AI HR MS

📄 Product Requirements Document (PRD)

Project Name: AI-Powered HRMS – Recruitment Intelligence Module

Theme: Build the Future of HR Management with AI-Powered Solutions

Version: Hackathon MVP v1.1

---

1. 🎯 Objective

We are building the **Recruitment Intelligence Module** of a next-gen HRMS.

This MVP must:

- Parse resumes and extract structured data.
- Scrape and analyze links inside resumes (GitHub, LinkedIn, portfolio).
- Match candidates against Job Descriptions (JDs) with AI scoring.
- Provide recruiters with a dashboard showing ranked candidates and AI insights.
- Conduct conversational AI screening (chat/voice) and generate evaluation reports.

---

2. 🧑‍💻 User Roles

- **Candidate**
    - Upload resume.
    - Apply to jobs.
    - Participate in AI screening.
- **Recruiter**
    - Post job descriptions.
    - View ranked candidates.
    - Review AI screening results.

- **Admin**
    - View system-wide stats (number of candidates processed, average fit scores).

---

3. 📦 Core Features (MVP Scope)

3.1 Resume Upload & Parsing

- Candidate uploads PDF/DOC.
- AI (Gemini via Python service) extracts: skills, education, experience.
- Extracts links (GitHub, LinkedIn, portfolio).
- Scrapes metadata from those links.
- Stores structured data in Supabase.

3.2 AI Matching & Scoring

- Recruiter creates JD in system.
- AI compares candidate profile + external data with JD.
- Generates fit score (0–100).
- Provides explainable highlights (e.g., "Strong in Python, missing AWS").

3.3 Recruiter Dashboard

- Role-based access via Supabase Auth.
- Displays candidate list with scores.
- Shows digital footprint cards (GitHub activity, LinkedIn summary).
- Shows AI summary card (strengths, weaknesses, recommendations).

3.4 Conversational Screening

- AI chatbot/voice bot asks 2–3 adaptive questions.
- Candidate answers via text/voice.
- AI generates transcript + evaluation (communication, domain knowledge, overall score).
- Recruiter sees report in dashboard.

---

4. 🚫 Out of Scope (Hackathon MVP)

- Payroll, attendance, performance management.
- Multi-language support.
- Enterprise integrations (ERP, ATS).
- Advanced analytics.

---

5. 🏗️ Tech Stack

- **Frontend:** Next.js (React) + TailwindCSS + Shadcn
- **Backend (Data):** Supabase (Postgres + Auth + Storage)
- **Backend (AI):** Python (FastAPI) microservice
    - Uses Gemini via OpenRouter API
- **Auth:** Supabase Auth (RBAC: Admin, Recruiter, Candidate)
- **Hosting:**
    - Frontend → Vercel
    - Python AI service → Render
    - Supabase → Managed service
- **Testing:** PyTest (backend)
- **CI/CD:** GitHub Actions

---

6. 📊 Data Model (Supabase)

Tables:

- `candidates`
    - id, name, email, resume_url, parsed_data (JSONB)
- `jobs`
    - id, title, description, requirements
- `applications`
    - id, candidate_id, job_id, fit_score, highlights
- `screenings`

    - id, application_id, transcript, ai_summary, score
- `digital_footprints`
    - id, candidate_id, github_data, linkedin_data, portfolio_data

---

7. 📂 Folder Structure

```
ai-hrms/
├── frontend/ (Next.js)
│   ├── app/(auth)/login/
│   ├── app/(dashboard)/recruiter/
│   ├── app/(dashboard)/admin/
│   ├── app/candidates/upload/
│   ├── app/jobs/[id]/apply/
│   ├── components/
│   └── lib/{supabaseClient.ts, api.ts, auth.ts}
│
├── backend/ (FastAPI)
│   ├── app/main.py
│   ├── app/api/{candidates.py, jobs.py, applications.py, screenings.py}
│   ├── app/services/{ai_parser.py, ai_matching.py, ai_screening.py, link_s
craper.py}
│   ├── app/models/{candidate.py, job.py, screening.py}
│   └── app/core/{config.py, security.py, logging.py}
│
├── supabase/migrations/
│   ├── 001_init.sql
│   └── 002_rls.sql
│
├── .github/workflows/{ci.yml, deploy.yml}
├── .env.example
└── README.md
```

---

8. ✅ Success Metrics

- Resume parsing accuracy: >85% correct extraction.
- Recruiter agrees with AI ranking in >70% of cases.

- Conversational screening reduces manual screening time by 50%.
- Demo flow runs end-to-end in <3 minutes.

---

9. 🚀 Demo Flow

1. Candidate uploads resume → AI parses + enriches with GitHub/LinkedIn.
2. Recruiter posts JD → AI ranks candidates.
3. Recruiter clicks candidate → sees fit score + digital footprint card.
4. Recruiter triggers conversational screening → AI asks 3 questions.
5. Recruiter sees transcript + AI evaluation → decides to shortlist.

---

10. 📅 Future Extensions

- Add attendance, payroll, performance modules.
- Expand dashboards for Managers and Employees.
- Integrate calendars for interview scheduling.
- Add analytics (time-to-hire, attrition prediction).

---

11. 📋 Task Breakdown (Hackathon-Friendly)

Frontend

- [ ] Setup Next.js project with Tailwind + Shadcn.
- [ ] Implement Supabase Auth (login, role-based redirects).
- [ ] Candidate: Resume upload page.
- [ ] Candidate: Job application page.
- [ ] Recruiter: Dashboard with candidate list + scores.
- [ ] Recruiter: Candidate detail view (fit score, digital footprint, screening report).
- [ ] Admin: Simple stats dashboard.

Backend (Supabase)

- [ ] Create tables: candidates, jobs, applications, screenings, digital_footprints.

- [ ] Define RLS policies for role-based access.
- [ ] Setup Supabase storage for resumes.

Backend (Python AI Service)

- [ ] Setup FastAPI project.
- [ ] Implement `/parse_resume` → calls Gemini, extracts data, scrapes links.
- [ ] Implement `/match_candidate` → compares candidate vs JD, returns score + highlights.
- [ ] Implement `/screening` → runs conversational AI, returns transcript + evaluation.
- [ ] Write PyTest tests for each service.

Integration

- [ ] Connect frontend → Supabase (auth, data).
- [ ] Connect frontend → FastAPI AI endpoints.
- [ ] Display AI results in recruiter dashboard.

DevOps

- [ ] Setup GitHub Actions (lint, test, build).
- [ ] Deploy frontend to Vercel.
- [ ] Deploy backend to Render.
- [ ] Configure Supabase project + migrations.
