.g., location, skills required, type of employment).

- Previous Experience with Job Finder Tools: This section investigates whether the respondents have used any job finder tools previously and which ones, to gauge the chatbot's uniqueness and utility compared to existing solutions.

- Willingness to Use AI-driven Job Finder: Questions here probe the respondents' openness to using an AI-powered tool like the chatbot, based on the details provided by the users themselves, their GitHub profiles, and CVs.

- Feedback on Chatbot Usage: Respondents are asked about any problems encountered while using the chatbot and any additional questions or features they would like to see, providing direct insights into user experience and satisfaction.

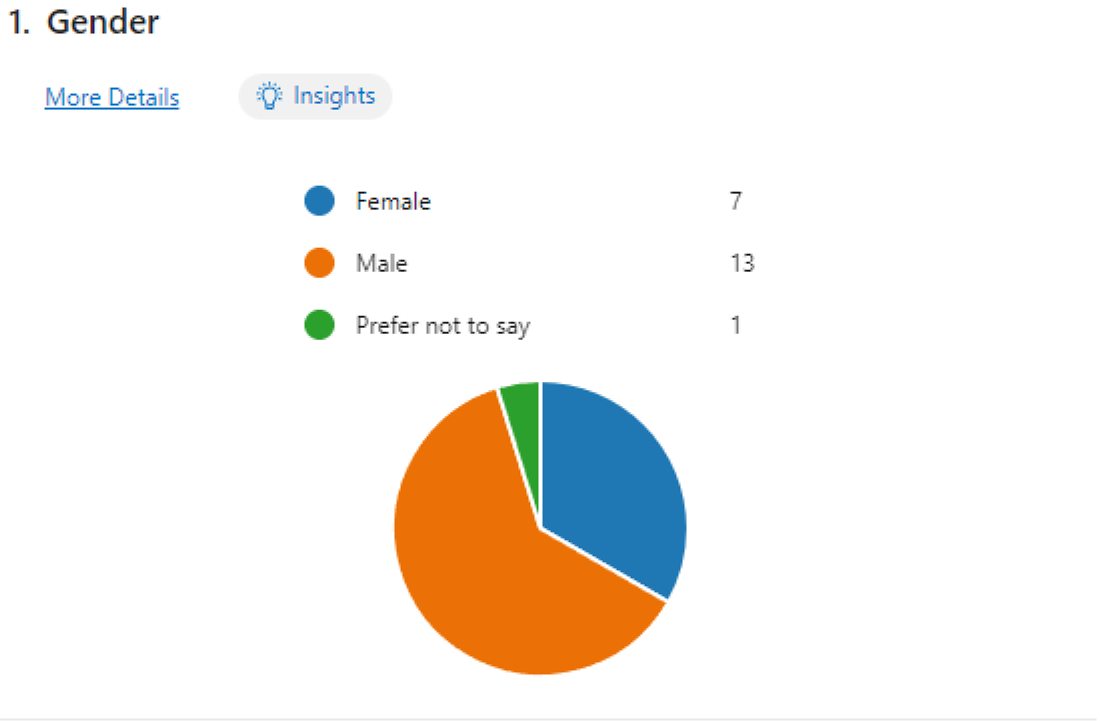
The survey emphasizes data anonymity, stating that personal data collected will not be used in any identifiable manner and will only be used in anonymized form for research purposes. Contact information for the project members is also provided for any queries or issues.

5.2 Results

Summary and Analysis of Survey Data:

Demographic Information:

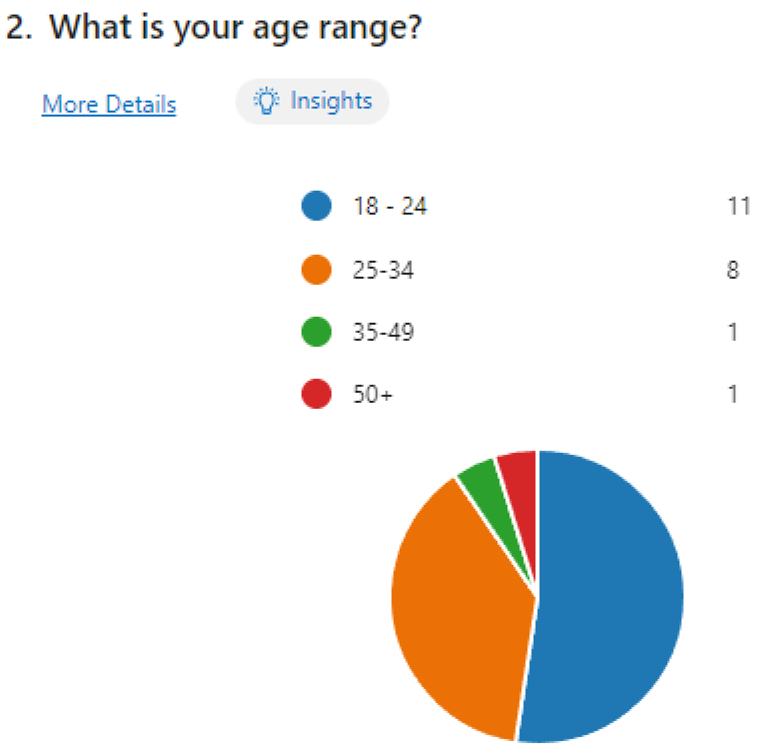
- Gender Distribution: The survey reveals a male-dominated respondent pool, with 13 out of 21 participants identifying as male. This demographic skew may influence perceptions and feedback related to the chatbot's interface and functionalities.



1. Gender

● Female	7
● Male	13
● Prefer not to say	1

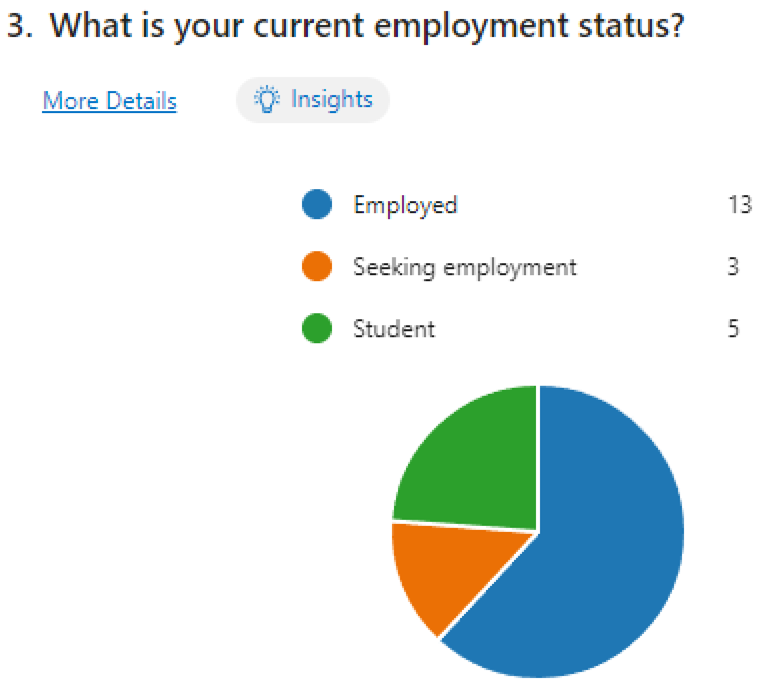
- Age Range: A significant majority of the respondents fall within the 18-24 age bracket (11 out of 21), suggesting that the chatbot's user base is predominantly young, likely reflective of early career job seekers.



2. What is your age range?

●	18 - 24	11
●	25-34	8
●	35-49	1
●	50+	1

- Employment Status: Over half of the respondents are employed (13 out of 21), which may indicate that the chatbot is also being used by those seeking better opportunities rather than just by job seekers out of employment.

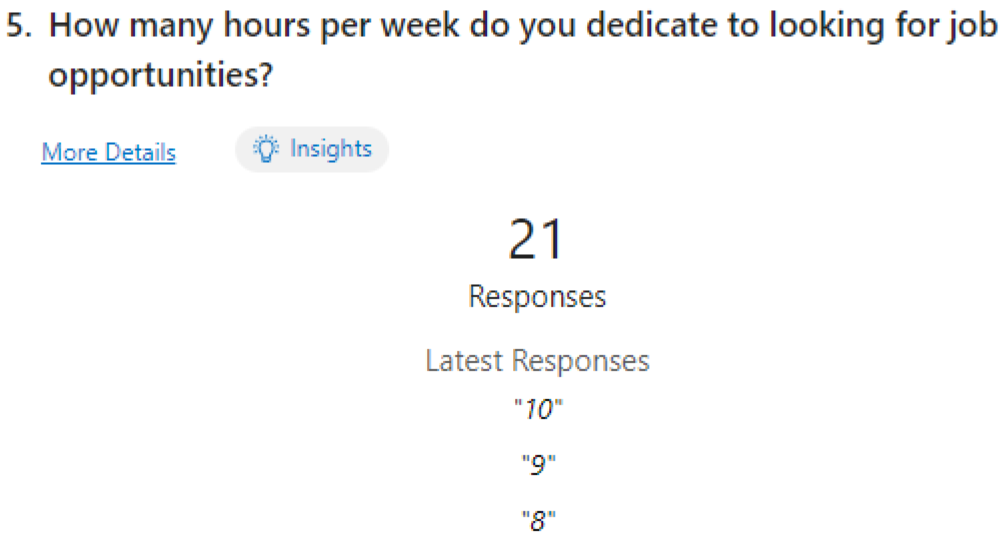


3. What is your current employment status?

●	Employed	13
●	Seeking employment	3
●	Student	5

Job Search Behavior:

- Hours per Week: Respondents spend an average of approximately 5.4 hours per week on job searches, with some dedicating as much as 10 hours weekly. The moderate variation in the time spent (standard deviation of 3.56 hours) highlights different levels of job search intensity among users, which could be a factor in their expectations and satisfaction with the chatbot.



5. How many hours per week do you dedicate to looking for job opportunities?

21

Responses

Latest Responses

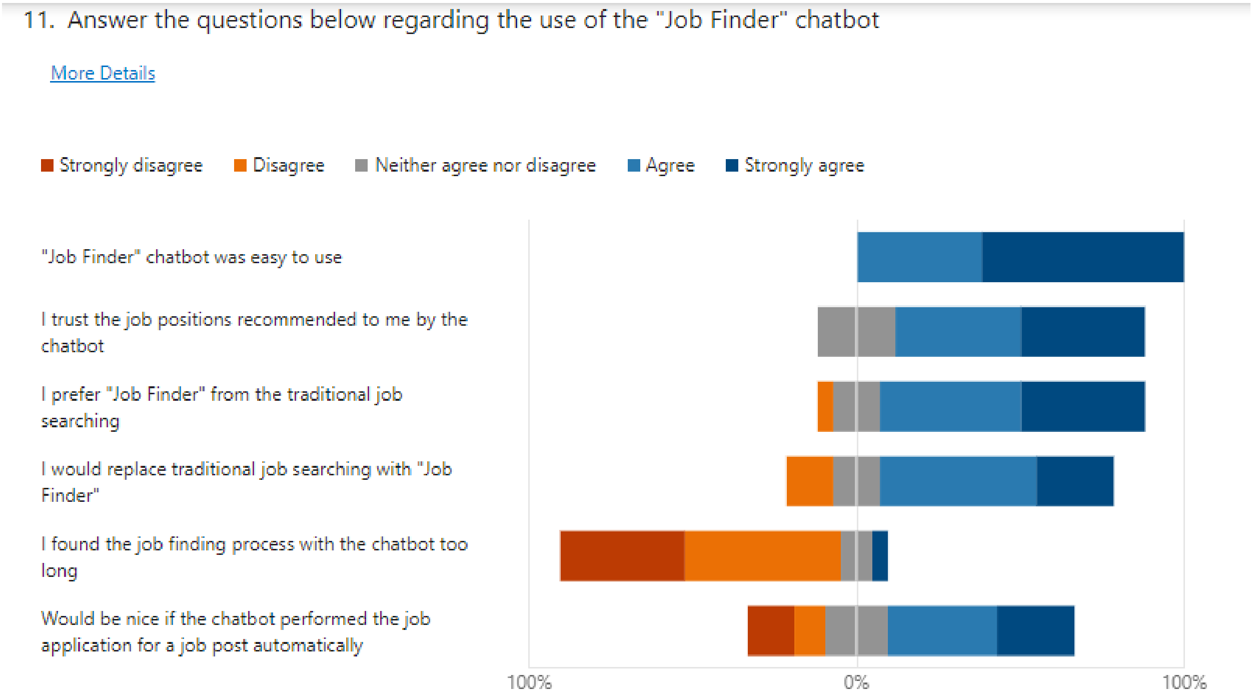
"10"

"9"

"8"

Feedback on the Chatbot:

- Ease of Use: A clear majority find the chatbot easy to use, with strong agreement (13 respondents) and agreement (8 respondents). This indicates that the chatbot's user interface and interaction design are effectively meeting user needs.
- Trust in Recommendations: The trust level in the chatbot’s job recommendations is high, mirroring the feedback on ease of use, which is crucial for user retention and continuous use.
- Preference Over Traditional Methods: A significant portion of the user base (15 respondents) shows a preference for the AI-driven job search method provided by the chatbot over traditional job searching techniques, indicating a shift towards technology-driven job search solutions.
- Job Finding Process: Opinions vary considerably regarding the length of the job finding process; a total of 18 respondents expressed dissatisfaction (disagree or strongly disagree), suggesting there is a considerable speeding up in the job searching process.
- Automatic Job Application Feature: The response to the automatic job application feature is mixed, with some users welcoming the convenience while others express reservations, likely due to concerns about control over the applications sent on their behalf.



Correlation Analysis:

- The weak correlation (-0.072624742) between the hours spent on job searching and satisfaction with the chatbot suggests that the user's engagement level (in terms of time spent searching) does not necessarily impact their satisfaction with the chatbot's functionalities.
- A notable negative correlation(-0.550047749) between the hours spent on job searches and satisfaction with the process length suggests that more intensive job seekers may have higher expectations regarding efficiency.

The "Job Finder Chatbot" is generally well-received, particularly in terms of usability, trustworthiness in job recommendations and the length of the job search process. However, the reception of the automatic job application feature indicates areas needing improvement.

Recommendations for Future Development:

- Customizable Automation Features: Introduction of customizable settings for automatic applications, allowing users to set preferences regarding which jobs the chatbot should apply for on their behalf, thus addressing concerns about automation and control.
- Continuous User Feedback Integration: Establish a regular feedback loop with users to continually adapt the chatbot to user needs and preferences, ensuring that it evolves in line with job market dynamics and user expectations.

- Demographic Expansion: Strategies to make the chatbot appealing to a broader demographic to ensure diverse feedback and usage patterns, enhancing the chatbot’s overall robustness and appeal to users unrelated to software engineering.

CHAPTER 6 Future and Conclusion

6.1 Summary

This thesis presented the development, implementation, and evaluation of the "Job Finder Chatbot," an AI-driven system designed to enhance the job search process for software engineers. This system matches individuals' resumes, GitHub activities and collects data from the chatbot with relevant job listings, leveraging advanced technologies and methodologies to optimize the job search experience.

Key Achievements:

- Technological Integration: The project successfully integrated several cutting-edge technologies, including Python, Selenium, BeautifulSoup4, MongoDB, Docker, Flask, JWT, RASA, and React. This diverse tech stack enabled the creation of a robust, scalable, and efficient chatbot that performs complex data handling and user interactions.
- Data Collection and Backend Development: Using Selenium and BeautifulSoup4 for scraping job listings and Lightcast for analyzing job market trends ensured a rich database of job opportunities. MongoDB's flexibility supported an efficient data management system that could handle the scale and complexity of the data involved.
- Frontend and User Interface: React was used to build a responsive and user-friendly interface, which played a crucial role in facilitating seamless interactions between the users and the chatbot.
- Chatbot Functionality: The implementation of RASA for natural language understanding and dialogue management allowed the chatbot to conduct meaningful and context-aware conversations with users, significantly enhancing user experience.

Evaluation Insights:

- User Feedback: The evaluation through user surveys highlighted the chatbot's ease of use and effectiveness in providing trustworthy job recommendations. Users appreciated the AI-driven approach for its modernity and efficiency compared to traditional job search methods.
- Areas for Improvement: Feedback pointed to the need for reducing the duration of the job search filtering process and introducing more customizable features, particularly in the automatic job application function, to cater to user preferences for control and specificity.

Recommendations:

- Enhanced Customization: Developing features that allow users to set detailed preferences for how the chatbot handles job applications on their behalf could address concerns about automation and increase the chatbot’s utility.
- Continuous Adaptation and Feedback: Implementing a mechanism for ongoing user feedback will be crucial for continuous improvement, ensuring that the chatbot remains effective and relevant as job market dynamics evolve.

6.2 Future Work

The "Job Finder Chatbot" has demonstrated significant potential in streamlining the job search process for software engineers through an AI-driven approach. To further enhance its effectiveness and broaden its applicability, the following future work is proposed:

1. Expansion to All Job Types

- Universal Job Matching: Adapt the chatbot's algorithms and databases to support a broader range of job categories beyond software engineering. This would involve integrating a wider variety of job listing sources and refining the chatbot’s natural language processing capabilities to understand and categorize a diverse array of job descriptions and qualifications across different industries.
- Sector-Specific Customization: Develop modular components within the chatbot that can be customized for different sectors, such as healthcare, finance, and education. Each module could feature tailored recommendations and search parameters relevant to the specific needs and characteristics of each sector.

2. User Interface Enhancements

- Intuitive Design Improvements: Redesign the user interface to be more intuitive and accessible, ensuring that users of all technical skill levels can navigate and utilize the chatbot with ease. This might include simplifying the layout, enhancing visual elements, and improving responsiveness across various devices and screen sizes.
- Interactive Elements: Implement interactive elements such as sliders, dropdown menus, and auto-complete text fields to make the data input process more engaging and less error-prone.

3. Creation of an Interactive Tutorial

- Guided Onboarding Process: Develop an interactive tutorial that guides new users through the chatbot's features and functionalities at the start of their first session. This tutorial should be easily accessible at any point for refresher guidance.
- Context-Sensitive Help: Offer context-sensitive tips and help icons throughout the chatbot interface. These tips can provide users with immediate explanations of certain fields or features, thereby enhancing their understanding and efficiency in using the chatbot.

4. Enhanced Security Measures

- Secure Logging Mechanism: Implement a secure logging system to track user interactions and system performance without compromising user privacy. This logging should be compliant with data protection regulations such as GDPR, ensuring that all data collected is done so with user consent and is securely stored.
- Regular Security Audits: Establish a routine for conducting security audits and vulnerability assessments to identify and address potential security threats. This will help in maintaining the integrity and trustworthiness of the chatbot.
- Advanced Authentication: Upgrade the authentication system to include multi-factor authentication (MFA) to provide an additional layer of security for accessing user accounts and sensitive information.

5. Continuous Learning and Adaptation

- Machine Learning Enhancements: Enhance the chatbot's machine learning models to continuously learn from user interactions and feedback. This adaptive learning approach can help in refining the accuracy of job matches and user responses over time.
- Feedback Loop Integration: Create a systematic feedback loop that allows users to provide direct feedback on job matches and overall experience. This feedback can be instrumental in continuously refining the chatbot’s algorithms and user interface.

6.3 Conclusions

The development and evaluation of the "Job Finder Chatbot" as detailed in this thesis represents a significant advancement in the use of artificial intelligence to facilitate the job search process. This project has demonstrated the profound capabilities of AI-driven technology to match software engineers with relevant job opportunities by analyzing their resumes and GitHub activities. The system's robust integration of cutting-edge technologies such as RASA for natural language processing, MongoDB for data management, and React for the user interface, has established a strong foundation for an efficient and user-centric job search tool.

The chatbot has proven to be highly effective in reducing the complexities associated with the traditional job search process. It has offered users a personalized experience by leveraging machine learning to accurately align job listings with the candidate’s profile. This personalization extends beyond mere qualification matching, considering the user’s preferred job locations, desired company culture, and other personal preferences, thereby enhancing user satisfaction and engagement.

The system has been rigorously evaluated through user surveys, which have provided insightful feedback on its performance. Users have appreciated the chatbot’s ease of use and the trustworthiness of its job recommendations. However, the feedback also highlighted areas for improvement, particularly concerning the duration of the job search process and the automation of job applications.

Looking ahead, the "Job Finder Chatbot" has ample room for expansion and enhancement. Key future developments include:

- Expanding the scope of the chatbot to cover various job sectors, making it a universal tool for job seekers across different industries.
- Enhancing the user interface to ensure it is more intuitive and accessible to a broader demographic, further simplifying the job search process.
- Incorporating an interactive tutorial to help new users navigate the chatbot’s functionalities effectively.
- Implementing robust security measures, such as secure logging and multi-factor authentication, to protect user data and increase the overall security of the system.

These enhancements will not only address the current limitations but also amplify the chatbot’s utility and reach, making it a more versatile and indispensable tool in the job market.

The "Job Finder Chatbot" represents a pivotal step towards revolutionizing the job search landscape. By aligning the latest advancements in AI with the specific needs of job seekers and recruiters, this project contributes significantly to the evolving field of AI in employment. This thesis not only showcases a practical application of AI but also paves the way for future innovations that could further transform how individuals and organizations approach job searching and recruitment. As we continue to explore and push the boundaries of AI capabilities, projects like the "Job Finder Chatbot" underscore the transformative potential of technology to create more efficient, personalized, and accessible job search experiences.

BIBLIOGRAPHY

1) Sridevi G.M et al. (2022), “AI based suitability measurement and prediction between job description and job seeker profiles” https://doi.org/10.1016/j.jjimei.2022.100109
2) V.R Uma et al. (2023), “Recruitment Analytics: Hiring in the Era of Artificial Intelligence” https://www.emerald.com/insight/content/doi/10.1108/978-1-80382-027-920231008/full/html
3) Connie R. Wanberg et al. (2020), “Job Seeking: The Process and Experience of Looking for a Job” https://doi.org/10.1146/annurev-orgpsych-012119-044939
4) Vivek Kaul MD et al. (2020), “History of artificial intelligence in medicine” https://doi.org/10.1016/j.gie.2020.06.040
5) https://simplify.jobs/
6) https://www.sonara.ai/
7) https://dream.jobs/
8) https://www.careerflow.ai/
9) Hamed Taherdoost et al. (2023), “AI Advancements: Comparison of Innovative Techniques” https://doi.org/10.3390/ai5010003
10) Tatwadarshi P. Nagarhalli et al. (2020), “A Review of Current Trends in the Development of Chatbot Systems” https://doi.org/10.1109/ICACCS48705.2020.9074420
11) Peter Gordon Roetzel (2019), “Information overload in the information age: a review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development” https://doi.org/10.1007/s40685-018-0069-z
12) Qing Zhou et al. (2019), “Job recommendation algorithm for graduates based on personalized preference.” https://doi.org/10.1007/s42486-019-00022-1
13) https://www.selenium.dev/
14) https://pypi.org/project/beautifulsoup4/
15) https://lightcast.io/
16) https://www.mongodb.com/
17) https://www.docker.com/
18) https://flask.palletsprojects.com/en/latest/api/
19) https://jwt.io/
20) https://rasa.com/
21) https://react.dev/
22) Chowdhary, K.R. (2020). Natural Language Processing. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_19
23) Shneiderman, B., & Plaisant, C. (2010). *Designing the user interface: Strategies for effective human-computer interaction*. Pearson Education India.
24) https://vercel.com/
25) **Derek S. Chapman and Jane Webster** (2003). "The use of technologies in the recruiting, screening, and selection processes for job candidates,"
26) https://ngrok.com/

APPENDIX

Job Docker

```
version: '3.8'

services:
  mongodb:
    image: mongo:latest
    container_name: mongodb
    restart: always
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: password
      MONGO_INITDB_DATABASE: jobs

volumes:
  mongo-data:
```

User Docker

```
version: '3.8'

services:
  mongodb:
    image: mongo:latest
    container_name: mongodbUsers
    restart: always
    ports:
      - "27018:27017"
    volumes:
      - mongo-data-users:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: password
      MONGO_INITDB_DATABASE: users

volumes:
  mongo-data-users:
```

Rasa Action Endpoint

```
action_endpoint:
 url: "http://localhost:5055/webhook"
```

Rasa Config

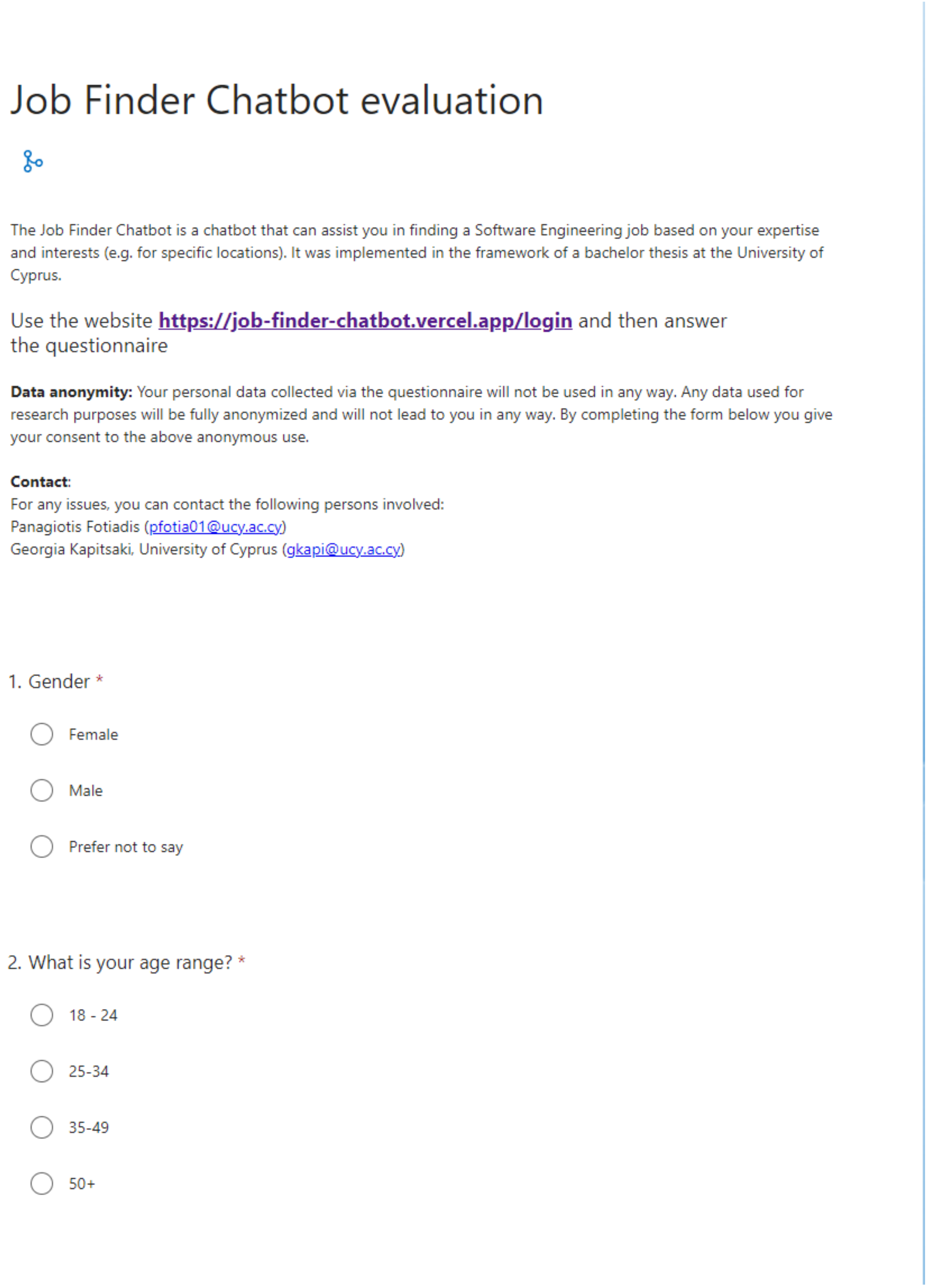
```
recipe: default.v1

assistant_id: 20240109-002533-amber-axel

language: en

pipeline:
  - name: WhitespaceTokenizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 4
  - name: DIETClassifier
    epochs: 200
    entity_recognition: true
    intent_classification: true
    entity_classification: true
  - name: EntitySynonymMapper
  - name: RegexEntityExtractor
    case_sensitive: false
    use_lookup_tables: true
  - name: ResponseSelector
    epochs: 100

policies:
  - name: MemoizationPolicy
  - name: TEDPolicy
    max_history: 15
    epochs: 200
```

API '/get_results'

```
def jaccard_similarity(set1, set2):
    intersection = len(set1 & set2)
    union = len(set1 | set2)
    return intersection / union

@app.route('/get-results', methods=['POST'])
def get_results():
    data = request.json
    collection, client = initialize_connection_jobs()

    # Specify the desired "Location" value to match
    query_builder = {}
    # Check if the "Location" field in data is not empty
    if data.get("Location"):
        query_builder["Location"] = {"$in": data["Location"]}

    # Check if the "Company" field in data is not empty
    if data.get("Company") != "None":
        query_builder["Company"] = data["Company"]

    # Check if the "Employment Type" field in data is not empty
    if data.get("Employment Type"):
        query_builder["$or"] = [
            {"Employment Type": {"$in": data["Employment Type"]}},
            {"Employment Type": ""}
        ]

    # Check if the "Years of Exp" field in data is not empty
    if data.get("Years of Exp"):
        query_builder["$or"] = [
            {"Years of Exp": {"$lte": data["Years of Exp"]}},
            {"Years of Exp": "not given"}
        ]

    # Check if the "Education Level" field in data is not empty
    if data.get("Education Level"):
        query_builder["$or"] = [
            {"Education Level": {"$in": data["Education Level"]}},
            {"Education Level": ""}
        ]

    # Check if the "Education Type" field in data is not empty
    #if data.get("Education Type"):
```

```
    #    query_builder["Hard Skills"] = {"$in": data["Education Type"]}

    matching_entries = collection.find(query_builder)


    similar_documents = []

    for entry in matching_entries:
        document_hard_skills = set(entry.get("Hard Skills", []))  # Extract the
"Hard Skills" field from the document
        document_soft_skills = set(entry.get("Soft Skills", []))  # Extract the
"Soft Skills" field from the document
        hard_skill_similarity = jaccard_similarity(set(data["Hard Skills"]),
document_hard_skills)
        soft_skill_similarity = jaccard_similarity(set(data["Soft Skills"]),
document_soft_skills)
        if hard_skill_similarity >= 0.2 and soft_skill_similarity >= 0.2:  #
Check if the similarity is at least 50%
            similar_documents.append(entry)
        elif hard_skill_similarity >= 0.2 and document_soft_skills == set([]):
            similar_documents.append(entry)
        elif soft_skill_similarity >= 0.2 and document_hard_skills == set([]):
            similar_documents.append(entry)
        elif document_soft_skills == set([]) and document_hard_skills == set([]):
            similar_documents.append(entry)


    matching_entries_list = list(similar_documents)
    urls = [entry['URL'] for entry in matching_entries_list]
    json_result = json.dumps(urls, default=str, indent=4)
    client.close()

    return json_result
```

Survey



Job Finder Chatbot evaluation

The Job Finder Chatbot is a chatbot that can assist you in finding a Software Engineering job based on your expertise and interests (e.g. for specific locations). It was implemented in the framework of a bachelor thesis at the University of Cyprus.

Use the website **https://job-finder-chatbot.vercel.app/login** and then answer the questionnaire

Data anonymity: Your personal data collected via the questionnaire will not be used in any way. Any data used for research purposes will be fully anonymized and will not lead to you in any way. By completing the form below you give your consent to the above anonymous use.

Contact:
For any issues, you can contact the following persons involved:
Panagiotis Fotiadis (pfotia01@ucy.ac.cy)
Georgia Kapitsaki, University of Cyprus (gkapi@ucy.ac.cy)

1. Gender *

○ Female

○ Male

○ Prefer not to say

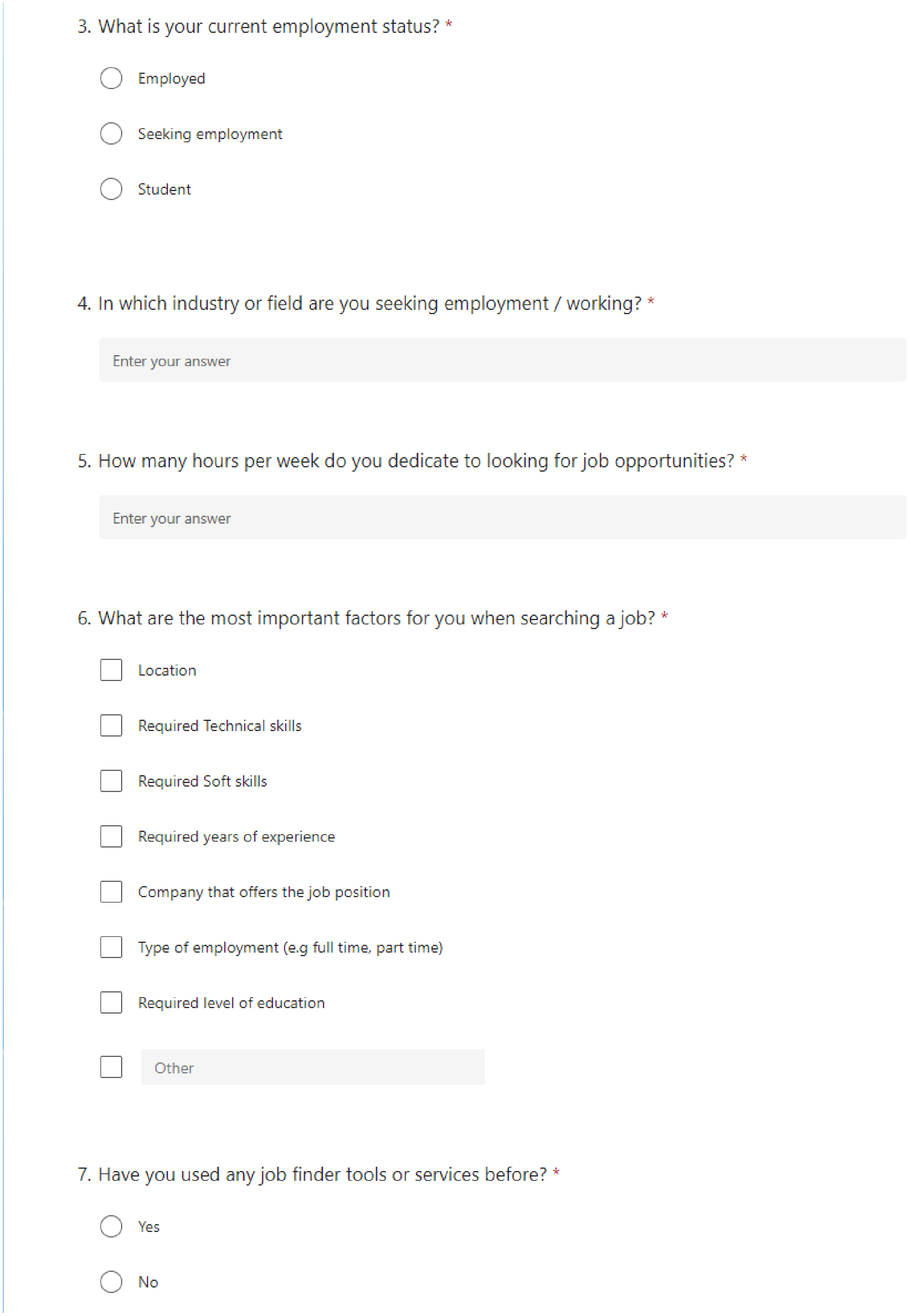
2. What is your age range? *

○ 18 - 24

○ 25-34

○ 35-49

○ 50+



3. What is your current employment status? *

○ Employed

○ Seeking employment

○ Student

4. In which industry or field are you seeking employment / working? *

Enter your answer

5. How many hours per week do you dedicate to looking for job opportunities? *

Enter your answer

6. What are the most important factors for you when searching a job? *

☐ Location

☐ Required Technical skills

☐ Required Soft skills

☐ Required years of experience

☐ Company that offers the job position

☐ Type of employment (e.g full time, part time)

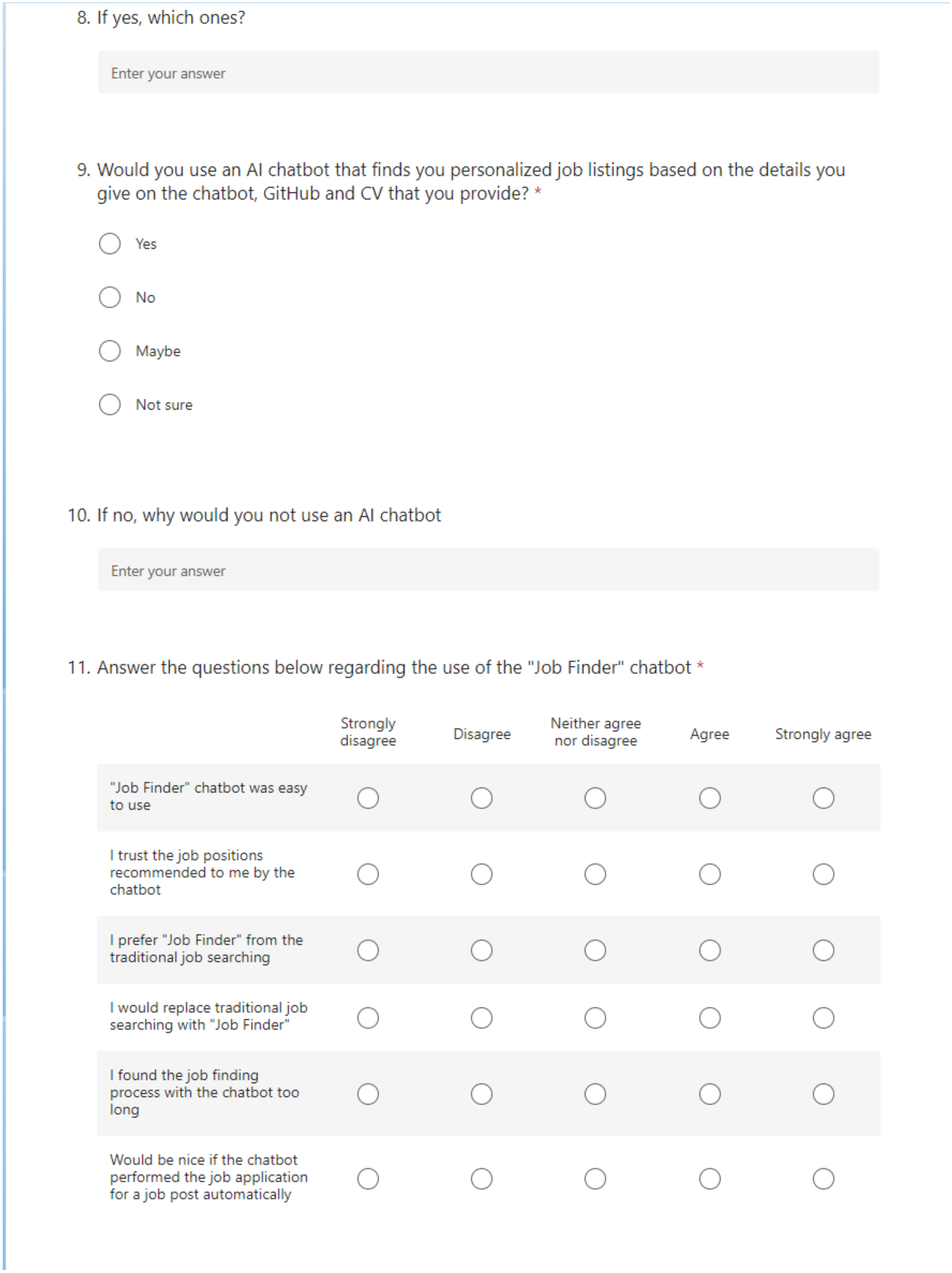
☐ Required level of education

☐ Other

7. Have you used any job finder tools or services before? *

○ Yes

○ No



8. If yes, which ones?

Enter your answer

9. Would you use an AI chatbot that finds you personalized job listings based on the details you give on the chatbot, GitHub and CV that you provide? *

○ Yes

○ No

○ Maybe

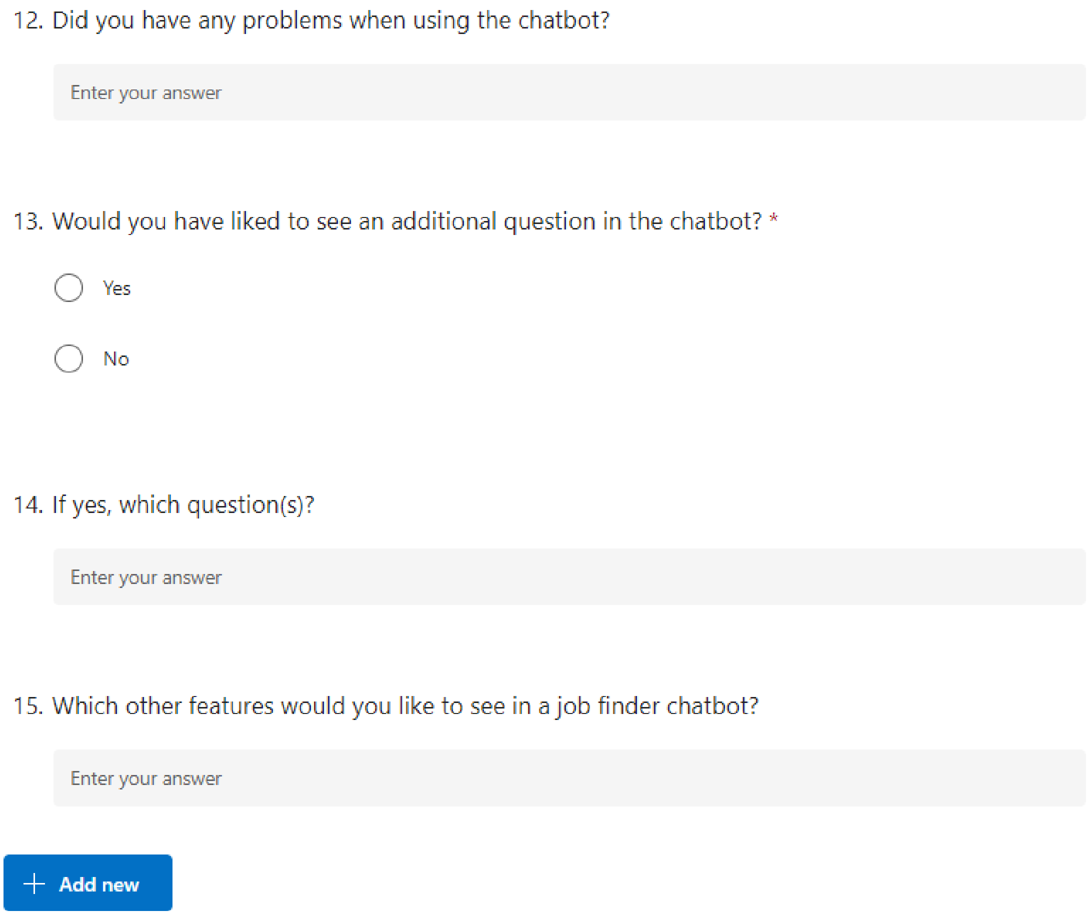
○ Not sure

10. If no, why would you not use an AI chatbot

Enter your answer

11. Answer the questions below regarding the use of the "Job Finder" chatbot *

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
"Job Finder" chatbot was easy to use	○	○	○	○	○
I trust the job positions recommended to me by the chatbot	○	○	○	○	○
I prefer "Job Finder" from the traditional job searching	○	○	○	○	○
I would replace traditional job searching with "Job Finder"	○	○	○	○	○
I found the job finding process with the chatbot too long	○	○	○	○	○
Would be nice if the chatbot performed the job application for a job post automatically	○	○	○	○	○



12. Did you have any problems when using the chatbot?

Enter your answer

13. Would you have liked to see an additional question in the chatbot? *

○ Yes

○ No

14. If yes, which question(s)?

Enter your answer

15. Which other features would you like to see in a job finder chatbot?

Enter your answer