# JobBot Project:

End-to-End Automated Tailored Job Application Generator

### Project Overview

JobBot is an automated system that takes job descriptions clipped from the web, extracts key insights using OpenAI’s GPT models, and generates tailored resumes and cover letters customized for each job. It then converts these documents to PDFs, uploads them to Google Drive, and updates a Google Sheet with links and status, ready for a virtual assistant (VA) to use for job applications.

### Step-by-Step Pipeline

**1. Clip Job Descriptions from the Web**

* **Purpose:** Capture job description data directly from job posting websites into a Google Sheet.
* **How:** A custom **Chrome extension** or bookmarklet runs JavaScript on the job page to extract:
  + Job title
  + URL
  + Full job description text
* **Representative file:**
  + javascript\_to\_clip2GS.txt — contains the JavaScript snippet used to clip job data and send it via a POST request to a Google Apps Script Web App, which appends the data as a new row in the Google Sheet.

**2. Google Sheet as Central Job Data Store**

* **Purpose:** Store all clipped job descriptions as rows with columns like Timestamp, Title, URL, JD (Job Description), Status, Resume Link, Cover Letter Link, Folder Link.
* **How:** The Google Sheet acts as the single source of truth for job data and tracks processing status.
* **Sheet columns:** Timestamp | Title | URL | JD | Status | Resume Link | Cover Letter Link | Folder Link

**3. Main Orchestration Script —**main.py

* **Purpose:** Automate the entire pipeline for each new job row in the Google Sheet.
* **How:** For each job row with Status not marked as done:
  1. **Save the job description text** to a .txt file.
  2. **Extract keywords and insights** from the job description using GPT-3.5 (keyword\_extraction\_gpt3\_5.py).
  3. **Generate a tailored resume** using GPT-4o (write\_updated\_resume.py), feeding in:
     + The extracted keywords JSON.
     + The master resume text.
  4. **Generate a diff** between the original and tailored resume to see changes.
  5. **Generate an AI-powered summary** of changes for review.
  6. **Generate a tailored cover letter** using GPT-4o (write\_cover\_letter.py).
  7. **Render both resume and cover letter Markdown files to PDFs** (generate\_pdf\_resume.py and generate\_pdf\_cover.py).
  8. **Upload PDFs to Google Drive and update the Google Sheet** with links and status (upload\_to\_drive\_and\_update\_sheet.py).
* **Key features:**
  1. Handles file naming consistently using sanitized job titles.
  2. Skips rows already marked done.
  3. Saves all outputs in a pipeline\_output/ folder.
  4. Uses subprocess calls to modular scripts for PDF generation and upload.
  5. Includes detailed logging and error handling.

**4. Keyword Extraction —**keyword\_extraction\_gpt3\_5.py

* **Purpose:** Analyze the job description text to extract key role insights, responsibilities, skills, culture, and keywords useful for tailoring resumes.
* **How:** Sends the job description text to GPT-3.5 with a system prompt guiding it to output JSON-formatted insights.
* **Output:** JSON file with keywords and role details saved as {jobname}\_keywords.json.

**5. Tailored Resume Generation —**write\_updated\_resume.py

* **Purpose:** Use the extracted keywords and master resume to generate a customized resume optimized for the specific job.
* **How:** Sends a prompt to GPT-4o including:
  + The keyword JSON.
  + The master resume text.
  + Instructions for formatting, keyword density, tone, and section structure.
* **Output:** Markdown-formatted resume saved as {jobname}\_resume.txt.

**6. Resume Diff Generation (In**main.py**)**

* **Purpose:** Compare the original master resume and the tailored resume to highlight changes.
* **How:** Uses Python’s difflib to generate a unified diff text file.
* **Output:** Diff saved as {jobname}\_resume\_diff.txt.

**7. AI-Powered Resume Change Summary (In**main.py**)**

* **Purpose:** Generate a natural language summary of what changed in the tailored resume.
* **How:** Sends both original and tailored resumes to GPT-4o with a prompt to summarize key edits, additions, removals, tone shifts.
* **Output:** Summary saved as {jobname}\_resume\_change\_summary.txt.

**8. Cover Letter Generation —**write\_cover\_letter.py

* **Purpose:** Generate a one-page, compelling cover letter tailored to the job description and company culture.
* **How:** Sends a prompt to GPT-4o including:
  + The keyword JSON.
  + Optional snippets from the tailored resume.
  + Detailed instructions on structure, tone, and length.
* **Output:** Markdown cover letter saved as {jobname}\_coverletter.txt.

**9. PDF Rendering —**generate\_pdf\_resume.py**&**generate\_pdf\_cover.py

* **Purpose:** Convert Markdown resume and cover letter files into styled PDFs.
* **How:** Reads Markdown files, converts to HTML with markdown2, applies CSS styles, then renders PDFs using pdfkit (which requires wkhtmltopdf installed).
* **Output:** PDFs saved as {jobname}\_resume.pdf and {jobname}\_coverletter.pdf.

**10. Google Drive Upload and Sheet Update —**upload\_to\_drive\_and\_update\_sheet.py

* **Purpose:** Upload generated PDFs to a dedicated Google Drive folder and update the Google Sheet row with links and status.
* **How:**
  + Uses Google Drive API to find or create a job-specific folder inside a parent folder.
  + Uploads resume and cover letter PDFs, sets sharing permissions.
  + Updates the Google Sheet row with:
    - Status = "Done"
    - Shareable links to resume PDF, cover letter PDF, and Drive folder.
* **Input:** Receives the Google Sheet row number and sanitized jobname as command-line arguments from main.py.
* **Output:** Updated Google Sheet row with URLs and status.

### Additional Notes

* **File Naming Consistency:** All files use sanitized job titles for consistent and predictable naming, enabling easy chaining across modules.
* **Error Handling & Debugging:** Each step includes logging, token count checks, and fallback parsing to ensure robustness.
* **Modular Design:** Each major step is a standalone Python script or function, making maintenance and upgrades easy.
* **VA Support:** The final Google Sheet with live links and status allows a VA to quickly access customized application materials and submit applications efficiently.

### Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| Step # | Purpose | Script/File | Output Files/Artifacts |
| 1 | Clip job description to Google Sheet | javascript\_to\_clip2GS.txt | New row in Google Sheet with job info |
| 2 | Main orchestration | main.py | Controls workflow, calls all other scripts |
| 3 | Extract keywords from job description | keyword\_extraction\_gpt3\_5.py | {jobname}\_keywords.json |
| 4 | Generate tailored resume | write\_updated\_resume.py | {jobname}\_resume.txt |
| 5 | Generate resume diff | main.py (uses difflib) | {jobname}\_resume\_diff.txt |
| 6 | Generate AI summary of resume changes | main.py (calls GPT-4o) | {jobname}\_resume\_change\_summary.txt |
| 7 | Generate tailored cover letter | write\_cover\_letter.py | {jobname}\_coverletter.txt |
| 8 | Render PDFs from Markdown | generate\_pdf\_resume.py, generate\_pdf\_cover.py | {jobname}\_resume.pdf, {jobname}\_coverletter.pdf |
| 9 | Upload PDFs and update Google Sheet | upload\_to\_drive\_and\_update\_sheet.py | Updates Google Sheet row with links and "Done" status |

### Final overview

**This pipeline combines:**

* Browser automation (clipper),
* Cloud spreadsheets (Google Sheets),
* AI content generation (OpenAI GPT-3.5 & GPT-4o),
* Document processing (Markdown → PDF),
* Cloud storage & sharing (Google Drive),
* Workflow orchestration (modular Python scripts).

It’s designed to be robust, modular, and easily extensible for scaling job application automation and supporting my virtual assistant (VA) team.