**Cahier des charges : Système de gestion de personnel avec React et Node.js**

**1. Introduction**

Ce document décrit les spécifications fonctionnelles et techniques d’un système de gestion de personnel destiné à une entreprise ou une organisation. L’objectif est de développer une application web moderne permettant de gérer efficacement les employés, leurs informations, leurs absences, leurs performances et d’autres aspects liés aux ressources humaines.

* **Technologies utilisées :**
  + **Frontend :** React.js (bibliothèque JavaScript pour les interfaces utilisateur)
  + **Backend :** Node.js avec Express.js (framework pour la création d’API RESTful)
  + **Base de données :** MySQL
  + **Authentification :** JSON Web Token (JWT)
* **Public cible :** Responsables RH, administrateurs, employés.
* **Objectif du projet :** Démontrer une maîtrise des technologies web modernes dans le cadre d’un projet de fin d’études (Licence 3).

**2. Fonctionnalités principales**

**2.1 Gestion des utilisateurs**

* **Inscription et authentification :**
  + Création de comptes pour les administrateurs et les employés (email, mot de passe).
  + Connexion sécurisée avec JWT.
  + Déconnexion.
  + Réinitialisation du mot de passe via email.
* **Rôles et permissions :**
  + Administrateur : accès complet au système.
  + Employé : accès limité à son profil et certaines fonctionnalités (ex. : demandes de congé).
* **Gestion des profils :**
  + Mise à jour des informations personnelles (nom, prénom, email, numéro de téléphone, photo de profil).
  + Visualisation des détails d’un utilisateur spécifique.

**2.2 Gestion des employés**

* **Ajout d’un employé :**
  + Formulaire pour ajouter un nouvel employé (nom, prénom, date de naissance, adresse, poste, département, salaire, date d’embauche).
* **Liste des employés :**
  + Affichage sous forme de tableau avec filtres (par département, poste, statut) et recherche par nom.
  + Pagination pour gérer un grand nombre d’employés.
* **Modification et suppression :**
  + Édition des informations d’un employé.
  + Suppression d’un employé (avec confirmation).
* **Historique :**
  + Suivi des modifications effectuées sur les profils (date, auteur de la modification).

**2.3 Gestion des absences et congés**

* **Demande de congé :**
  + Soumission d’une demande (type : congé payé, maladie, etc., dates de début et fin, motif).
  + Validation ou rejet par un administrateur.
* **Calendrier :**
  + Vue mensuelle/annuelle des absences (intégration d’une bibliothèque comme FullCalendar).
  + Affichage des congés approuvés et en attente.
* **Statistiques :**
  + Nombre de jours de congé restants par employé.
  + Rapport sur les absences par département.

**2.4 Gestion des performances**

* **Évaluation :**
  + Ajout d’évaluations périodiques (note, commentaire, date).
  + Historique des performances d’un employé.
* **Objectifs :**
  + Définition d’objectifs individuels (description, échéance, statut : en cours, terminé).
  + Suivi de l’avancement des objectifs.

**2.5 Gestion des salaires**

* **Fiches de paie :**
  + Génération automatique des fiches de paie (PDF téléchargeable) avec détails (salaire de base, primes, déductions).
* **Historique des paiements :**
  + Liste des salaires versés par mois et par employé.
* **Configuration :**
  + Ajout de primes ou déductions personnalisées.

**2.6 Tableau de bord (Dashboard)**

* **Vue d’ensemble :**
  + Nombre total d’employés, absences en cours, congés à valider.
  + Graphiques (ex. : répartition par département, taux d’absence) avec une bibliothèque comme Chart.js.
* **Notifications :**
  + Alertes pour les demandes de congé en attente ou les échéances d’évaluation.

**2.7 Paramètres et personnalisation**

* **Départements :**
  + Ajout, modification et suppression des départements.
* **Postes :**
  + Gestion des titres de poste (ex. : développeur, manager).
* **Configuration système :**
  + Réglages globaux (ex. : nombre de jours de congé annuel par défaut).

**2.8 Exportation et rapports**

* **Exportation :**
  + Export des données au format Excel ou CSV (liste des employés, absences, salaires).
* **Rapports :**
  + Génération de rapports personnalisés (ex. : taux d’absentéisme, performances globales).

**3. Architecture technique**

**3.1 Frontend (React)**

* **Structure des composants :**
  + Composants réutilisables (boutons, formulaires, tableaux).
  + Pages principales : Connexion, Dashboard, Liste des employés, Profil, Gestion des congés, etc.
* **Gestion d’état :**
  + Utilisation de Redux ou Context API pour gérer les données globales (ex. : utilisateur connecté).
* **Requêtes API :**
  + Utilisation d’Axios pour communiquer avec le backend.
* **Interface utilisateur :**
  + Design responsive avec CSS (ou Tailwind CSS/Bootstrap).
  + Animations légères pour une meilleure expérience utilisateur.

**3.2 Backend (Node.js avec Express)**

* **API RESTful :**
  + Endpoints pour chaque fonctionnalité (ex. : POST /api/employees, GET /api/absences).
* **Modèles de données :**
  + Employé : nom, email, poste, département, etc.
  + Congé : employé, type, dates, statut.
  + Évaluation : employé, note, commentaire, date.
* **Sécurité :**
  + Validation des entrées (avec Joi ou équivalent).
  + Hachage des mots de passe (avec bcrypt).
  + Protection des routes avec middleware JWT.

**3.3 Base de données**

* **Schéma suggéré (MongoDB) :**
  + Collection users : informations des utilisateurs.
  + Collection employees : données des employés.
  + Collection leaves : demandes de congé.
  + Collection evaluations : évaluations des performances.
* **Relations :**
  + Liaisons entre employés et absences/performances via des références.

**4. Contraintes**

* **Performance :** Temps de réponse des API < 2 secondes.
* **Sécurité :** Protection contre les attaques courantes (injections SQL, XSS).
* **Compatibilité :** Fonctionne sur les navigateurs modernes (Chrome, Firefox, Edge).
* **Délai :** Projet à finaliser avant la date de soutenance (à préciser).

**5. Livrables**

* Code source complet (frontend et backend) sur un dépôt Git (ex. GitHub).
* Documentation technique (installation, utilisation, structure du code).
* Base de données préremplie avec des données de test.
* Présentation PowerPoint pour la soutenance.

**6. Plan de développement**

1. **Analyse et conception :** 1 semaine (cahier des charges, maquettes).
2. **Mise en place du backend :** 2 semaines (API, base de données).
3. **Développement du frontend :** 3 semaines (interfaces, connexion API).
4. **Tests et débogage :** 1 semaine.
5. **Documentation et préparation de la soutenance :** 1 semaine.

Ce cahier des charges est exhaustif et couvre un large éventail de fonctionnalités pour impressionner lors de votre soutenance. Vous pouvez l’adapter en fonction de vos compétences ou du temps disponible. Si vous souhaitez des précisions ou une aide sur une partie spécifique (ex. : code, maquettes), n’hésitez pas à me le demander ! Bonne chance pour votre projet !