## 📝 ****1. Analyse des besoins****

### 🎯 Objectif général

Développer une **application web intelligente** destinée à **automatiser** et **optimiser** la gestion RH d'une grande entreprise (100 000 employés et plus).

### 📌 ****Acteurs principaux****

| **Acteur** | **Rôle** |
| --- | --- |
| 👤 **Candidat** | Consulte les offres et postule via le site ou e-mail |
| 🧑‍💼 **RH (Responsable Ressources Humaines)** | Gère les offres, sélectionne les candidats, gère salaires, absences, primes, droits |
| 👨‍🔧 **Employé interne** | Accède à son profil, ses points, son solde de congés, ses documents |

### ⚙️ ****Fonctionnalités principales****

#### 🔹 A. Partie externe : Gestion intelligente des candidatures

1. Publication des offres d’emploi sur le site web.
2. Candidature directe (formulaire ou e-mail automatique).
3. **Classification automatique** des CV via IA :
   * Détection de CV non pertinents → écartés.
   * Extraction des CV potentiels (ex. : 1 000 sur 10 000).
4. **Scoring IA** des meilleurs profils (ex. : top 50).
5. Suivi du processus : Entretien RH → Entretien Technique → Validation.

#### 🔹 B. Partie interne : Gestion RH

| **Module** | **Fonctionnalités** |
| --- | --- |
| 💰 **Salaires** | Calcul automatique, primes, absences, retards, bonus |
| 🏆 **Points employés** | Attribution de points selon performance, sanction, ponctualité |
| 📅 **Gestion des congés/droits** | Affichage solde, demande de congé, validation RH |
| 📩 **Mailing RH automatique** | Remerciements collectifs, annonces via SMTP |
| 🤖 **Chatbot RH** | Répond aux questions fréquentes, fournit documents (site + WhatsApp) |
| 🔒 **Accès employés** | Espace personnel sécurisé pour consulter données, points, documents |

## 🧱 ****2. Architecture du projet****

### 🧭 A. Architecture fonctionnelle (vue métier)

css

CopierModifier

[ Candidat ] → [ Portail emploi ] → [ Moteur IA de tri de CV ]

↓

[ Scoring / Classement des CV ]

↓

[ Tableau RH Sélection ]

css

CopierModifier

[ RH ] → [ Interface RH ]

→ Gérer les offres

→ Gérer les salariés

→ Valider primes / congés / absences

→ Consulter score IA des candidats

css

CopierModifier

[ Employé ] → [ Espace personnel ]

→ Voir : salaire, absences, points, congés

→ Faire demande de congé / attestation

→ Utiliser chatbot (Web ou WhatsApp)

### 🧰 B. Architecture technique (stack recommandée)

| **Couche** | **Technologies** |
| --- | --- |
| 🌐 **Frontend Web** | React.js / Vue.js (ou HTML/CSS/JS simple au début) |
| 🧠 **Backend API** | Flask / FastAPI (Python) ou Node.js |
| 💽 **Base de données** | PostgreSQL / MongoDB |
| 🧠 **Modèle IA** | TensorFlow / Keras / HuggingFace / Scikit-learn |
| 📁 **Stockage fichiers CV** | Google Cloud / S3 ou stockage local |
| 📬 **Mails automatiques** | SMTP (via smtplib ou SendGrid) |
| 🤖 **Chatbot Web** | Rasa, Botpress, ou OpenAI API |
| 🤖 **Chatbot WhatsApp** | Twilio WhatsApp API |
| 🔐 **Auth** | JWT ou OAuth2 |
| ☁️ **Déploiement** | Docker, Heroku, Render, ou VPS (DigitalOcean, AWS) |

## 🧠 ****3. Modèle IA proposé****

### 🔍 Étape 1 – Extraction et prétraitement

* Utilisation de PyMuPDF pour lire les CV PDF
* Nettoyage + normalisation NLP (spaCy, nltk)

### 🤖 Étape 2 – Matching CV <-> Offre

* Embedding avec SentenceTransformer (BERT)
* Similarité cosine avec l’offre (score entre 0 et 1)

### 🧠 Étape 3 – Scoring & Ranking

* Entraîner un modèle supervisé (optionnel) :
  + Input : vecteurs CV + Offre
  + Output : score de pertinence
  + Modèles : XGBoost, RandomForest, LogisticRegression

## ✨ ****4. Recommandations / Ajouts****

### ✅ Modules à ajouter pour renforcer :

* **Historique des candidatures** (tableau par date)
* **Historique de performance salarié** (graphique interactif)
* **Journal d’activité RH** (logs sécurisés)
* **Système de notification (mails / SMS)**

**🧱 2. Technologies recommandées pour ton projet RH**

**🎯 Objectif : Application web intelligente RH avec IA (CV Matching, salaires, chatbot)**

**🔧 Technologies front-end :**

| **Module** | **Outils recommandés** |
| --- | --- |
| UI | React.js ou Vue.js |
| Auth UI | Firebase Auth UI / JWT Auth |
| Tableaux / Stats | Chart.js, Recharts |
| Admin Dashboard | Tailwind UI / Bootstrap Admin Template |
| Formulaires upload CV | React Dropzone / HTML5 input[type=file] |

**🔧 Technologies backend :**

| **Module** | **Framework** |
| --- | --- |
| API Backend | **Flask** (léger, rapide, parfait pour IA) ✅ recommandé |
| ou bien | Django (plus lourd, très structuré, bon pour admin) |
| Base de données | PostgreSQL (recommandé), MongoDB (si données non relationnelles) |
| Auth | Flask-JWT / Flask-Login |

**🧠 Modèle IA (CV scoring) :**

| **Tâche** | **Outils** |
| --- | --- |
| Embedding | SentenceTransformer |
| Similarité | CosineSimilarity |
| Classification (top N) | XGBoost, RandomForest, LogisticRegression |
| OCR PDF / parsing | PyMuPDF, pdfminer.six |

**🤖 Chatbot RH :**

| **Plateforme** | **Technologie** |
| --- | --- |
| Web App Bot | Rasa, Botpress, OpenAI API |
| WhatsApp Bot | Twilio API (WhatsApp) + Flask webhook |

**📤 Mailing automatique :**

| Protocole | SMTP (SendGrid, Gmail SMTP avec Flask-Mail) |

**📁 3. Structure du projet recommandée (GitHub/GitLab)**

kotlin

CopierModifier

rh-app/

├── frontend/

│ ├── public/

│ └── src/

│ ├── components/

│ ├── screens/

│ └── App.js

│

├── backend/

│ ├── app.py

│ ├── routes/

│ ├── models/

│ ├── services/

│ │ ├── ai\_cv\_scoring.py

│ │ ├── pdf\_parser.py

│ │ └── email\_service.py

│ └── requirements.txt

│

├── chatbot/

│ ├── rasa/ (ou openai\_chatbot.py)

│ └── whatsapp\_bot/ (twilio)

│

├── uploads/ → fichiers PDF

├── data/ → scores, logs

├── .gitignore

├── README.md

└── docker-compose.yml (optionnel)