Frontend Integration Guide – AI Resume Screening (FastAPI + Supabase) 🎯🚀📘

This guide provides everything a frontend engineer (Next.js or any web frontend) needs to integrate with the AI Resume Screening backend API. 💡🔧🧩

Flow Pipeline 🔁📈📁

1. User logs in using Supabase (Google Sign-In)
2. User sees their previous screenings (/my-screenings)
3. User clicks Add Screening, fills form, uploads ZIP of resumes
4. Screening is processed → Ranking calculated → Results displayed
5. User can:
    - Compare candidates
    - Filter & search
    - Add notes/tags
    - Export CSV
    - Re-screen with new weights

Authentication (Supabase) 🔐🧠💻

1. Install SDK

npm install @supabase/supabase-js

2. Initialize Supabase Client

import { createClient } from '@supabase/supabase-js';

export const supabase = createClient('https://<project>.supabase.co', '<anon_key>');

3. Google Sign-In

await supabase.auth.signInWithOAuth({ provider: 'google' });

4. Get Access Token

const session = await supabase.auth.getSession();

const accessToken = session.data.session.access_token;

Use this token in every request as:

Authorization: Bearer <accessToken>

API + Integration 🔭📨🧪

1. /my-screenings

GET /my-screenings

Authorization: Bearer <token>

Displays a list of job titles, job IDs, and creation dates 🗂️📋🗓️

---

2. Upload New Screening 📥📝📜

POST /upload-resumes/

Content-Type: multipart/form-data

Authorization: Bearer <token>

Form fields:

- file: zip file of resumes
- job_title: string
- job_description: string
- weight_experience: int
- weight_projects: int

Server extracts resumes, analyzes them, ranks candidates. Returns: 📊🧠✅

{ "message": "Resumes processed!", "job_id": "uuid" }

---

3. Fetch Ranked Candidates 🧮📈📑

GET /export?job_id=<id>&format=json

Returns detailed ranked list:

[{ resume_id, total_score, rank, analysis, upload }, ...]

Optional: use format=csv to trigger a CSV download 🗃️📥📊

---

4. View Analytics 📊🔍📌

GET /analytics?job_id=...`

Returns:

- total resumes
- top skills
- score distribution

---

5. Search Resumes 🔎💬📄

GET /search?job_id=...&query=python

Returns list of matching resumes by skill, content

---

6. Compare Candidates ⚖️👥🧠

GET /compare-candidates?resume_ids=<id1>&resume_ids=<id2>

Returns side-by-side comparison data

---

7. Add or Get Notes 📝📌🔗

POST /add-note

{ resume_id, notes, tagged_users }

GET /notes?resume_id=... → returns note, tags

---

8. Re-Screen with New Weights ♻️⚙️📈

POST /re-screen

{ job_id, weight_experience, weight_projects }

Updates score & reranks

---

9. Resume History Across Screenings 📜📁⏳

GET /history?resume_id=...

Returns prior job_ids where this resume appeared

UI Tips 🧰🎨📐

Feature	UI Element Suggestion
Upload	Modal with file input + weights form
Ranked List	Table with sorting, highlighting, filters
Compare	Side-by-side profile cards or split grid

Feature	UI Element Suggestion
Notes	Sticky sidebar or in-row dropdown
Export	Button for CSV or JSON download
Re-screen	Sliders + resubmit button

Folder Outputs 📂 📄 🗂️

- /resumes/{job_id} → where individual resumes are stored
- /processed_data/resume_analysis.json → final structured analysis

---

JWT Validation on Backend 🔑 📜 🔍

If needed, JWT is decoded to get user_id in FastAPI:

import jwt

user_id = jwt.decode(token, options={"verify_signature": False})["sub"]

Ready to Go! 🎉 ✅ 🛠️

The backend is complete, documented, and supports all features. You can start plugging in frontend components now.

Need mock data or frontend code snippets? Just ask! 🧪 📦 🔧