Job Application Assistant Tool: An Integrated Chrome Extension and AI-Powered Web Platform

Rojan Upreti
Department of Computer Science
William Paterson University
Wayne, NJ, USA
Email: upretir@student.wpunj.edu

***Abstract*—This paper presents an integrated job application assistant tool comprised of a Chrome browser extension ("JobFiller") and a companion web platform. The system addresses inefficiencies in online job applications by streamlining repetitive form filling, centralizing applicant profile data, and providing intelligent assistance. By leveraging Firebase for real-time data synchronization and an AI-driven keyword analyzer, the framework improves user productivity and consistency in applications. Key contributions include one-click form autofill across job portals and automated extraction of top relevant keywords from job descriptions to help candidates tailor resumes for Applicant Tracking Systems (ATS). The implementation and initial evaluations demonstrate enhanced efficiency in the job application process and outline future directions for more advanced intelligent assistance features.**

***Index Terms*—Job Application Automation, Chrome Extension, Artificial Intelligence, Firebase, ATS Optimization.**

I. Introduction

Job hunting is a time-consuming process often characterized by repetitive data entry and fragmented document management across multiple hiring platforms. Applicants frequently must re-type the same personal details and employment history on job sites such as LinkedIn, Indeed, and Workday, leading to frustration and increased likelihood of errors. Existing tools and platforms provide limited relief; for example, browser-based autofill features are generic and don't adapt well to varied application forms, and few systems offer unified profile management or guidance for optimizing application content. Furthermore, the rise of automated resume screening using Applicant Tracking Systems (ATS) means many qualified candidates are filtered out before human review if their resumes lack specific keywords. This creates a need for tools that not only save time but also improve the quality and ATS-friendliness of job applications.

Researchers and industry have explored intelligent form-filling and user assistance to mitigate these issues. Early work on auto-filling web forms established techniques for propagating user inputs across different applications. More recently, large language models have been applied to form-filling and content generation. For instance, Aveni et al. introduced OmniFill, a domain-agnostic system that uses contextual data and language models to suggest form inputs across various websites. Chen et al demonstrated that generative models can produce realistic input data to automate web application testing, hinting at the potential of AI in understanding and populating forms. Other studies have improved browser extensions for such tasks: Zhao and Tran [**?**] showed that a Chrome extension backed by cloud services (Firebase) can scale reliably for data-intensive tasks. Jin et al. [**?**] quantified the performance impact of extensions on browser responsiveness, underscoring the importance of lightweight design and caching in our approach. In addition, research on web form semantics and test generation [**?**] provides a foundation for understanding complex application forms, while real-time database synchronization techniques [**?**] inform our strategy for keeping data consistent between the extension and web app.

Project Objective: In light of these challenges and advances, we developed an Intelligent Job Application Assistant Tool that integrates a browser extension with a web application to streamline and enhance the job application workflow. The proposed system, called JobFiller, automatically fills out job application forms with a single click using stored profile data and helps users tailor their application materials. A key feature is an AI-powered keyword generator that analyzes job descriptions to extract the top relevant keywords, helping applicants customize their resume and cover letters to better pass ATS filters. By centralizing user data (personal information, work history, education, skills, and documents) in a cloud-backed web platform, and syncing this data with the Chrome extension in real time, the system ensures consistency and saves effort across multiple applications.

Contributions: This paper details the design and implementation of the Job Application Assistant Tool and discusses its significance. The integrated solution combines three core components – a Chrome extension, a dynamic job application portal, and a web dashboard – unified via a cloud backend. The system provides a unified profile management interface and cross-platform autofill capabilities, improving efficiency and reducing errors in the application process. It also introduces an AI-driven keyword analysis within the application workflow, which to our knowledge is a novel feature among job application management tools. We present the system architecture, key methodologies for data synchronization and AI integration, and an evaluation of the tool's functionality. The results indicate that our approach can significantly streamline job applications and potentially improve applicant success rates.

Future work will explore extending the tool's intelligence and reach, including integration with major job board platforms and advanced AI personalization.

II. System Architecture and Methodology

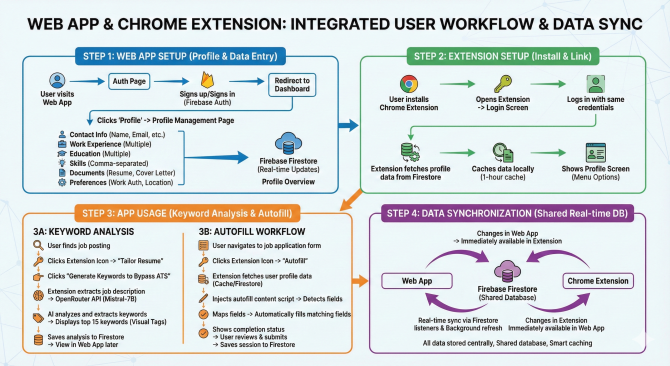

Fig. 1. System architecture and workflow of the Job Application Assistant Tool, illustrating the integrated user workflow. The process includes: (Step 1) user profile setup on the web app; (Step 2) Chrome extension installation and linking via authentication; (Step 3A) AI-driven keyword analysis of job descriptions; (Step 3B) one-click form autofill on job application pages; and (Step 4) real-time data synchronization between the web application and Chrome extension through a shared cloud database.

The Job Application Assistant Tool consists of a web application (cloud portal) and a Chrome browser extension that work in tandem, with Firebase serving as the cloud backend for authentication, data storage, and real-time synchronization. Figure 1 provides an overview of the system's architecture and user workflow.

In Step 1, users create an account or sign in on the web application and enter their profile details (contact information, work experience, education, skills, etc.), which are stored in a Firebase Firestore database. In Step 2, the user installs the Chrome extension (JobFiller) and logs in with the same credentials, linking the extension to their cloud profile. The extension immediately fetches the user's profile data from Firestore (or from a local cache if available) and provides a popup interface for quick actions.

Step 3 constitutes the main usage: Step 3A is the Keyword Analysis feature where the user, when viewing a job posting online, can click the extension's "Tailor Resume" option to perform an AI-driven analysis of the job description. The extension will extract the job description text from the page, send it to an AI service, and retrieve a list of recommended keywords (typically the top 10–15 relevant skills or terms) for the user to potentially include in their resume or cover letter. Step 3B is the Autofill Workflow: when the user is on a job application form (for example, on the custom portal or an external site), clicking the "Autofill" button in the extension causes it to inject the stored profile information into the form fields automatically.

Finally, Step 4 shows the continuous Data Synchronization: any changes the user makes to their profile via the web app are synced in real time to the extension's datastore (and vice versa), ensuring both components always use up-to-date information. This synchronization is achieved through Firebase's real-time database listeners and a smart caching mechanism in the extension, which we detail later. In the following subsections, we describe each major component and the methodology behind their implementation.

A. Chrome Extension (JobFiller)

The Chrome extension, JobFiller, is built on the Manifest V3 framework and serves as the client-side automation tool for form filling and job-specific analysis. It provides a lightweight popup interface (approximately 500×600 pixels) that is accessible by clicking the extension icon in the browser toolbar. The extension interface allows users to quickly view their profile data, download stored documents (like their resume or cover letter), and trigger autofill or keyword analysis actions. To maintain a responsive user experience in the browser, the extension was designed with performance considerations in mind, aligning with recent studies [**?**] that highlight the need for efficiency in extension development. The extension's key features and capabilities include:

- **Instant Profile Access:** Users can view their saved profile information (personal details, work history, etc.) directly from the extension popup. This provides a quick reference and ensures that the data to be filled into forms can be reviewed at a glance.
- **Document Management:** The extension allows downloading of documents stored in the user's profile (such as the resume or cover letter) so that users can easily attach them if a job portal requires file upload instead of text fields.
- **Smart Caching:** To minimize repeated network calls and improve speed, the extension caches the user's profile data locally for a short duration (e.g., a one-hour cache with background refresh). This means if the user repeatedly applies to multiple jobs in a short time, the extension can autofill forms without fetching from the cloud each time, significantly reducing latency.
- **Session Persistence:** The extension maintains user login sessions for up to seven days. Using Firebase Authentication tokens, the extension stays logged in (unless manually logged out), so users do not have to re-enter credentials frequently.
- **Autofill Automation:** The extension can detect common form fields on job application pages and inject the corresponding profile data. It uses content scripts to map profile fields to form input fields by matching field names or placeholders. Once the user clicks "Autofill", the extension script fills all matching fields and highlights them, then provides a completion status.
- **AI Keyword Generator:** The extension integrates an AI-powered feature labeled "Tailor Resume" that generates relevant keywords from a job description. When activated, it extracts the job posting content from the current page, sends the text to an AI API (OpenAI's GPT model or an equivalent via a service like OpenRouter using the

Mistral-7B model), and receives a set of key terms that are highly relevant to that job.

- **Data Sync and Custom API Calls:** Because Chrome Manifest V3 extensions have certain limitations in using Firebase SDKs directly, the extension employs a lightweight custom REST client approach to communicate with Firebase services. It interacts with Firebase Authentication for login and token management, Firestore for fetching and updating profile data, and Firebase Storage for retrieving document files.

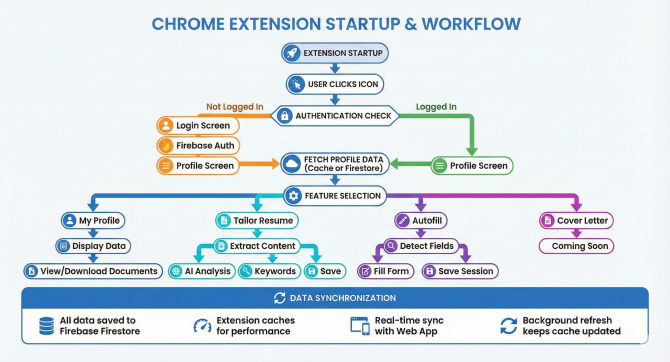

Fig. 2. Chrome extension startup and workflow, including authentication, data retrieval, and feature selection. The flowchart illustrates how the extension operates: when the JobFiller extension is clicked, it checks authentication and then fetches the user's profile data. In the Feature Selection stage, the user can choose among options like "My Profile", "Tailor Resume", or "Autofill". The Tailor Resume sequence (center path) shows the Keyword Generation Process. The Autofill sequence (right path) shows how the extension detects form fields and fills them with profile data.

Figure 2 depicts the internal workflow of the Chrome extension from startup to performing its key features. When the user clicks the extension icon, the extension initializes and checks if the user is authenticated. If not logged in, it prompts for login; upon successful login, it loads the user's Profile Screen. If the user is already logged in, the extension directly proceeds to fetch the latest profile data – either from its local cache (if the cache is fresh) or from the Firebase Cloud Firestore. Next is the feature selection step, where the extension presents the user with options. As illustrated in Figure 2, choosing Tailor Resume initiates the Keyword Analysis sequence. Choosing Autofill triggers the sequence of detecting all input fields in the current job application form, mapping each to the user's profile data, and then programmatically filling those fields.

B. Job Application Portal

In order to test and demonstrate the autofill capabilities in a controlled environment, the project includes a custom Job Application Portal that mimics a typical multi-step online job application form. This portal is a web-based form workflow (implemented as part of the web application) designed to resemble the user experience on real corporate career sites. The portal guides the user through five sequential steps:

1) **Contact Information & Resume Upload:** Collects basic personal details and provides an option to upload a resume file.
2) **Work Experience and Skills:** Asks for the user's employment history and relevant skills. This allows free-form text as well as structured entries.
3) **Education:** Gathers academic background information, potentially including multiple degrees or institutions.
4) **Work Preferences:** Covers preferences such as desired job location, willingness to travel or relocate, and other job-specific questions.
5) **Equal Employment Opportunity (EEO):** Includes voluntary demographic questions (commonly found in job applications for compliance), such as gender, race/ethnicity, and veteran status.

Each step of the portal form is built with client-side validation and features interactive elements. From the perspective of the JobFiller system, this portal serves as a target for the extension's autofill feature. All data submitted through the job portal is saved to the Firebase Firestore database under the user's profile.

C. AI Keyword Generator

A distinguishing component of this project is the AI Keyword Generator. The goal of this component is to help job applicants tailor their application materials to specific job postings by identifying the important keywords that ATS algorithms or recruiters might look for.

The keyword generation process works as follows: when the user is viewing a job description on any website, they can invoke the keyword generator via the extension. The extension will gather the textual content of the job description. Using the OpenAI API (or an alternative AI service like Mistral-7B via OpenRouter), the extension sends this job description text to a language model with a prompt to extract the most relevant keywords or phrases. The extension then filters and formats these keywords and presents the top ~ 15 to the user in its popup interface. We also log the AI-generated keywords to the Firestore database, attached to the user's profile and optionally linked with the specific job.

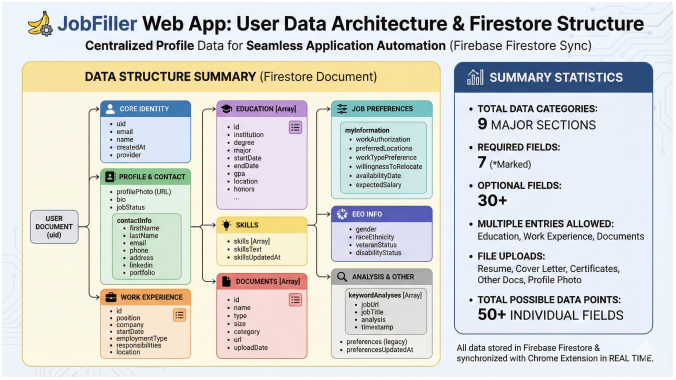

Fig. 3. Data Structure

D. Web Application and Integration

The web application serves as the central hub for the Job Application Assistant Tool, providing a user-facing dashboard

for managing profile data and tracking the job application process. It is a MERN-stack inspired web platform developed with Node.js and Express on the backend and standard web technologies on the client side.

When a user accesses the web app, they can create a profile with the following sections:

- **Personal Details:** Name, email, phone, address, and other contact info.
- **Work Experience:** A list of past job positions.
- **Education:** Academic qualifications.
- **Skills:** A set of skills or keywords the user identifies for themselves.
- **Documents:** Management of resume and cover letter files via Firebase Storage.
- **Job Preferences:** Optional fields such as desired location and salary range.

Crucially, the web app implements real-time listeners on the Firestore database so that any changes made by the extension are immediately reflected on the web dashboard. Additionally, the web app provides an Application Tracker view, which is a log of applications the user has submitted.

III. Results and Discussion

After implementing the Job Application Assistant Tool, we conducted a series of tests to validate its functionality. Key findings include:

- **Cross-Platform Data Consistency:** The integration via Firebase proved effective. Updating profile information on the web app was reflected in the Chrome extension almost instantly.
- **Autofill Efficiency:** Using the Chrome extension to fill out application forms significantly reduced the time and effort required. In tests with the five-step job portal form, the extension was able to populate all fields in roughly 2–3 seconds, whereas manual entry could take several minutes.
- **AI Keyword Relevance:** The "Tailor Resume" keyword suggestions feature was tested with a variety of job postings. In each case, the extension returned a list of key terms that generally matched the important skills and qualifications in the description.
- **Performance and Reliability:** The Chrome extension's memory footprint remained low, and its use of caching minimized redundant network calls. The web application was responsive, with page loads and dynamic updates happening quickly.
- **User Experience:** Informal feedback was gathered from a few users who tried the tool. They appreciated the time saved by autofill and found the interface of both the extension and web app intuitive.

One challenge encountered was ensuring the extension's autofill works across different sites with varying HTML structures. While our custom portal provided a consistent testbed, real-world job applications sometimes have non-standard form fields.

IV. Conclusion and Future Work

This project demonstrates the design and implementation of an integrated job application assistant that combines web technologies, cloud services, and artificial intelligence to improve the job search experience. The Job Application Assistant Tool streamlines repetitive tasks by unifying user data in a central profile and automatically populating job application forms via a Chrome extension.

Moving forward, there are several avenues to enrich and extend this work:

- **Broader Autofill Integration:** Expand the extension's autofill capability to support major job boards and application systems such as LinkedIn, Indeed, Workday, and Taleo.
- **Advanced AI Assistance:** Integrate more powerful AI models to provide deeper customization of application materials, such as auto-generating tailored cover letters.
- **Application Tracking & Analytics:** Develop a comprehensive application tracking system within the web app with dashboards and analytics.
- **Notifications and Real-Time Updates:** Implement a notification mechanism to alert users about key events.
- **Privacy and Trust Enhancements:** Future research should explore how to perform keyword analysis or form autofill in a way that minimizes exposure of user data.

References

[1] T. J. Aveni, A. Fox, and B. Hartmann, "OmniFill: Domain-Agnostic Form Filling Suggestions Using Multi-Faceted Context," *arXiv preprint arXiv:2310.17826*, 2023.

[2] T. Zhao and K. Tran, "A Powerful Chrome Extension: Translation Program Using Python, Website Analysis and Google Firebase Services," in *Proc. CS & IT Conf.*, vol. 13, no. 7, 2023.

[3] B. Jin, H. Li, and Y. Zou, "Impact of Extensions on Browser Performance: An Empirical Study on Google Chrome," *Empirical Software Engineering*, vol. 30, no. 103, 2025.

[4] P. Alian *et al.*, "Semantic Constraint Inference for Web Form Test Generation," in *Proc. ISSTA*, pp. 932–944, 2024.

[5] F. Chen *et al.*, "Using Large Language Model to Fill in Web Forms to Support Automated Web Application Testing," *Information*, vol. 16, no. 2, p. 102, 2025.

[6] JSREM Journal, "Real-Time Database Synchronization Techniques in Firebase for Mobile App Development," *IJSREM*, vol. 7, pp. 1–10, 2023.

[7] X. Lin, P. Ilia, and J. Polakis, "Fill in the Blanks: Empirical Analysis of the Privacy Threats of Browser Form Autofill," in *Proc. ACM CCS*, pp. 1–13, 2020.

[8] E. Habibi and S. Mirian-Hosseinabadi, "Event-Driven Web Application Testing Based on Model-Based Mutation Testing," *Information and Software Technology*, vol. 67, pp. 159–179, 2015.

[9] T. Furche *et al.*, "OPAL: Automated Form Understanding for the Deep Web," in *Proc. 21st Int. Conf. on WWW*, pp. 829–838, 2012.

[10] L. Yang and Q. Zeng, "Autofill Browser Extension: A User-Friendly Solution to Optimize Metadata Workflow in Digital Repositories," *Journal of Library Metadata*, vol. 16, no. 1, pp. 44–52, 2016.

[11] R. De Virgilio and R. Torlone, "Modeling Heterogeneous Context Information in Adaptive Web-Based Applications," in *Proc. 6th Int. Conf. on Web Engineering*, pp. 56–66, 2022.