EXAMEN DE FIN DE MODULE RÉGIONAL
AU TITRE DE L'ANNÉE : 2025/2026
DIRECTION RÉGIONALE FÈS-MEKNÈS

Année de formation : 2A
Niveau de formation : TS
Filière : Développement Digital Option Web Full-Stack
Intitulé du module : M204_Développer en front-end

Épreuve : ☐ Théorique ☐ Pratique ☑ Synthèse
Durée : 02h30 *Barème : / 40 pts*

VARIANTE : 1

PARTIE THÉORIQUE : _____ / 10 pts

Q1. Écrire les commandes permettant de créer un projet react. **(2pts)**

Q2. Écrire les commandes permettant d'installer les bibliothèques redux, axios et react-router. **(3pts)**

Q3. Traduire la fonction suivante en une fonction fléchée et remplacer **if** par l'opérateur ternaire (- -? --: --) **(3 pts)**

```
function calcul(n) {
    if (n%2==0)
        return "pair"
    else
        return "impair"
}
```

Q4. Qu'est-ce qu'un thunk dans le contexte de Redux ? **(1pt)**

a) Une fonction qui retourne un objet action simple.
b) Une fonction qui peut contenir une logique asynchrone (comme un appel API).
c) Un état initial d'un slice.
d) Un composant React.

Q5. Quelle fonction de Redux Toolkit permet de créer facilement un thunk pour des opérations asynchrones ? **(1pt)**

a) configureStore
b) createSlice
c) createAsyncThunk
d) Provider

PARTIE PRATIQUE : _____ ***/ 30 pts***

On souhaite développer une plateforme **HAJZ.ma** de réservation des hôtels, des vols et des voitures de location au Maroc.

Votre 1ère mission est de créer des pages **admin** permettant de gérer les voitures de location comme indiqué dans l'**illustration 1**.

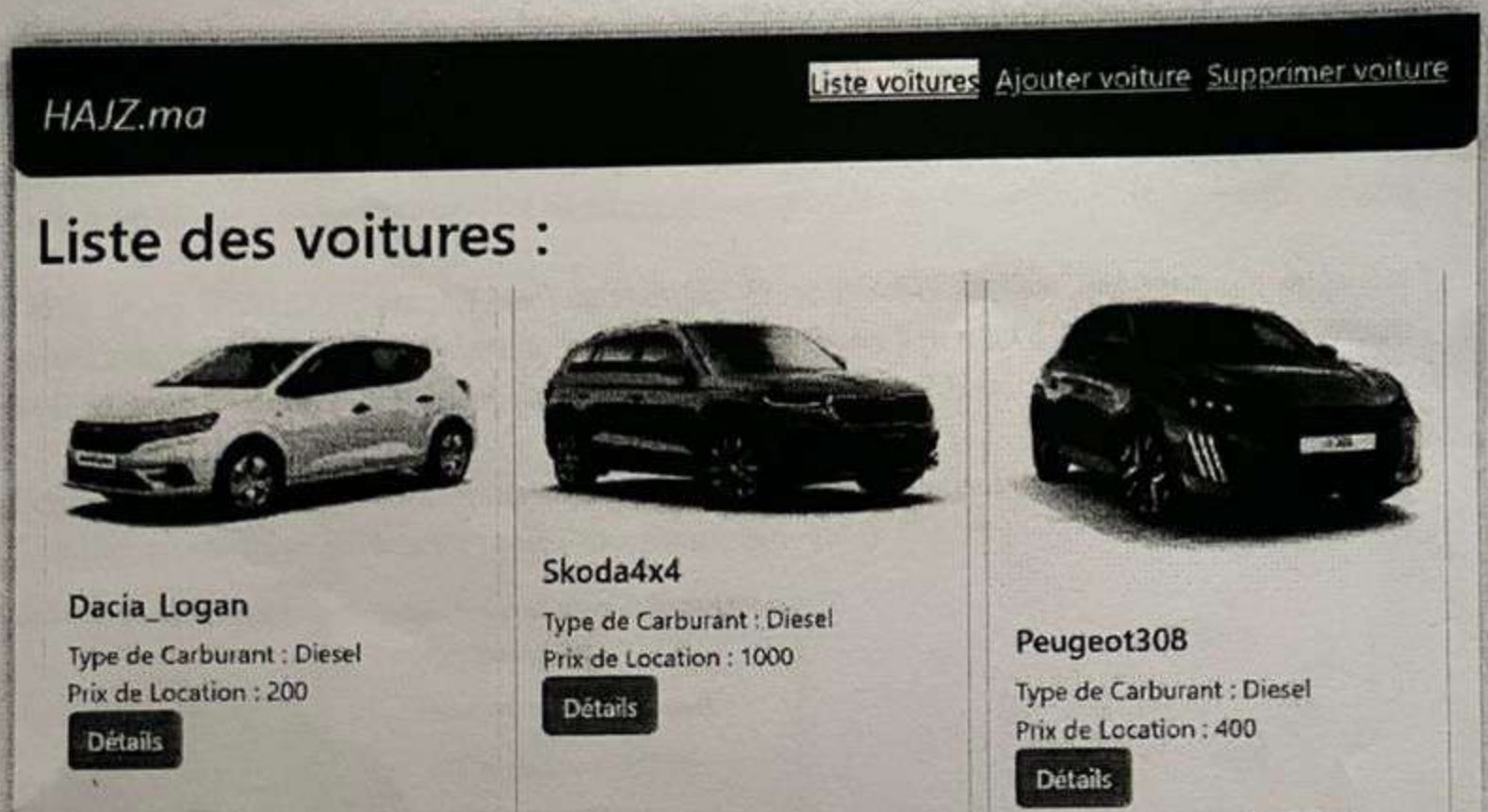


Illustration 1

Soit le state suivant :

```
const initState = {
"voitures": [
    {"id":"v1","Marque":"Dacia_Logan","TypeCarburant":"Diesel","PrixLocation":200,"
image":"dacia1.jpg"}, ...
  ]}
```

1. Écrire le code du fichier **voitureSlice.js** permettant de créer un slice avec le state initial décrit ci-dessus, en y ajoutant les trois **reducers** (actions) suivantes : **(1pt)**
   a. **addVoiture** : permet l'ajout d'une nouvelle voiture à la liste des voitures du store. **(1pt)**
   b. **updateVoiture** : permet de modifier les information d'une voiture (**l'id** de la voiture ne peut pas être modifier). **(2pts)**
   c. **deleteVoiture** : permettant de supprimer une voiture du store. **(2pts)**
2. Écrire le code du fichier **VoitureStore.js** permettant de créer ou de configurer le store redux avec le slice créé précédemment dans le fichier **voitureSlice.js. (1pt)**

3. Réécrire le code du fichier **index.js** afin de prendre en considération le store de la question précédente. **(1pt)**

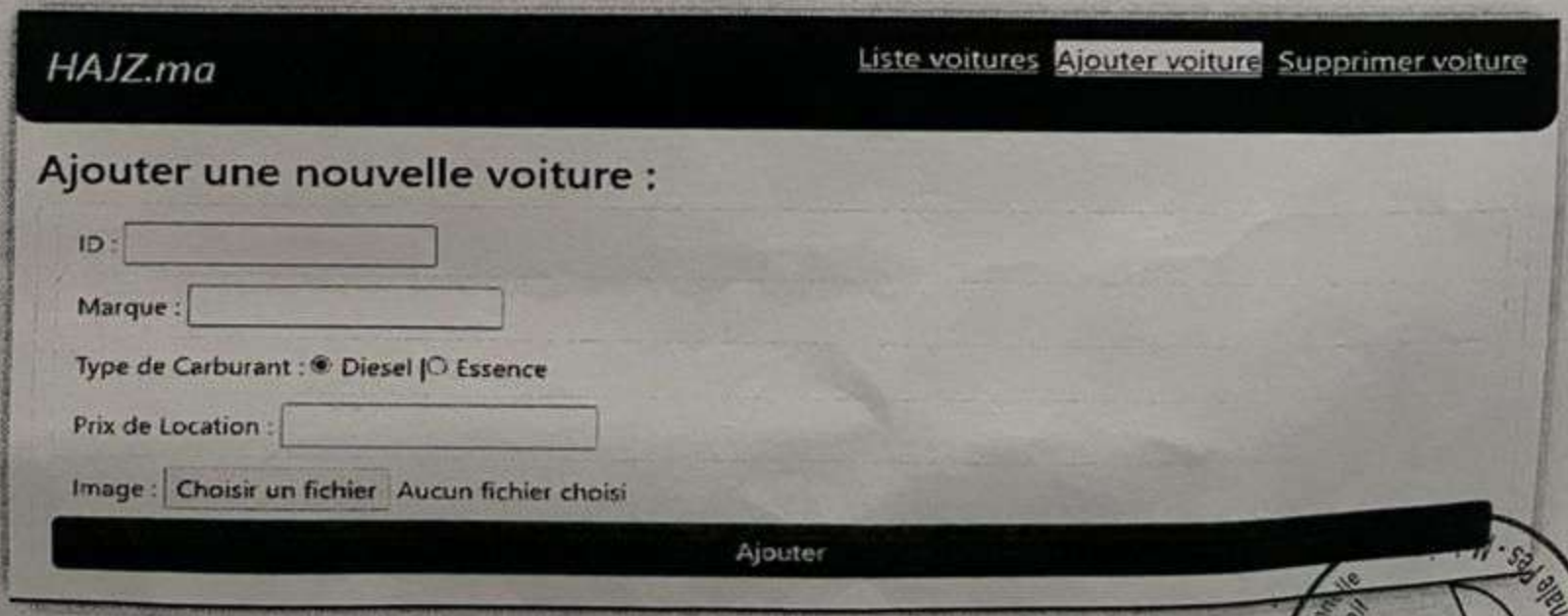
```
import ReactDOM from 'react-dom/client';
import App from './efm/app';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <App />
);
```

4. Écrire le code du composant **Voiture.jsx** pour afficher les informations d'une voiture dans une carte bootstrap (voir illustration 1). Les données sont passées au composant via les **props**. **(3pts)**

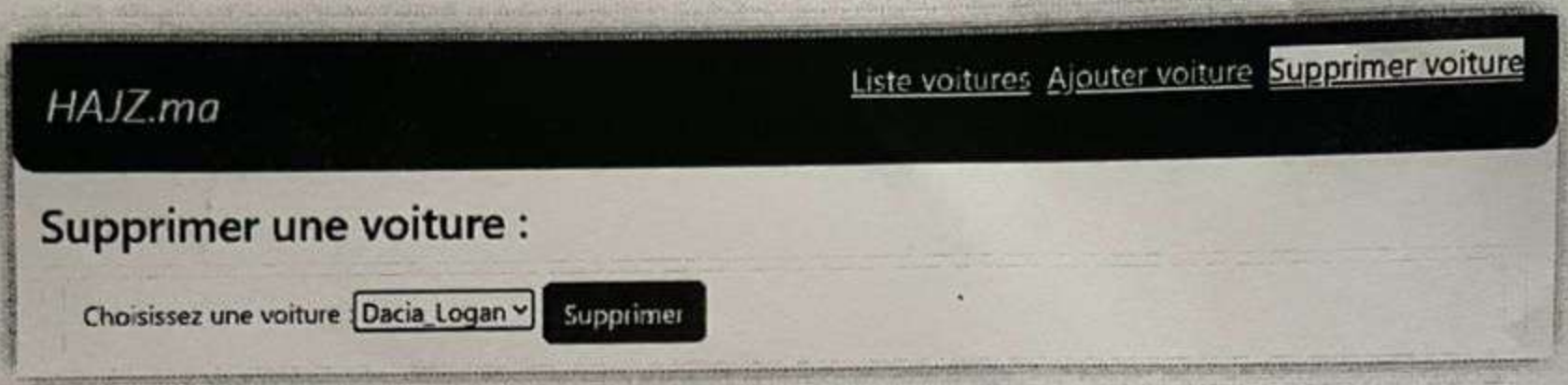
5. Écrire le code du composant **ListeVoitures.jsx** pour afficher la liste des voitures du store, conformément à illustration 1. Le clic sur le bouton « Détails » permet d'accéder à la page détails de la voiture, en passant son identifiant (id) comme paramètre. **(3pts)**

6. Écrire le code du composant **DetailsVoitures.js** pour afficher toutes les informations de la voiture dont l'**id** est passé en paramètre (voir l'illustration suivante). **(3pts)**

7. Écrire le code du composant **AjouterVoiture.js** pour afficher un formulaire permettant d'ajouter une nouvelle voiture à la liste des voitures du store (voir l'illustration suivante). **(3pts)**



8. Écrire le code du composant **SupprimerVoiture.js** pour afficher une liste déroulante (automatique) permettant de choisir la voiture à supprimer. Le clic sur le bouton **Supprimer** supprime la voiture sélectionnée après confirmation (voir l'illustration suivante). **(3pts)**



9. Écrire le code du composant **App.jsx** pour afficher les menus de navigation comme indiqué dans l'illustration 1 et définir les routes selon le tableau ci-dessous. **(4pts)**

Route	Composant
http://localhost:3000/	**ListeVoitures**
http://localhost:3000/v1	**DetailsVoitures** avec v1 : Id passé en paramètre
http://localhost:3000/add	**AjouterVoiture**
http://localhost:3000/delete	**SupprimerVoiture**

10. Écrire le code du composant **VoitureAPI.jsx** permettant d'afficher, sous forme de tableau, les données récupérées depuis l'API http://localhost:8000/api/voitures. Sans utiliser redux. **(3pts)**