Proposition de Solution : Test Technique Développeur IA

E) Choix technologiques (Fondation)

Avant de rentrer dans les détails, voici les choix technologiques justifiés pour garantir performance, scalabilité et maintenabilité.

- Langage principal : Python 3.10+
  - *Justification :* L'écosystème le plus riche pour l'IA (PyTorch, TensorFlow, Scikit-learn, spaCy) et le traitement de données.
- Librairies IA & NLP :
  - spaCy : Pour le NLP "industriel" rapide (tokenisation, entity recognition) et l'extraction d'informations.
  - Scikit-Learn : Pour les calculs vectoriels (TF-IDF/Cosine Similarity) si besoin, et les métriques.
  - LangChain / LlamaIndex (Optionnel) : Si on utilise des LLM pour l'explication des scores.
- Type de modèle : Hybride
  - *Règles (Rule-based)* : Pour les filtres durs (Localisation, Disponibilité, Années d'expérience minimales).
  - *Vectoriel (Embeddings)* : Pour la compréhension sémantique des compétences (ex: "ML" proche de "Machine Learning").
  - *LLM (Large Language Model)* : Pour générer les explications naturelles du score.
- Base de données:
  - PostgreSQL : Stockage des données structurées (Users, Offres, Métriques).
  - Qdrant ou Milvus (Vector DB) : Stockage des embeddings des CV et Offres pour la recherche sémantique rapide.

A) Matching intelligent (Score de compatibilité)

1. Critères utilisés

- Compétences (Hard Skills) : Correspondance sémantique et exacte.
- Expérience : Comparaison entre les années requises et celles du candidat.
- Localisation : Correspondance géographique (ou télétravail).
- Disponibilité : Le candidat est-il "immédiat" ou " futur" ?

2. Méthode de calcul (Formule)

Le score est une moyenne pondérée sur une échelle de 0 à 100.

$Score = (Wskills \times Sskills) + (Wexp \times Sexp) + (Wloc \times Sloc) + (Wavail \times Savail)$

- *Sskills*
  = Ratio de compétences du candidat trouvées dans l'offre (Jaccard Index ou Overlap).
- *Sexp*
  = Score basé sur le ratio **Années Candidat / Années Requises** (plafonné à 1).
- *Sloc*
  = 1 si correspond, 0 sinon.
- ***Savail***
  = 1 si immédiat, 0.5 si sous 3 mois, 0 sinon.

3. Exemple chiffré

- Offre : Data Scientist (Python, ML, SQL). 5 ans d'exp. Paris.
- Candidat : Data Analyst (Python, SQL, Excel). 3 ans d'exp. Paris.
- Calcul :
  - Skills : 2/3 trouvés (Python, SQL) $\rightarrow 0.66$
  - Exp : 3/5 = 0.6 (Pénalité légère)
  - Loc : Paris = Paris $\rightarrow 1.0$
  - Score = $0.5(0.66)+0.3(0.6)+0.2(1.0)=0.33+0.18+0.2=0.71$

  * Score final : 71/100

4. Explication pour le recruteur (Générée par LLM ou Template)

"Le candidat correspond à 71% du profil. Il possède les compétences clés (Python, SQL) mais manque d'expérience sur le 'Machine Learning' (3 ans vs 5 ans requis). Il est disponible et basé à Paris."

B) Analyse automatique des CV

Pipeline de traitement

1. Ingestion (Upload) : Réception du fichier (PDF, DOCX).

2. OCR / Parsing : Conversion du document en texte brut. Utilisation de librairies comme **pdfplumber** ou **Tesseract** pour les scans.
3. Nettoyage (Cleaning) : Suppression des stop words, ponctuation, et normalisation des textes.
4. Extraction (NER) : Utilisation de modèles NLP (spaCy ou Transformers) pour extraire :
   - *Entities* : Noms, Emails, Téléphones.
   - *Skills* : Extraction basée sur une taxonomie (technique, soft skills).
   - *Experience* : Détection des patterns date (Regex) et calcul de la durée.
5. Structuration : Sauvegarde en JSON dans la base de données.

Gestion des CV mal formatés

- Validation de schéma : Si des champs critiques manquent (Email, Skills), on rejecte ou on flag en "Statut : À vérifier manuellement".
- Redondance : Utilisation de plusieurs parseurs (ex: un pour PDF, un pour DOCX) et agrégation des résultats.

C) Recommandation intelligente

Différence entre Matching et Recommandation

- **Matching** : Est statique et symétrique. C'est la mesure objective de l'écart entre deux profils à l'instant T (ex: "Est-ce que ce CV colle à cette offre ?").
- **Recommandation** : Est dynamique et contextuelle. C'est le classement (ranking) prenant en compte le comportement passée.
  - *Exemple :* Même si le score de matching est de 90%, si le recruteur *ne répond jamais* aux candidats de ce type, le système baissera son rang de recommandation.

Logique de recommandation

$$Score_{final} = (w1 \times Score_{matching}) + (w2 \times Score_{popularite\_offre}) + (w3 \times Score_{affinite\_recruteur})$$

Amélioration temporelle (Feedback Loop)

Système de Learning to Rank. Chaque action du recruteur (clic, masquage, entretien) est un signal positif ou négatif. Le système apprend à ajuster les poids ($w1, w2, w3$) pour maximiser la probabilité d'un "oui".