ECF2

Contexte de l'ECF :

Vous travaillez pour une startup appelée **"FoodieShare"**, qui souhaite créer une plateforme de partage de recettes en ligne. Les utilisateurs peuvent soumettre leurs recettes, les consulter, les commenter, et les classer. Le projet inclut à la fois le développement d'une interface utilisateur (front-end) avec React, et d'une partie serveur (back-end) avec Node.js, en se connectant à une base de données MongoDB pour gérer les recettes et les utilisateurs.

Objectif de l'ECF :

Cette ECF a pour objectif de vous évaluer sur :

Le **développement d'une interface utilisateur (front-end)** avec React, HTML, CSS et JavaScript.
Le **développement de la partie serveur (back-end)** avec Node.js.
L'**utilisation d'APIs** pour les interactions entre le front-end et le back-end.
L'**intégration d'une base de données** (MongoDB) pour stocker et gérer les données.

Étapes de l'ECF :

1. Développement de l'interface utilisateur (front-end)

Tâche : Créer une interface utilisateur simple permettant aux utilisateurs de soumettre, consulter, et commenter des recettes.
Livrables attendus :

Une **page d'accueil** avec un affichage des recettes les plus populaires.
Un **formulaire de soumission** pour permettre aux utilisateurs d'ajouter de nouvelles recettes (avec champs pour le titre, description, ingrédients, et étapes).
Une **page de consultation de recette** avec les détails de la recette et une section pour les commentaires.

Technologies à utiliser :

React pour construire des composants dynamiques et réactifs.
HTML/CSS pour la structure et la mise en page.
JavaScript pour la logique du front-end et les appels API.

Critères d'évaluation :

Respect de la structure **responsive** (adaptation mobile).

Utilisation des composants React pour modulariser l'interface.
Gestion des états (formulaire de soumission, affichage des recettes) avec **React Hooks**.

2. Développement de la partie serveur (back-end)

Tâche : Créer une API REST avec Node.js qui permettra de gérer les recettes (ajout, affichage, modification, suppression).

Livrables attendus :

Un **serveur Node.js** avec des routes pour gérer les requêtes HTTP (GET, POST, PUT, DELETE).
Une **route POST** pour ajouter une recette.
Une **route GET** pour récupérer toutes les recettes disponibles.
Une **route GET** pour récupérer une recette spécifique en fonction de son ID.
Une **route DELETE** pour supprimer une recette.

Technologies à utiliser :

Node.js et **Express** pour créer les routes et gérer les requêtes HTTP.
MongoDB pour stocker les recettes et les utilisateurs.

Critères d'évaluation :

Respect des bonnes pratiques de développement en **Node.js** (gestion des erreurs, validation des données).
Création d'une API REST **modulaire** et bien structurée.
Utilisation correcte des **middlewares** (par exemple, pour l'authentification).

3. Utilisation d'APIs et interaction front-end/back-end

Tâche : Intégrer le front-end et le back-end en faisant des appels API pour interagir avec les données de l'application.

Livrables attendus :

Appels API **asynchrones** (avec **fetch** ou **Axios**) pour récupérer les données depuis le serveur et les afficher dans l'interface utilisateur.
Soumission de formulaire (ajout d'une recette) qui envoie les données au serveur via une requête **POST** et affiche un message de succès ou d'échec.
Consultation des recettes depuis la base de données via un appel API **GET**.

Technologies à utiliser :

Axios ou **fetch** pour les appels API.
React pour la gestion des états et des données récupérées depuis l'API.

Critères d'évaluation :

Gestion des **promesses** et des erreurs lors des appels API.
Synchronisation correcte des données entre le front-end et le back-end.
Respect des **principes REST** (utilisation des bonnes méthodes HTTP, etc.).

4. Gestion de la base de données (MongoDB)

Tâche : Configurer une base de données MongoDB pour stocker les informations sur les recettes et les utilisateurs.

Livrables attendus :

Un **schéma MongoDB** pour les recettes, avec des champs pour le titre, description, ingrédients, étapes, et les commentaires des utilisateurs.
Intégration de **Mongoose** pour la gestion de la base de données dans l'application Node.js.
Méthodes pour **créer, lire, modifier et supprimer** des recettes dans la base de données.

Exemple de schéma de recette :
javascript

```
const recipeSchema = new mongoose.Schema({
    title: { type: String, required: true },
    description: String,
    ingredients: [String],
    steps: [String],
    comments: [{
        user: String,
        message: String,
        date: { type: Date, default: Date.now }
    }]
});
```

Eventuellement faire quelques tests avec Jest.