## 4. Offer Export Options (PDF/CSV) for the Analysis Report

**How to:** Use Python libraries such as pandas to export tabular data (skills matched, missing) to CSV.

Use FPDF or reportlab to create PDF reports summarizing score, matches, explanations.

**Implementation:** Add buttons in Streamlit to download CSV/PDF dynamically generated from results.

Structure the report clearly with ATS score, matched/missing skills, semantic scores, and recommendations.

## 5. Allow Recruiters or Users to Upload Multiple Resumes for Bulk Processing

**How to:** Modify your Streamlit uploader to accept multiple files (accept\_multiple\_files=True).

Loop through each resume file and analyze sequentially or in batch.

**Implementation:** Display a summary table with each resume filename, ATS score, and key feedback.

Allow exporting batch results as CSV for easy recruiter review.

## 7. Add Multi-Word and Phrase Matching for Important Compound Skills

**How to:** Enhance master\_skills.csv to include multi-word skills like “unit testing”, “cloud computing”, “data science”.

Normalize skill extraction logic to handle longer phrases using regex or phrase matching.

**Implementation:** Match using full phrase regex patterns with word boundaries for accurate multi-word detection.

Maintain skill aliases for phrases too for normalization (e.g., "ci/cd", "continuous integration").

## 8. Incorporate Named Entity Recognition (NER) or Custom Keyword Extractors

**How to:** Use spaCy's NER capabilities or train your own NER models fine-tuned on recruitment data to extract skill entities.

Use libraries like flashtext or RAKE or custom rule-based extractors for domain-specific terms.

**Implementation:** Combine NER outputs with skill dictionary matching for more comprehensive extraction.

Use this to supplement or improve skill detection beyond exact string matching.

## Brand Names & Taglines

* **ResumiQ** Tagline: “Smart Insights. Stronger Resumes.”
* **SkillSnapshot** Tagline: “Picture Your Potential.”
* **HireVibe** Tagline: “Turn Resumes Into Offers.”
* **VerveScan** Tagline: “Spotlight Your Edge. Land Your Role.”
* **CVisionary** Tagline: “See Beyond The Resume.”
* **AspireAtlas** Tagline: “Map Your Path. Maximize Your Impact.”
* **LaunchSheet** Tagline: “Resume. Refined. Ready for Takeoff.”
* **TalentTune** Tagline: “Optimize Your Story. Amplify Success.”
* **NextStep Analyzer** Tagline: “Your Resume’s Best Future, Today.”
* **CareerCanvas** Tagline: “Paint a Brighter Future.”

Here’s a list of components you can add to your resume analyzer homepage to make it more engaging, informative, and user-friendly:

## Recommended Homepage Components

**Navigation Menu:** Home, Features, About, Contact, Dashboard, Login, etc.

**How It Works / Steps** Short, visual steps or a small infographic explaining how your analyzer functions for different user roles.

**Testimonials or Success Stories** Authentic student/recruiter feedback (can be placeholders/raw quotes for now).

**Demo Video** (or animation) Quick overview showing the resume analyzer in action..

**FAQs** Common questions and answers about your service.

**Contact Us / Support** Email, chat button, or contact form for inquiries.

**Newsletter Signup** Collect emails for launch updates and promotions.

**Social Proof/Partners** Logos of universities, companies, or tools you integrate or aspire to integrate with.

## Optional Components for Added Value

**Blog or Updates Feed** Latest news, tips, and career advice.

**Interactive Resume Analyzer Demo** Mini version where users can test with sample resumes.

**Downloadable Resources** Resume templates, guides, etc.

## Next-Level Enhancements

**Weighting by Skill Importance**

Not all JD skills are equal (e.g., “Python” more important than “Excel”).

You could weight based on order in JD, frequency, or even job seniority level.

**Experience Context Matching**

Distinguish between a skill in a project/experience section vs. just listed in “Skills” → recruiters value demonstrated usage more.

**Section-Aware Warnings**

If JD demands “3+ years” and resume shows “Internship only” → flag as gap.

**Fairness / Bias Probe**

Optional: check for unnecessary demographic data (photo, marital status, age, etc.), which can bias ATS and recruiters.

### ****Core Recruiter Dashboard Features****

**Candidate Resume Upload / Pool Management**

Upload multiple resumes at once (PDF/DOCX).

Store resumes in a searchable candidate pool.

Auto-extraction of skills, education, work history.

**Job Description Management**

Create, save, and edit job postings.

Ability to assign multiple JDs to one candidate pool.

Tag JDs by department/role.

**ATS Match Scores (Multi-Candidate View)**

View candidates ranked by ATS score for a given JD.

Side-by-side comparison: Candidate vs JD required skills.

Color-coded score bands (green = strong match, yellow = partial, red = weak).

**Advanced Insights**

**Skill Gap Analysis**

See missing skills for each candidate directly in the recruiter view.

Aggregate "most missing skills" across all candidates → helps recruiters refine job postings.

**Experience Gap Alerts**

Auto-flag underqualified (too few years of experience) and overqualified candidates.

Option to filter out resumes below minimum requirements.

**Semantic Search**

Recruiters type: “Python Django backend engineer 5+ YOE” → the system ranks candidates semantically, not just by keywords.

**Diversity & Fairness Indicators**

Anonymized candidate view (hide name, gender, photo, university initially).

Helps reduce unconscious bias.

### ****Fairness & Transparency Tools (CVisionary Differentiators)****

**Scoring Transparency (Recruiter View)**

Show recruiters exactly how a candidate scored (semantic vs keyword vs experience).

Helps them trust the ranking system.

**Customizable Scoring Weights**

Allow recruiters to adjust sliders (e.g., more weight to experience, less to skills).

Re-rank candidates instantly.

**Candidate Insights Report**

One-click download: "Why Candidate X scored 62% — missing Spring Boot, Kubernetes, 3+ YOE gap."

Helps recruiters justify rejection/selection decisions.

**Bias Warnings**

Detect if the JD has biased language (e.g., “rockstar developer”, “native English speaker”).

Suggest inclusive alternatives.

### ****Future-Proof Add-ons****

**Recruiter–Student Communication**

Chat or schedule interviews directly from the dashboard.

Integration with email/LinkedIn.

**Team Collaboration**

Recruiter team members can leave notes/tags on candidates.

Voting system → “Hire”, “Maybe”, “Reject”.

**Analytics Dashboard**

Which JDs get most matches.

Time-to-fill metrics.

Candidate skill heatmaps (what skills are most common/least common).

If you want, I can design a **Recruiter Dashboard wireframe UI** (React + Tailwind) showing these sections — candidate ranking table, filters, job posting manager, and transparency panel.

## Step 1. Add Authentication (Login/Signup)

Use **JWT (JSON Web Tokens)** for backend auth.

Frontend: React + react-router-dom + axios + context **B**ackend (FastAPI )

/auth/signup → Register user (role: student/recruiter).

/auth/login → Issue JWT token.

Store hashed password (bcrypt).

👉 In React, store the token in localStorage and attach it with every request:

axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

## 📤 Step 2. Student Upload Workflow

Once a student calculates their ATS score → give them an **"Upload Resume to CVisionary"** button.

### Backend (FastAPI)

Endpoint: /student/upload-resume

Input: resume file, job description, user\_id (from JWT).

Save in DB (Postgres/Mongo):

Resume text (parsed).

Resume file (S3/Cloud storage or local path).

Calculated ATS score + metadata.

## Step 3. Recruiter Dashboard – Candidate Ranking

Recruiter selects a **JD** → backend queries all resumes uploaded against it, then sorts by **ATS score**.

### Backend

Endpoint: /recruiter/candidates?jd\_id=123

Returns list of resumes with:

Candidate name

Score

Skills matched/missing

Experience gap

### Frontend: Show as a ****ranked table****:

| **Rank** | **Candidate** | **ATS Score** | **Matched Skills** | **Missing Skills** | **Experience Gap** |
| --- | --- | --- | --- | --- | --- |
| 1 | Alice | 82% | Python, Django | Kubernetes | No gap |
| 2 | Bob | 64% | C++, SQL | Spring Boot | Gap: Needs 5+ YOE |

## 🔐 Step 4. Role-Based Access

Student = upload + view own scores.

Recruiter = view all resumes uploaded for JD.

Middleware checks role from JWT before serving endpoints.

## Step 5. Database Schema (High-Level)

You’ll need **3 main tables/collections**:

### users

id

name

email

password (hashed)

role (student/recruiter)

### resumes

id

user\_id (FK → users)

file\_path / s3\_url

text\_extracted

ATS score

matched\_skills, missing\_skills, warnings

### job\_descriptions

id

recruiter\_id (FK → users)

jd\_text

created\_at

Recruiter dashboard joins job\_descriptions with resumes → ranks candidates.

## ✅ Roadmap Order

Add **Auth system** (students + recruiters).

Add **Resume upload + ATS score storage** for students.

Add **Job description manager** for recruiters.

Add **Candidate ranking table** in recruiter dashboard.

(Later) Add collaboration, fairness tools, analytics.