HR Agent

Goal

Build a **chat-only** assistant that helps a recruiter:
search candidates → save a shortlist → draft an outreach email (HTML preview) → show tiny analytics.

Must-Have Features

1. **Candidate Search**
   - Input: free text (e.g., “React intern in Casablanca, 0–2y, available this month”).
   - Output: Top-N candidates with a short “why matched” note.
2. **Shortlist**
   - Save selected indices as a named shortlist (e.g., “Frontend-Intern-A”).
3. **Email Draft (Preview Only)**
   - Generate **subject + plain text + simple HTML** for either a single candidate or a shortlist.
   - Show preview; allow one edit to subject or a closing line; re-preview.
4. **Tiny Analytics**
   - Return counts by **stage** and top **3 skills** frequency across the current candidate set.

Data (JSON files or MongoDb)

Create a `data/` folder with two files:
`candidates.json` (≥12 items)

```JavaScript
[
  {
    "firstName": "Amina",
    "lastName": "El Idrissi",
    "email": "amina@example.com",
    "location": "Casablanca",
    "experienceYears": 1,
    "skills": ["React", "JS", "HTML"],
    "availabilityDate": "2025-10-01",
    "stage": "SOURCED",
    "notes": "Portfolio: …"
  }
]
```

`jobs.json` (2–3 items max)

```JavaScript
[
  {
    "title": "Frontend Intern",
    "location": "Casablanca",
    "skillsRequired": ["React", "JS", "Git"],
    "jdSnippet": "We build UI with React and Git
workflows."
  }
]
```

Shortlists can be kept in memory or a tiny `shortlists.json` you write to.

Minimal Scoring (keep it simple)

```JavaScript
score = +2 per required skill match
      +1 if location exact match
      +1 if experience within user range (±1 year ok)
      +1 if availabilityDate within next 45 days
```

Return Top-5 with a one-line reason, e.g.
“React+JS match (+4), Casablanca (+1), 1y fits (±1) → score 6”

Required Functions

- `parse_query(text) -> {role?, skills[], location?, minExp?, maxExp?, availabilityWindowDays?}`
- `search_candidates(filters) -> [{candidate, score, reason}]`
- `save_shortlist(name, candidate_indices) -> ok`
- `draft_email(recipients, job_title, tone='friendly') -> {subject, text}`
- `html_template(email) -> "<html>..."` (simple inline CSS)
- `analytics_summary() -> {countByStage, topSkills: [(skill, count)]}`

  Keep one small **prompt chain**:
  `classify intent → extract entities (parse_query) → route to function → (if email) generate text → wrap HTML → preview`.

CLI Flow example:

- `> Find 5 React interns in Casablanca with 0–2 years, available this month`
- `> Save #1 #3 #4 as "FE-Intern-A"`
- `> Draft outreach email for "FE-Intern-A" using job "Frontend Intern"`
- Assistant prints **subject + HTML preview** and asks: “Edit subject or closing? (y/n)”

- `> Show analytics`

Tiny Analytics (example output)

```JavaScript
Pipeline by stage: SOURCED=7, SCREEN=3, INTERVIEW=2
Top skills: React(8), Python(6), SQL(5)
```

Deliverables

1. **Single Python file or notebook** (no frameworks).
2. `data/` folder with `candidates.json`, `jobs.json`.
3. **README** (how to run + 3 example prompts).
4. **Screenshot** of the HTML preview in the console (printed string is fine).

Evaluation (50 pts total)

- Intent & routing work (10)
- Matching quality + reasons (10)
- Email draft + HTML preview (15)
- Analytics correctness (10)
- Code clarity & README (5)

Seed Prompts (copy/paste)

1. `Find top 5 React interns in Casablanca, 0–2 years, available this month`
2. `Save #1 #3 as "FE-Intern-A"`
3. `Draft an outreach email for "FE-Intern-A" using job "Frontend Intern" in friendly tone`
4. `Change the subject to "Quick chat about a Frontend Intern role?" and re-preview`
5. `Show analytics`