Étape 1 : Préparer ton environnement

Créer le projet React :	npx create-react-app immobilier cd immobilier npm install react-router-dom json-server zustand
Créer le fichier `db.json` à la racine du projet Va dans `F:\1Work\4eme\S2\React\immobilier`	{ "properties": [{ "id": 1, "title": "Maison de campagne", "price": 250, "available": true, "views": 0, "evaluations": [4, 5] }, { "id": 2, "title": "Appartement S+2", "price": 400, "available": true, "views": 0, "evaluations": [] }, { "id": 3, "title": "Villa avec piscine", "price": 1000, "available": false, "views": 12, "evaluations": [5, 5] }] }
`Lance le serveur JSON :`	npx json-server --watch db.json --port 3001
Modifier App.js pour mettre en place le Routing Exemple de code :	// src/App.js import React from "react"; import { BrowserRouter as Router, Routes, Route } from "react-router-dom"; import Properties from "./components/Properties"; // Pour la liste des propriétés import ReservationForm from "./components/ReservationForm"; // Pour le formulaire de réservation function App() { return (<Router> Agence Immobilière <Routes> <Route path="/" element={<Properties />} />

<table>
<tr><td></td><td>

```
        <Route path="/reserve"
element={<ReservationForm />} />
      </Routes>
    </div>
   </Router>
  );
}

export default App;
```

</td></tr>
<tr><td>crée dans le dossier src/components
les trois fichiers suivants :
• Properties.jsx
• Property.jsx
• ReservationForm.jsx</td><td></td></tr>
<tr><td>Question 1</td><td>

```jsx
// src/components/Properties.jsx
import React, { useEffect, useState } from "react";
import Property from "./Property";

function Properties() {
  const [properties, setProperties] = useState([]);
  const [min, setMin] = useState("");
  const [max, setMax] = useState("");
  useEffect(() => {
    fetch("http://localhost:3001/properties")
      .then((res) => res.json())
      .then((data) => setProperties(data));
  }, []);
  // Filtrer les propriétés par prix
  const filteredProperties = properties.filter((p) => {
    const price = p.price;
    return (
      (!min || price >= parseInt(min)) &&
      (!max || price <= parseInt(max))
    );
  });
  return (
    <div>
      <h2>Liste des Propriétés</h2>
      {/* Formulaire de recherche */}
      <div>
        <input
          type="number"
          placeholder="Prix min"
          value={min}
          onChange={(e) => setMin(e.target.value)}
        />
        <input
          type="number"
          placeholder="Prix max"
          value={max}
          onChange={(e) => setMax(e.target.value)}
        />
      </div>
      {filteredProperties.map((property) => (
        <Property key={property.id} property={property} />
      ))}
    </div>
  );
}
export default Properties;
```

</td></tr>
<tr><td>Property.jsx</td><td>

```jsx
// src/components/Property.jsx
import React from "react";

function Property({ property }) {
  return (
    <div style={{ border: "1px solid black", margin: "10px", padding: "10px" }}>
      <h4>{property.title}</h4>
      <p>Prix : {property.price} DT</p>
      <p>Disponible : {property.available ? "Oui" : "Non"}</p>
      <p>Vues : {property.views}</p>
    </div>
  );
}

export default Property; // ☑ TRÈS IMPORTANT
```

</td></tr>
</table>

Lance json-server en parallele	npx json-server --watch db.json --port 3001
Va sur http://localhost:3000	
Question 3 Avant return : Dans return && div :	``` const navigate = useNavigate(); const handleReserve = () => { fetch(`http://localhost:3001/properties/${property.id}`, { method: "PATCH", headers: { "Content-Type": "application/json" }, body: JSON.stringify({ views: property.views + 1 }) }).then(() => { navigate("/reserve", { state: { property } }); }); }; <button onClick={handleReserve}>Réserver la propriété</button>)} ```
Question 4	``` const handleReserve = () => { // Étape 1: incrémenter les vues côté API fetch(`http://localhost:3001/properties/${property.id}`, { method: "PATCH", headers: { "Content-Type": "application/json", }, body: JSON.stringify({ views: property.views + 1, }), }).then(() => { // Étape 2: rediriger vers /reserve en transmettant la propriété navigate("/reserve", { state: { property: { ...property, views: property.views + 1 } }, }); }); }; ```
Question 5	
Le formulaire sera initialisé Dans le return, tu dois avoir : • 1 champ pour le nom complet • 2 champs date : début et fin • 1 bouton Valider	``` return (Réservation de : {property.title} {message ? ({message}) : (<form onSubmit={handleSubmit}> <input type="text" placeholder="Nom complet" value={name} onChange={(e) => setName(e.target.value)} required /> <input type="date" value={startDate} onChange={(e) => setStartDate(e.target.value)} required /> <input type="date" value={endDate} onChange={(e) => setEndDate(e.target.value)} required /> <button type="submit">Valider</button> </form>)}); } ```
Math.ceil((new Date(date de fin de la location) - new Date(date début de la location)) / (1000 * 60 * 60 * 24))	``` const duration = Math.ceil((new Date(endDate) - new Date(startDate)) / (1000 * 60 * 60 * 24)); ```
Affiche un message comme : 👉 Mr/Mme [Nom], votre réservation pour [Titre] est confirmée pour un prix de [X] DT.	``` // Afficher le message setMessage(`Mr/Mme ${name}, votre réservation pour ${property.title} est confirmée pour un prix de ${total} DT.`); ```
Après 5 secondes	``` // Rediriger vers / après 5 secondes setTimeout(() => { navigate("/"); }, 5000); ```

	`` }; ```
👉 Va dans le dossier src/components 📄 Crée un fichier nommé ReservationForm.jsx	```jsx // src/components/ReservationForm.jsx import React, { useState } from "react"; import { useLocation, useNavigate } from "react-router-dom"; function ReservationForm() { const location = useLocation(); const navigate = useNavigate(); const { property } = location.state; const [name, setName] = useState(""); const [startDate, setStartDate] = useState(""); const [endDate, setEndDate] = useState(""); const [message, setMessage] = useState(""); const handleSubmit = (e) => { e.preventDefault(); const duration = Math.ceil((new Date(endDate) - new Date(startDate)) / (1000 * 60 * 60 * 24)); const total = duration * property.price; // Marquer la propriété comme non disponible fetch(`http://localhost:3001/properties/${property.id}`, { method: "PATCH", headers: { "Content-Type": "application/json", }, body: JSON.stringify({ available: false }), }); // Afficher le message setMessage(`Mr/Mme ${name}, votre réservation pour ${property.title} est confirmée pour un prix de ${total} DT.`); // Rediriger vers / après 5 secondes setTimeout(() => { navigate("/"); }, 5000); }; return (Réservation de : {property.title} {message ? ({message}) : (<form onSubmit={handleSubmit}> <input type="text" placeholder="Nom complet" value={name} onChange={(e) => setName(e.target.value)} required /> <input type="date" value={startDate} onChange={(e) => setStartDate(e.target.value)} required /> <input type="date" value={endDate} onChange={(e) => setEndDate(e.target.value)} required /> <button type="submit">Valider</button> </form>)}); } export default ReservationForm; ```
PARTIE B — Gestion d’évaluations avec Zustand	``` const [showEval, setShowEval] = useState(false); ```
	`` <button onClick={() => setShowEval(!showEval)}> Ajouter une évaluation </button> ```

Ajouter une évaluation Dans le composant `Property.jsx`, tu dois :	
une liste déroulante avec des valeurs de 1 à 5 s'affiche. en dessous du bouton "Ajouter une évaluation", ajoute :	`{showEval && (` `<div>` `<select value={note} onChange={(e) => setNote(Number(e.target.value))}>` `{[1, 2, 3, 4, 5].map((n) => (` `<option key={n} value={n}>{n}</option>` `))}` `</select>` `<button onClick={handleEval}>Valider</button>` `</div>` `)}`
Et plus haut dans le composant	`const [note, setNote] = useState(1);` `const handleEval = () => {` `console.log("Note sélectionnée :", note);` `};`
a. L'évaluation est **affichée dans la console** b. La moyenne des évaluations pour cette propriété est **calculée** c. Si la moyenne dépasse **4.5**, une alerte s'affiche : 📁 Va dans `src/store` 📄 Crée un fichier : `useEvaluationStore.js`	`// src/store/useEvaluationStore.js` `import { create } from "zustand";` `const useEvaluationStore = create((set) => ({` `evaluations: {},` `addEvaluation: (propertyId, note) =>` `set((state) => {` `const prev = state.evaluations[propertyId] || [];` `const updated = [...prev, note];` `const average =` `updated.reduce((a, b) => a + b, 0) / updated.length;` `console.log(`Évaluations pour ${propertyId}:`, updated);` `console.log(`Moyenne:`, average);` `if (average > 4.5) {` `alert("Propriété excellente !");` `}` `return {` `evaluations: {` `...state.evaluations,` `[propertyId]: updated,` `},` `};` `}),` `}));` `export default useEvaluationStore;`
Tout en haut de Property.jsx	`import useEvaluationStore from "../store/useEvaluationStore";`
Et remplace ta fonction `handleEval` par	`const { addEvaluation } = useEvaluationStore();` `const handleEval = () => {` `addEvaluation(property.id, note);` `};`
address already in use :::3001	netstat -ano | findstr :3001 taskkill /PID 21576 /F