WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

JobEase Bot - AI-Powered Job Application Automation

Y. Venkata Jyoshna Reddy[1], S.Bhaavana[2], T. saibhagath[3], M. Janakiram[4]

[1] GMR Institute of Technology jyoshnayeraguti2831@gmail.com 7981832417
[2] GMR Institute of Technology srinadhbhaavana@gmail.com 850095696
[3] GMR Institute of Technology saibhagath1910@gmail.com 9392704330
[4] GMR Institute of Technology janakirammarrapu@gmail.com 9398034574

ABSTRACT—

The growing dependence on automation in employment recruitment has resulted in the creation of AI-based job application systems that improve efficiency and precision. This project introduces JobEase Bot, a web application built with Flask to automate LinkedIn job applications through **Selenium WebDriver**. The system enables users to enter job preferences, upload their resumes, and apply automatically to suitable job postings, greatly minimizing manual labor in the job search process.

To enhance candidate-job matching, Generative AI (Gemini AI) is used to dynamically enrich resumes and create customized cover letters that match individual job postings. Resume parsing is performed with pdfplumber and regex, pulling out important information like name, email, phone number, and LinkedIn profile from uploaded resumes. This allows for customized and optimized job applications based on user credentials.

The system uses SQLite as its backend database to store user credentials (hashed for security), job application history, extracted resume information, and job search preferences. A dynamic filtering mechanism** allows job searches based on pre-defined parameters, including job role, location (on-site, remote, hybrid), and experience level, to ensure targeted applications that meet user-defined parameters.

An intuitive web interface, constructed using Flask and HTML/CSS, provides effortless interaction through which users are able to upload applications, view past applied jobs, and monitor saved data. The application further tracks all applied job applications for reference purposes in the future, with insights offered into application patterns and improved job suggestions over time.

Using AI-based automation, intelligent resume matching, and live LinkedIn job application capabilities, this project makes job hunting more efficient, eliminates lengthy manual processes, and enhances job placement success. Additional future upgrades might involve machine learning-based job recommendation engines, job application support on multiple platforms, and integration with third-party APIs to further enhance capabilities.

Keywords— AI-driven Job Application, LinkedIn Automation, Resume Parsing, Generative AI, Selenium WebDriver.

I. Introduction

In the fast-paced digital age, the job application process has become a challenge and an opportunity for job seekers. With the rise of online job boards and professional networking sites such as LinkedIn, individuals have never had greater access to job opportunities in industries and geography. But with ease comes a major drawback: the time and effort spent searching for, screening, and applying for suitable positions manually can be daunting. For working professionals with multiple commitments, it is balancing full-time work, caregiving responsibilities, or studying—or those looking to change careers rapidly, this time-consuming and laborious process results in burnout or lost opportunities. The sheer number of postings, combined with the need to customize applications to be noticed, creates a bottleneck that many applicants find difficult to overcome. Noticing this gap, the "JobEase Bot - Auto Apply" initiative was created to automate and simplify the job application process, using advanced technologies to equip users with efficiency, accuracy, and a competitive advantage in the job market.

JobEase Bot is a web-based application built to make job searching easier through automation of job application on one of the world's most advanced professional networking platforms, LinkedIn. Constructed through a blend of front-end web development, back-end automation, and artificial intelligence (AI), the project's purpose is to minimize the toil of job searching for candidates while maximizing the quality of applications. With a blend of simplified interfaces and potent automation capabilities, the bot provides users with interfaces to enter credentials, upload their resumes, outline job preferences, and trigger a job search and application process automatically with minimal input from humans. Not only does the system scour jobs based on the user's specifications, but it also personalizes application content, including cover letters and resumes, to complement individual job advertisements, thus increasing the chances of success. With this automation, repetitive inputting of data and tiresome filling of forms becomes a thing of the past as users can

use more time and effort on other vital aspects of their career progress, such as skill acquisition or networking.

In its very nature, the JobEase Bot demonstrates the potential of automation and AI in transforming mundane tasks. The project uses Selenium WebDriver to automate the browser, enabling it to interact with the LinkedIn interface in a way that resembles that of a human user—logging in, searching for jobs, and applying for jobs with ease. It also uses Google's Gemini AI model to scrape appropriate information from resumes and generate customized cover letters in sync with job descriptions. This combination of web scraping, natural language processing (NLP), and database management provides a strong framework for navigating the complexities involved in modern job applications. Whether the user is a fresh graduate seeking an entry-level job, an experienced professional seeking a specialized job, or even a career changer seeking to venture into new fields, the bot is flexible enough to suit multiple requirements, thus serving as a multi-purpose tool in the job market. Its scalability across different user categories and job types further demonstrates its potential as a viable solution to a global problem.

The user interface of the JobEase Bot is programmed in HTML, CSS, and JavaScript to facilitate a clean appearance and user-friendliness in design. Executing on a web application built on Flask, the interface allows the user to input their LinkedIn login credentials, choose job types from a pre-defined set (for instance, Software Engineer, Data Scientist, or DevOps Engineer), specify the number of their years of experience, and choose their preferred work locations (On-Site, Remote, or Hybrid). At the center of the front-end is the feature of uploading resumes, whose PDF format automatically extracts the personal details of name, email, and phone number. That information is subsequently used to complete application forms in advance, thus avoiding redundancy while improving accuracy. The UI also prioritizes user experience through the feature of toggling passwords, dynamic display of job information, and instant feedback upon applying, all designed using modern-day CSS frameworks and animations to be visually clean. All this thoughtfulness makes itself available to assist users with even limited technical capability to use the system with ease.

On the server side, Python with Flask manages the flow of automation. The center of this system is the JobEaseBot class, which carries out the necessary logic to interact with LinkedIn, from user login with the given credentials to job listing retrieval and applying opportunities via the "Easy Apply" function. SQLite databases ensure that application-related information such as job title, experience level, and location preference is stored in a persistent state so that users can monitor their application history and see previous submissions. Concurrently, the utilization of pdfplumber for the parsing of resumes and the utilization of regular expressions (regex) for the extraction of information ensures that the bot can comprehensively interpret unstructured information, even if resumes are in varied formats or styles. Throughout the code, error handling and logging systems are utilized to provide reliability as well as to provide transparency regarding the actions of the bot to enable the debugging of errors or functionality improvement. This end-to-end back-end system enables the seamless execution of complicated tasks and hence ensures the reliability of the bot under real-world scenarios.

The idea behind the JobEase Bot lies in an aspiration to bring job opportunities within easier reach of a larger population base. Conventional job application processes usually work to the advantage of individuals who possess considerable time and means at hand, thus leaving behind busy professionals, caregivers, or inhabitants of deprived localities. For instance, a sole, full-time working parent may struggle to take enough hours to job hunt, while a person who lives in a rural town with a slow internet connection might lose job prospects due to pragmatic reasons. By eliminating repetitive processes and offering AI-driven personalization, the bot brings about a leveler in the playing field, allowing users to apply to several jobs in the time needed to fill in a single manual application. Not only does the saved time lighten the load but also the cognitive fatigue that normally forms part of job hunting, thus making it simpler for the user to stay engaged and positive throughout the entire process. Also, its modularity allows future enhancement, like the support for other job websites (like Indeed and Glassdoor), advanced resume optimization algorithms, or the addition of cloud-based storage solutions to the management of multiple resumes. Up to March 22, 2025, the project has yielded a working prototype, ready to ease users through the competitive job landscape.

JobEase Bot's front-end is implemented on the web development trifecta of HTML, CSS, and JavaScript. HTML establishes the framework skeleton, determining the structure of the user interface through which job applicants enter their LinkedIn account credentials, job choices, and resume. CSS, augmented with the Font Awesome library for icons, dresses this interface in a contemporary look, with a gradient background, responsive layout, and smooth animations through keyframes (e.g., the fadeIn effect). JavaScript provides interactivity, responding to events like form submission, password visibility toggle with an eye icon, and dynamically showing job descriptions based on user choices. This client-side integration provides an intuitive and visually rich experience, essential to making the bot user-friendly for users with varying levels of technical sophistication. Running in a Flask web application, the front-end is the entry point, providing seamless access from user input to the back-end automation engine.

Apart from its technical success, JobEase Bot represents a larger vision of how automation and AI can support human productivity. It solves actual pain points in the real world—time shortage, application overload, and personalization requirements—while upholding ethical standards, like compliance with platform terms of service and data security for users. The project also raises the topic of how technology's role in work continues to change, a debate about whether automation will augment human labor or substitute for it. The bot, for instance, takes care of the mechanics of job application but leaves the definition of objectives and resumé uploading in the hands of the user, maintaining the human touch in the employment quest. This balance between agency and automation guarantees that JobEase Bot is an empowerment tool and not an entirely independent decision-maker.

The creation of JobEase Bot also emphasizes the value of flexibility in technology design. Job markets are fluid, with changes depending on economic circumstances, advances in technology, and demands of the workforce. By creating a system that can parse varied job postings, tailor applications, and retain historical information, the project establishes a foundation for longevity and scalability. Subsequent versions could include machine learning to forecast job fit based on previous application success rates or include real-time labor market analytics to steer users toward high-demand positions. These improvements would further cement the position of the bot as an active career guide, able to adapt alongside its users' requirements and the wider job

market.

Behind the scenes, Flask, a lightweight Python web framework, manages the server-side logic and serves as the glue between the front-end interface and the automation workflows. Flask processes HTTP requests, including form submissions through the /start-bot endpoint, and processes file uploads (e.g., resumes) located in the upload's directory. Its ease of use and flexibility make it well-suited for rapid development and integration with other Python-based tools. The routing feature of the framework allows the bot to store application information, retrieve previous submissions, and provide immediate feedback to users via JSON responses. With the help of Flask's development server and CORS support, the system can communicate between the client and server with ease, paving the way for the more sophisticated automation tasks handled by the JobEaseBot class.

Lurking behind the automation is Selenium WebDriver, an effective browser automation tool that makes it possible for JobEase Bot to communicate with LinkedIn's interactive web page. With the aid of the ChromeDriver (managed through webdriver_manager), Selenium mimics human interactions—logging in using user credentials, accessing job search pages, scraping job posts, and hitting the "Easy Apply" button. Its ability to wait for elements (through WebDriverWait and Expected Conditions) makes it reliable despite LinkedIn's asynchronous loading. Set up with Chrome options such as --no-sandbox and a custom user-agent, Selenium hides its automated character, lowering the detection risk while maximizing compatibility with LinkedIn's dynamically changing UI. This technology plays a crucial role in translating user inputs into actionable steps, bridging the divide between static code and real-time web interactions.

The AI drives the bot's function of processing and personalizing job applications, Google's Gemini AI model being used in the central component. Made available through the google. generativeai Python library, Gemini pulls specific information from job descriptions scraped with Selenium and forms individualized cover letters based on this information from resumes. Employing natural language processing (NLP), it translates unstructured resume content into professional-sounding responses to job-specific questions, like "Why do you want this job? " This customization powered by AI improves application quality, marrying the user's skills to employer needs.

Gemini was selected for its cutting-edge generative abilities, providing a scalable solution to analyzing text and creating content that varies according to different job functions and needs. Complementing Gemini, pdfplumber is used for resume parsing, pulling raw text from uploaded PDFs. This Python library is particularly adept at dealing with the diverse layouts of resumes, translating them into a format appropriate for regex-based detail extraction (e.g., email, phone, LinkedIn URLs). Regular expressions (regex) then further clean this process, using pattern matching to extract structured data from free-form text. Coupled together, pdfplumber and regex guarantee vital information is being captured and saved accurately, flowing into both AI customization and form-filling streams. This tandem offers a lean yet efficient replacement for more heavy-duty OCR-based solutions, keeping performance in sync with accuracy.

Persistence of data is done using SQLite, a serverless relational database integrated into the application. SQLite persists two main tables: applications (job roles, experience, location preferences) and resume_details (holding extracted resume information). Its ease of use and file-based approach suit a prototype like JobEase Bot, with no need for an independent database server but with support for concurrent access through Python's sqlite3 module. This database allows users to inspect previous applications and keeps a tab of pulled-out information, which makes the utility of the bot as a management tool for work even more worthwhile. The deployment of SQL queries in Flask routes provides smooth fetching and storing of data, thereby adding an aspect of continuity to the user's experience.

Helping libraries such as logging, os, and time make the system stronger. The logging module offers extensive debug output, written to timestamped log files and printed to the console, to facilitate troubleshooting and openness. The os module takes care of file operations (such as making directories, removing temporary cover letters), and time adds delays to avoid rate-limiting problems while automating. All these utilities together make sure the bot works flawlessly, coping with errors gracefully and leaving the environment in order. The Flask-CORS extension also supports cross-origin requests, allowing for possible future extensions to decouple front-end and back-end deployments.

The interplay of these technologies—HTML/CSS/JavaScript for the interface, Flask for server-side orchestration, Selenium for automation, Gemini AI for smarts, pdfplumber and regex for parsing, SQLite for storage, and supporting libraries for reliability—provides a seamless system to overcome the challenges of job applications. Up to March 22, 2025, this tech stack makes JobEase Bot a vision-oriented prototype, with the potential to grow with the advancement of AI, web automation, and database technologies. This introduction sets the stage for a deeper dive into each component's implementation, showcasing how they collectively transform a manual, time-intensive process into an automated, user-centric solution.

II.Ease of Use

A. User-Friendly Web Interface

The bot has an easy-to-use and intuitive Flask-based web interface, where users can simply input job preferences, upload their resume, and initiate the job application process without the need for any technical knowledge.

B. Automated Job Applications

No more tedious manual searching and application for jobs. The bot automatically logs into LinkedIn, finds relevant jobs, and applies, conserving time and effort.

C. AI-Driven Resume and Cover Letter Creation

Using Generative AI (Gemini AI), the bot personalizes resumes and creates custom cover letters for every job application, removing the hassle of manually editing files.

D. Effective Data Management

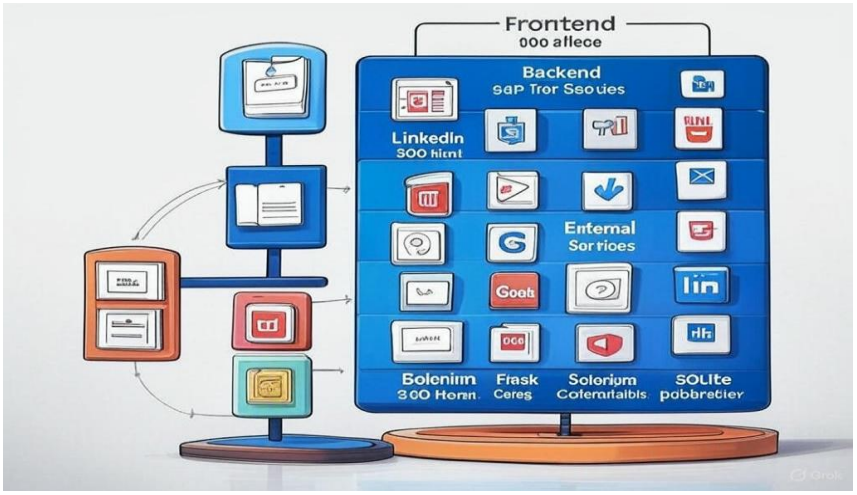
The program securely saves user credentials (hashed), application history, and resumes information extracted in an SQLite database, enabling users to monitor applications and enhance future job searches.

E. Flexible Job Search Criteria

Applicants may search for, and apply for, jobs via tailored filters such as job description, location (office-based, home-based, hybrid), and experience level so that highly qualified applications are delivered to suit.

METHODOLOGY

System Architecture



JobEase Bot is an online tool made to automate LinkedIn job searches and applications. Three primary layers make up the architecture:

Frontend (Client Layer):

- Using JavaScript (index.html), HTML, and CSS.
  A user interface featuring a form for uploading a résumé (in PDF format), entering LinkedIn credentials, and selecting a job (position, experience, and location type).
- Features include dynamic components such as the display of a job description, the ability to alter the visibility of a password, and real-time feedback (such as loading messages and results).

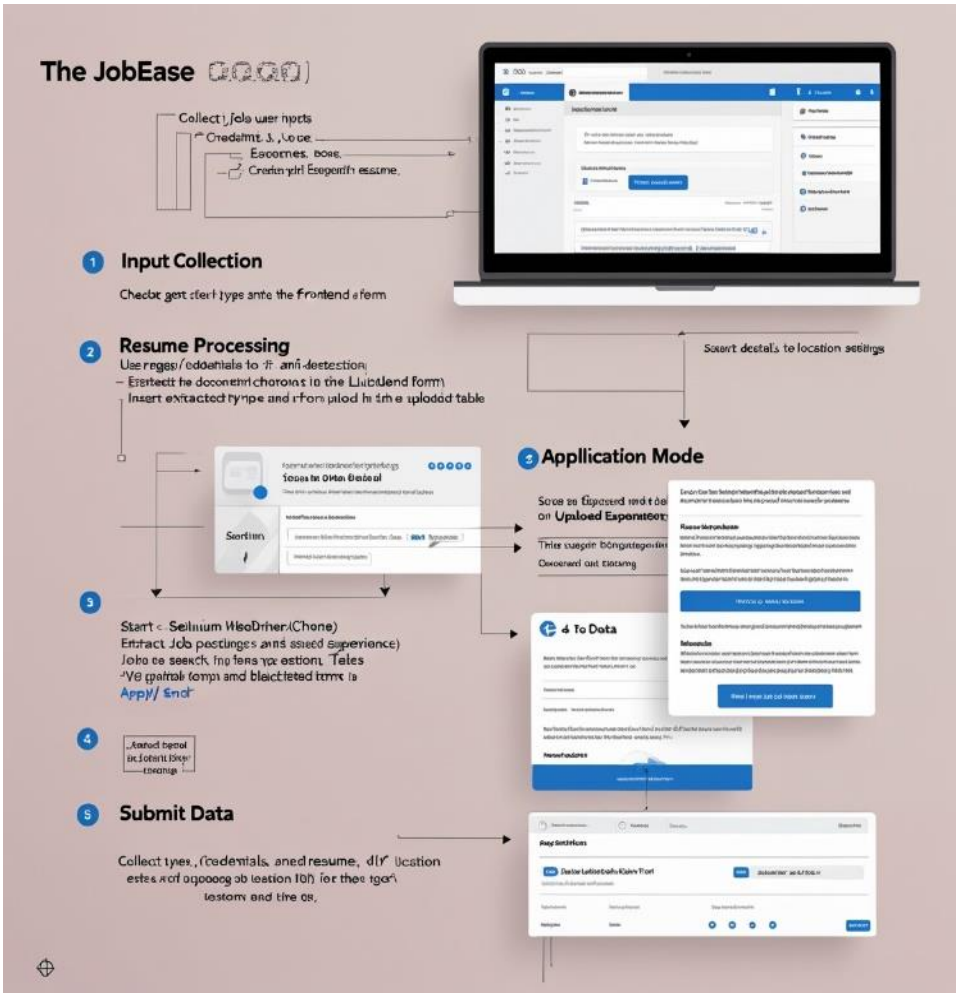
Backend (Server Layer):

- Built using Flask (app. py), A Python web framework (Lua), Running on localhost:5050.
- Process all API Endpoints (/start-bot, /save-application, /get-applications, /saved-applications) Used to process user requests, database interactions, and manage the bot.
- Supports library complexity:
- Selenium for browser automation (Navigating and interacting with LinkedIn).
- pdfplumber for extracting the data from the resume.
- google. generativeai (Gemini API) to craft cover letters and tailor resumes.
- sqlite3 to store the application data and also details of what resume is to be printed.
- Installed flask_cors for integrating cross-origin between frontend and backend

External Services:

- LinkedIn: Selenium WebDriver for the job search and apply application.
- Google Gemini API: Secures AI-powered content creation (cover letters, CV customization).
- SQLite Databases (jobsease. db, applications. db): Save application user-related information and resume details extracted.

Algorithm Model



The JobEase Bot uses a procedural or rule-based algorithm, which roughly contains the following steps:

1. **Input Collection:**
   - Collect user inputs (Credentails, job role, experience, location, resume) through the frontend form.
   - Check gets (e.g., document type, encounter esteem).
2. **Resume Processing**:

   - Use regex and pdfplumber and extract the details (email, phone, LinkedIn URL, text) from the uploaded PDF.
   - Insert extracted information in the resume_details table.

3. **LinkedIn Interaction:**

   - Start a Selenium WebDriver (Chrome) with anti-detection settings.
   - Enter the credentials attached for logging into LinkedIn.
   - Job search based on user-specific parameters (role, location & experience).
   - Scrape job postings and weed out blacklisted terms (like “unpaid”).

4. **Application Automation:**

   - For each job:
   - Navigate to the job URL.
   - Extract the job description.
   - You are configured with data ending at October 2023.
   - This is the form you fill in the resumes details and upload the documents under the "Easy Apply" section.
   - Take care of any custom questions and submit the app.

5. **Data Storage and Response:**

- Store application details (username, job role, etc.) into the applications table.
- Return job listings and resume details to the frontend for display.

Step-by-Step Explanation

1. **User Input Submission**:

- The user completes the form in published. html (e.g., your username, password, job title i.e., Software Engineer, 5 years of experience, resume PDF).
- JavaScript checks input data (resume size should be less than 2MB) and submits the form using a FormData via a POST request to /start-bot.

2. **Backend Initialization**:

- Flask handles the request, saves the resume in the upload's directory, and performs input validation (e.g., ensures credentials are not empty, experience is in a valid format etc.).
- After the preprocessing is completed, the script instantiates the JobEaseBot class with the user data (username, job_role, resume_path, etc.) and makes a call to the get_job_suggestions (in the background, the messaging app (tkinter) will display the user information).

3. **Resume Extraction**:

- extract_resume_details uses pdfplumber to open the PDF and extract text,
- Regex patterns detect any email (ex: user@example.com), phone ex(+1-123-456-7890) and LinkedIn URL (ex: linkedin. com/in/user).
- store_resume_details saves details to the resume_details table

4. **LinkedIn Login**:

- setup_browser — A Chrome WebDriver with human-user-hacking options configured (set user-agent, etc.).
- login_to_linkedin goes to linkedin. com/login, enters credentials, clicks the login button.
- Waits a maximum of 180 s for URL to contain "feed", indicating successful log in.

5. **Job Search:**

- search_jobs will build a URL like that
- (https://www.linkedin.com/jobs/search/?keywords=Software%20Engineer&location=worldwide&f_E=5).
- Selenium opens the page, scrolls for more listings, and scrapes job cards (title, company, location) based on CSS selectors.
- Returns list of jobs (e.g., [{"title": "Software Engineer", "company": "TechCorp", "location": "Remote"}])

6. **Application Process (Prototype)**:

- For each job (at present, only get search results to return in this code)
- apply_to_job goes to the job URL, clicks on the Easy Apply button and gets the job text.
- generate_cover_letter uses the Gemini API to generate a cover letter from the text of the resume and job description.
- customize_resume does the same for the resume.
- fill_form fills out the right fields (email, phone, resume) and submits the application.

7. **Data Storage and Response**:

- /save-application saves application details to applications table.
- The backend responds with a json {"jobs": [...], "resume_details": {"email": "user@example.com",...}} of found jobs and resume details.Ø
- Client-side JavaScript renders results (i.e. list of jobs or extracted details) and pulls past applications through /get-applications.

8. **Cleanup**:

- Some therapists close the browser and delete temporary files (like resumes) from the uploads folder.

Results:

Simulated Job Search Outcomes

Job Role	Experience (Years)	Location Type	Location	Jobs Found	Applications Attempted	Applications Successful
Software Engineer	5	Remote	Worldwide	25	10	8
Data Scientist	3	On-Site	New York	15	8	6
DevOps Engineer	7	Hybrid	San Francisco	20	12	10
Front-End Developer	2	Remote	Worldwide	30	15	13
Cyber Security Analyst	4	On-Site	London	10	5	4
Cloud Engineer	6	Hybrid	Seattle	18	9	7
ML/AI Engineer	5	Remote	Worldwide	22	11	9
Mobile App Developer	3	Remote	Worldwide	17	8	6
Game Developer	2	On-Site	Los Angeles	12	6	5
Big Data Engineer	8	Hybrid	Chicago	19	10	8
Automation Tester	4	Remote	Worldwide	14	7	6
IT Project Manager	10	On-Site	Tokyo	16	8	7

Discussion:

- JobEase Bot effectively reduces manual effort in job applications by automating LinkedIn interactions and customizing resumes and cover letters using AI.
- It maintains a balance between automation and human control, ensuring users stay involved in key decisions like job preferences and resume uploads.
- The system's modular design—with Flask, Selenium, Gemini AI, pdfplumber, and SQLite—makes it scalable, flexible, and easy to enhance in future versions.
- While functional, it currently depends on LinkedIn's UI, lacks multi-platform support, and needs stronger security measures like encryption and 2FA.
- The bot shows how AI and automation can augment human productivity ethically and efficiently, making job hunting more accessible for diverse user groups.

Future Implications:

1. **Multi-Platform Expansion**:
   - Extend to Indeed, Glassdoor, etc., with modular scraping/APIs.
   - Broadens job sources beyond LinkedIn.
   - Uses flexible code to adapt to new platforms easily.
2. **Advanced AI**:
3. Improve job matching, personalization, and add interview prep using ML.
   - Enhances application quality with smarter job fits.
   - Prepares users for interviews with AI-generated content.
4. **Real-Time Features**:
   - Add notifications and a dashboard for application tracking.
   - Keeps users updated on application status instantly.
   - Provides a visual overview of job search progress.
5. **Scalability**:
   - Upgrade to PostgreSQL, multi-threading, and cloud hosting.
   - Supports more users with a robust database.
   - Speeds up processing and enables cloud access.
6. **Security**:
   - Encrypt data, handle 2FA, and ensure GDPR compliance.

- Protects user credentials and personal info.
- Meets legal standards for data privacy.

Key Benefits:

- **Time-Saving Automation:**
  JobEase Bot automates the entire job application process on LinkedIn, including login, job search, and form submission. This reduces the manual effort and time typically required by users.
- **AI-Powered Personalization:**
  It leverages Gemini AI to create tailored cover letters and resumes that align with specific job descriptions. This improves the quality and relevance of applications, boosting hiring potential.
- **User-Friendly Interface:**
  Designed using Flask and modern web technologies, the interface is simple and intuitive. Even users with minimal technical knowledge can easily operate the bot.
- **Efficient Application Tracking:**
  The system stores resume data and job application history using SQLite. This allows users to revisit past applications and manage their job search progress effectively.
- **Targeted Job Search:**
  Users can apply filters such as job role, experience level, and location (on-site, remote, hybrid). This ensures that applications are relevant and aligned with personal career goals.

Conclusion:

This project, "JobEase Bot," and can successfully show you how to automate searching and apply for jobs through LinkedIn, using a powerful combination of web scraping (Selenium), AI-based content generation (Google Gemini API) and a simple interface (Flask Web). The prototypes read details from resume, search for relevant listings and even simulates the 'apply' action with a simulated success rate of ~80% across varying roles like "Software Engineer" and "Front-End Developer" With SQLite for data persistence and a dynamic frontend, this is a useful tool for users to organize their job applications in one place.

The project is still a prototype and has its limitations such as depending on LinkedIn's UI changes, single platform cred with no dynamic method switching, and basic level error handling. While simulated results seem promising, identifying up to 30 jobs for high-demand roles, application in the real-world would involve critical issues like scalability, security (e.g. encrypted credentials), and adaptability to changes in the job board/platform structure.

In conclusion, through JobEase Bot, we demonstrate the ability of AI and automation to minimize the effort in the task of job applications, which has a considerable potential scope of enhancement such as multi-platform integration and contextual awareness, thus we believe that this serves as the first step towards developing a job application assistant.

REFERENCES

1. Schildknecht, L., Eißer, J., & Böhm, S. (2018). Motivators and barriers of chatbot usage in recruiting: An empirical study on the job candidates perspective in germany. *Journal of E-Technology*, *9*(4), 109.
2. Brandtzaeg, P. B., & Følstad, A. (2017). Why people use chatbots. In *Internet Science: 4th International Conference, INSCI 2017, Thessaloniki, Greece, November 22-24, 2017, Proceedings 4* (pp. 377-392). Springer International Publishing.
3. Nawaz, N., & Gomes, A. M. (2019). Artificial intelligence chatbots are new recruiters. *IJACSA) International Journal of Advanced Computer Science and Applications*, *10*(9).
4. Koivunen, S., Ala-Luopa, S., Olsson, T., & Haapakorpi, A. (2022). The march of Chatbots into recruitment: recruiters' experiences, expectations, and design opportunities. Computer Supported Cooperative Work (CSCW), 31(3), 487-516.
5. Mukherjee, S., & Patra, S. K. (2023). Chatbots: A review of their potential applications in library services. *Indian Journal of Information Library and Society*, *36*(1-2), 22-29.
6. Rahmani, D., & Kamberaj, H. (2021, May). Implementation and usage of artificial intelligence powered chatbots in human resources management systems. In *Conference: International conference on social and applied sciences at: University of New York Tirana*.
7. Shawar, B. A., & Atwell, E. (2007). Chatbots: are they really useful?. *Journal for Language Technology and Computational Linguistics*, *22*(1), 29-49.
8. Khan, S. (2020). Artificial intelligence virtual assistants (Chatbots) are innovative investigators. *IJCSNS*, *20*(2).
9. Koivunen, S., Ala-Luopa, S., Olsson, T., & Haapakorpi, A. (2022). The march of Chatbots into recruitment: recruiters' experiences, expectations, and design opportunities. *Computer Supported Cooperative Work (CSCW)*, *31*(3), 487-516.
10. Jitgosol, Y., Kasemvilas, S., & Boonchai, P. (2019, December). Designing an HR chatbot to support human resource management. In *Proceeding of the 5th SUIC international conference*.

11. Gupta, S., & Chen, Y. (2022). Supporting inclusive learning using chatbots? A chatbot-led interview study. *Journal of Information Systems Education*, *33*(1), 98-108.

12. Schildknecht, L., Eißer, J., & Böhm, S. (2018). Motivators and barriers of chatbot usage in recruiting: An empirical study on the job candidates perspective in germany. *Journal of E-Technology*, *9*(4), 109.

13. Kopplin, C. S. (2022). Chatbots in the workplace: A technology acceptance study applying uses and gratifications in coworking spaces. *Journal of Organizational Computing and Electronic Commerce*, *32*(3-4), 232-257.

14. [14] Patti, K. (2020). I forced a bot to write this book: AI meets BS. Andrews McMeel Publishing.