CAHIER DE CHARGE (shadow’s coders)

Gestion des emplois du temps et la disponibilité des salles

1. Introduction

L'application web permettra de gérer les emplois du temps des entités et des filières de L’université d’Abomey Calavi (UAC). Elle permettra de programmer des cours pour chaque promotion d'étudiants et d'envoyer les emplois du temps par mail. En parallèle, un système de gestion de la disponibilité des salles sera mis en place.

2. Objectifs

- **Gestion des emplois du temps** : Chaque entité (faculté, département, etc.) pourra gérer ses propres emplois du temps, définir les horaires des cours pour chaque promotion.
- **Communication avec les étudiants** : Les étudiants recevront leurs emplois du temps par mail.
- **Gestion des salles** : Les salles seront attribuées en fonction de la disponibilité et des cours programmés.

3. Technologies

- **Frontend** : Django (framework Python) pour l'interface utilisateur.
- **Backend** : Node.js avec Express pour la gestion des requêtes API.
- **Base de données** : PostgreSQL ou MySQL (selon les besoins), mais peut également être envisagé d'utiliser une base relationnelle simple comme SQLite dans un premier temps.
- **Envoi des emails** : Nodemailer (Node.js) pour envoyer les emplois du temps par mail aux étudiants.
- **Gestion des utilisateurs** : Utilisation de JWT pour l'authentification des utilisateurs (administrateurs, professeurs, étudiants).

4. Fonctionnalités principales

4.1. Gestion des utilisateurs

- **Types d'utilisateurs** :
    - **Administrateurs** : Gèrent l'ensemble de l'application (ajout d'entités, gestion des utilisateurs, gestion des salles, etc.).

      Créent et modifient les emplois du temps de leurs cours, définissent la salle et les horaires.
    - **Étudiants** : Consultent leur emploi du temps personnel, reçoivent des notifications par mail.
- **Fonctionnalité d'authentification** : Inscription, connexion et déconnexion avec gestion des rôles.

4.2. Gestion des entités et des filières

- **Création d'entités** : Ajout d'entités (faculté, département, etc.).
- **Création de filières** : Ajout et gestion des filières et des promotions d'étudiants.
- **Gestion des promotions** : Chaque promotion peut avoir un emploi du temps propre.

4.3. Gestion des emplois du temps

- **Programmation des cours** : Les professeurs peuvent créer, modifier et supprimer les cours pour chaque promotion.
- **Attribution des salles** : Les administrateurs attribuent les salles en fonction de la disponibilité.
- **Consultation des emplois du temps** : Les étudiants peuvent consulter leur emploi du temps en ligne et le recevoir par mail.

4.4. Gestion des salles

- **Création et gestion des salles** : Les administrateurs ajoutent les salles disponibles dans le système.
- **Disponibilité des salles** : Les salles sont marquées comme disponibles ou non en fonction des cours programmés.
- **Conflits de réservation** : Le système vérifie les conflits d'horaires pour éviter la réservation de la même salle par deux cours différents.

4.5. Envoi d'emails

- **Envoi automatique des emplois du temps** : Les étudiants reçoivent leur emploi du temps par email chaque semaine (ou à la demande).
- **Personnalisation des emails** : Le contenu des emails sera personnalisé pour chaque étudiant avec son emploi du temps spécifique.

5. Architecture technique

- **Frontend (Django)** : Utilisation de Django pour gérer les pages de l'application, le rendu des emplois du temps, les formulaires de saisie, etc.
    - Django ORM pour la gestion de la base de données.
    - Authentification et autorisation via Django.
    - Templates et views pour l'affichage des données.
- **Backend (Node.js)** : Gestion des API REST avec Express pour interagir avec le frontend et la base de données.
    - Création d'API pour récupérer les emplois du temps, les informations sur les utilisateurs, les salles, etc.
    - Gestion de la logique métier (scheduling, disponibilité des salles, envoi des emails).
    - Base de données relationnelle pour stocker les informations liées aux emplois du temps, aux utilisateurs, aux salles, etc.

6. Sécurité

- **Authentification JWT** : Sécurisation de l'accès avec des tokens JWT pour les API.
- **Protection des données sensibles** : Utilisation du chiffrement pour les mots de passe et autres données sensibles.
- **Contrôle d'accès** : Mise en place de permissions basées sur les rôles (administrateurs, professeurs, étudiants).

7. Déploiement

- **Serveurs** : Hébergement sur un serveur cloud (comme AWS, Heroku, DigitalOcean) pour l'application backend et frontend.
- **Base de données** : PostgreSQL ou MySQL sur un serveur cloud.

- **Gestion des emails** : Utilisation de services d'envoi d'emails comme SendGrid ou Mailgun.

8. Maintenance et support

- **Mises à jour** : Un plan de mise à jour régulier du système.
- **Support** : Un système de support technique pour les utilisateurs de l'application (administrateurs, professeurs, étudiants).

9. Planning

- **Phase 1** : Conception de la base de données, définition des modèles et des relations.
- **Phase 2** : Développement de l’API backend avec Node.js et Express.
- **Phase 3** : Développement du frontend avec Django et intégration des API.
- **Phase 4** : Tests, optimisation de la performance, déploiement.
- **Phase 5** : Maintenance continue.

10. Conclusion

Ce projet vise à automatiser et simplifier la gestion des emplois du temps et des ressources dans les universités, tout en offrant une expérience utilisateur fluide et accessible.