OFPPT	مكتب التكوين المهني وإنعاش الشغل
	Office de la Formation Professionnelle et de la Promotion du Travail
	Direction de la Recherche et de l'Ingénierie de la Formation Division Conception des Examens

Examen National de Fin d'année

Session de Juin 2025

Examen de Fin de Formation (Epreuve de Synthèse)

Eléments de correction

Secteur :	Digital et Intelligence Artificielle			Niveau :	Technicien Spécialisé
Filière :	Développement Digital Option Web Full Stack				
Variante	2	Durée :	4h00	Barème	/100

Consignes et Précisions aux correcteurs :

Veuillez respecter impérativement les consignes suivantes :

- **Les éléments de correction sont élaborés à titre indicatif,**
- **Eviter de sanctionner doublement le stagiaire sur les questions liées,**
- **Pour toutes les questions de synthèse et de compréhension le correcteur s'attachera à évaluer la crédibilité et la pertinence de la réponse du stagiaire. Et à apprécier toute réponse cohérente du stagiaire,**
- **Le stagiaire n'est pas tenu de fournir des réponses aussi détaillées que celles mentionnées dans le corrigé,**
- **En cas de suspicion d'erreur au niveau du corrigé, prière de contacter la Division de Conception des Examens.**
-

Détail du Barème :

N° Des Dossiers	Travaux à réaliser	Barème
Partie Théorique		
Dossier 1	Création d'une Application Cloud native	8 pts
Dossier 2	Préparation d'un projet web	6 pts
Dossier 3	Approche Agile	07 pts
Dossier 4	Gestion de données NOSQL	11 pts
Dossier 5	Web Dynamique PHP	08 pts
	Total Partie Théorique	/40 points
Partie Pratique		
Dossier 1	Gestion des données MySQL	12 pts
Dossier 2	Développement Front End	24 pts
Dossier 3	Développement Back End	24 pts
	Total Partie Pratique	/60 points
	Total Général	/100 points

Partie Théorique (40 pts)

Dossier 1 Création d'une Application Cloud native (8 pts):

1. A quoi sert l'outil Postman ? (2pts)

Tester des API, simuler des API, Automatiser des testes

2. Comment télécharger une image Docker depuis Docker Hub ? (2pts)

docker pull <nom_de_l'image>:<tag>

3. Comment arrêter un conteneur Docker en cours d'exécution ? (2pts)

docker stop <id_ou_nom_du_conteneur>

4. Quels sont les services cloud (as a service) offerts par Azure ? (2pts)

IaaS (Infrastructure as a Service) , PaaS (Platform as a Service) SaaS (Software as a Service)

Préliminaire :

La société « **FastFoodDelivery** » fournit à ses employés des trottinette électriques pour assurer la livraison des colis auprès de ses clients. Pour garantir la continuité et la qualité de ses services, elle a décidé de mettre en place un système de suivi de l'état des batteries de ses trottinettes. Ce système permettra d'anticiper le remplacement des batteries avant qu'elles ne se détériorent ou tombent en panne.

Trottinette(id, numero_serie, #id_batterie, date_derniere_maintenance, date_prochaine_maintenance)
Batterie(id, Numero_serie, capacite, sante_batterie, nombre_cycles, statut)
Remplacement(id, #id_trottinette, #id_ancienne_batterie, #id_nouvelle_batterie, date_remplacement, #id_reparateur,raison)
Reparateur(id, nom, telephone, email, specialite)

NB :La clé primaire est écrite en souligné, la clé étrangère est écrite en #

Dossier 2 : Préparation d'un projet web (6 pts) :

Donner le diagramme de cas d'utilisation correspondant au système suivant :(6 pts)

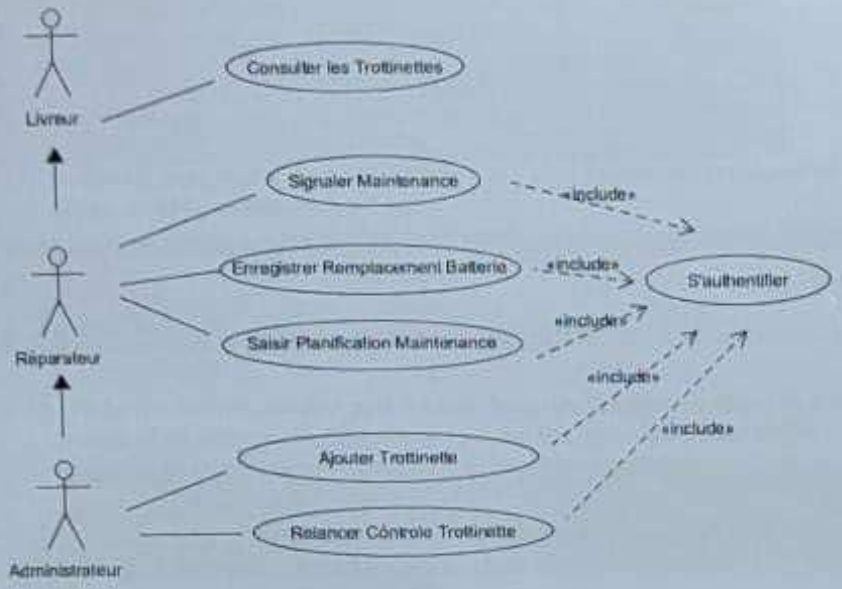
Le système sera utilisé par trois utilisateurs : **Administrateur**, **Réparateur et Livreur**

Le réparateur peut **signaler** qu'une trottinette a été maintenue, **enregistrer** les remplacements des batteries, **saisir** une planification de la prochaine maintenance et **consulter** la liste des trottinettes

L'administrateur en plus des fonctionnalités du réparateur peut gérer les trottinettes en les **ajoutant** au système, **consulter** leur liste et **relancer** le contrôle des trottinettes s'ils ne sont pas faits à temps

L'accès au système est sécurisé, et chaque utilisateur doit s'authentifier pour accéder à ses fonctionnalités respectives.

Le livreur à un accès libre en **consultation** des trottinettes

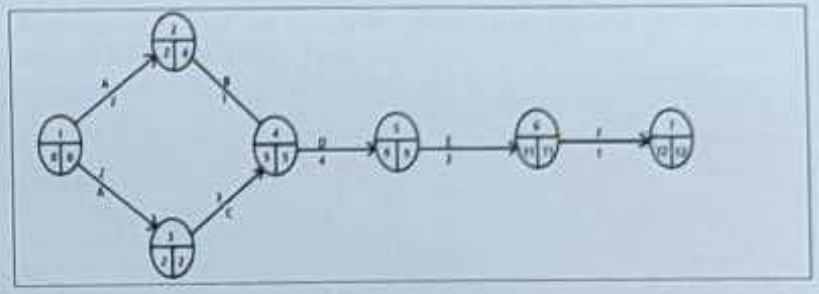


Dossier 3 : Approche Agile (07 pts)

Soit la liste des tâches suivantes :

Tache	Durée En jours	Tâches Antérieurs
A	2	-
B	1	A
C	3	A
D	4	B, C
E	2	D
F	1	E

1- Dresser le diagramme PERT(3pts)



2- Trouver le Chemin critique(1pts)

A → C → D → E → F

3- Déterminer la durée d'exécution du projet (1pts)

12 jours

4- La tache B a pris 4 jours au lieu de 1 jour prévu : (2pts)

a. Déterminer le nouveau chemin critique
b. Déterminer la nouvelle durée du projet

a- A → B → D → E → F
b- 13 jours

Dossier 4 : Gestion de données NOSQL (11 pts)

1) Créer une base de données appelée **FastFoodDelivery** et une collection nommée **batteries** (1pt)

```
use FastFoodDelivery
db.createCollection("batteries")
```

2) Insérer les documents suivants dans la collection batteries : (2pts)

```
[
 {"_id":"1", "numéro_série":"s123", "capacité":"50%",
  "santé_batterie":"bonne", "nombre_cycle":500,"statut":"en service"},

 {"_id":"2", "numéro_série":"s345", "capacité":"70%",
  "santé_batterie":"moyenne", "nombre_cycle":600, "statut":"en panne"}
]
```

```
db.batteries.insertMany
([
{ "_id": "1", "numéro_série": "s123", "capacité": "50%", "santé_batterie":
"bonne", "nombre_cycles": 500, "statut": "en service" },

{ "_id": "2", "numéro_série": "s345", "capacité": "70%", "santé_batterie":
"moyenne", "nombre_cycles": 600, "statut": "en panne" }
]);
```

3) Modifier la capacité et le statut de la batterie dont l'id est 2 par les valeurs respectives suivantes : 45%, « Entretenue » (2 pts)

```
db.batteries.updateOne(
{ "_id": "2" },
{ $set: { "capacité": "45%", "statut": "Entretenue" } }
);
```

4) Afficher les attributs **nombre_cycles, santé_batterie** et **capacité** des batteries dont la capacité est différente de 50% classées par nombre_cycles décroissant (2 pts)

```
db.batteries.find(
        { "capacité": { $ne: "50%" } },
        { "_id": 0, "nombre_cycles": 1, "santé_batterie": 1, "capacité": 1 }
).sort({ "nombre_cycles": -1 });
```

5) Afficher le nombre de batteries regroupées par nombre_***cycles*** (2pts)

```
db.batteries.aggregate([
        {
        $group: {
          _id: "$nombre_cycles",
          count: { $sum: 1 }
        }
        }
]);
```

6) Supprimer les batteries qui n'ont pas le statut ***en service*** ou ***entretenue*** (2pts)

```
db.batteries.deleteMany({ "statut": { $nin: ["en service", "Entretenue"] } });
```

Dossier 4 : Web Dynamique PHP (08 pts)

1) Donner le code PHP de la création de la classe ***Batterie*** avec le champ ***nombre_cycles*** définit privée (1 pts)

```
<?php
class Batterie
{
  public $id;
  public $numero_serie;
  public $capacite;
  public $sante_batterie;
  private $nombre_cycles;
  public $statut;
}
?>
```

2) Implémenter un getter ***getNombreCycles*** et un setter ***setNombreCycles*** pour la propriété ***nombre_cycles*** (1pts) :

```
<?php
class Batterie
{
  public $id;
  public $numero_serie;
  public $capacite;
  public $sante_batterie;
  private $nombre_cycles;
  public $statut;
  //Question 2
  public function getNombreCycles()
  {return $this->nombre_cycles;}
  public function setNombreCycles ($nombre_cycles)
  {$this->nombre_cycles=$nombre_cycles;}
}
?>
```

3) Ajouter à la classe ***Batterie*** une méthode ***batterieDétériorée***, qui retourne vrai si sa capacité est inférieure à 30% et retourne faux dans le cas contraire (2pts)

```
<?php
class Batterie
{
  public $id;
  public $numero_serie;
  public $capacite;
```

```
    public $sante_batterie;
    private $nombre_cycles;
    public $statut;
    //Question 2
    public function getNombreCycles()
    {return $this->nombre_cycles;}
    public function setNombreCycles ($nombre_cycles)
    {$this->nombre_cycles=$nombre_cycles;}
    //Question 3
    public function batterieDétériorée()
    {
      if($this->capacite<="30%")
      {return true;}
      else {return false;}
    }
}
?>
```

4) Créer un tableau de 2 objets ***Batteries*** avec des données de votre choix (2pts)

```
<?php
//Question 4
include "Batterie.php";
//Déclaration d'un tableau
$tab_batteries=array();
//Instancier un premier objet Batterie
$b1= new Batterie();
$b1->id=1;
$b1->numero_serie="n12";
$b1->capacite="20%";
$b1->sante_batterie="Bonne";
$b1->setNombreCycles(1000);
$b1->statut="En Service";
//Instancier un deuxième objet Batterie
$b2= new Batterie();
$b2->id=1;
$b2->numero_serie="n12";
$b2->capacite="20%";
$b2->sante_batterie="Bonne";
$b2->setNombreCycles(900);
$b2->statut="En Service";
//Ajouter les deux objets au tablaeu
array_push($tab_batteries,$b1);
array_push($tab_batteries,$b2);
```


5) Parcourir le tableau de la question 4 pour afficher le nombre de batterie détériorée en utilisant la méthode ***batterieDétériorée*** de la question 3 (2 pts)

```
<?php
//Question 4
include "Batterie.php";
//Déclaration d'un tableau
$tab_batteries=array();
//Instancier un premier objet Batterie
$b1= new Batterie();
$b1->id=1;
$b1->numero_serie="n12";
$b1->capacite="20%";
$b1->sante_batterie="Bonne";
$b1->setNombreCycles(1000);
$b1->statut="En Service";
//Instancier un deuxième objet Batterie
$b2= new Batterie();
$b2->id=1;
$b2->numero_serie="n12";
$b2->capacite="20%";
$b2->sante_batterie="Bonne";
$b2->setNombreCycles(900);
$b2->statut="En Service";
//Ajouter les deux objets au tablaeu
array_push($tab_batteries,$b1);
array_push($tab_batteries,$b2);
//Question 5
$nb_batterie_Détériorée=0;
for ($i=0;$i<count($tab_batteries);$i++)
{
  if($tab_batteries[$i]->batterieDétériorée())
  {
    $nb_batterie_Détériorée++;
  }
}
print("Le nombre de Batterie Détériorée = $nb_batterie_Détériorée");
?>
```

Partie Pratique (60 pts)

Dossier 1 : Gestion des données MySQL (12 pts)

En se basant sur le schéma de base de données ci-dessus, écrire les scripts MySQL qui répondent aux questions suivantes :

1- Ecrire le script de la création de la table **'Trottinette'**. **(2pts)**

```
CREATE TABLE Trottinette (
  id INT PRIMARY KEY AUTO_INCREMENT,
  Matricule VARCHAR(50) NOT NULL UNIQUE,
  id_batterie INT,
  date_derniere_maintenance DATE,
  date_prochaine_maintenance DATE,
  FOREIGN KEY (id_batterie) REFERENCES Batterie(id)
);
```

2- Modifier la table **'Trottinette'** en ajoutant la colonne **'nb_batteries'** qui est de type entier positif et non nul (2pts)

```
ALTER TABLE Trottinette
ADD COLUMN nb_batteries INT UNSIGNED NOT NULL;
```

3- Ecrire un trigger **TR1** qui contrôle la valeur de la colonne **'sante_batterie'** qui doit être un nombre compris entre 0 et 100, lors de l'ajout d'une nouvelle batterie. **(2pts)**

```
DELIMITER $$

CREATE TRIGGER TR1 BEFORE INSERT ON Batterie
FOR EACH ROW
BEGIN
  IF NEW.sante_batterie < 0 OR NEW.sante_batterie > 100 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'La santé de la batterie doit être comprise
entre 0 et 100.';
  END IF;
END$$

DELIMITER ;
```


4- Ecrire un trigger **TR2**, qui modifie la valeur de la colonne statut de 'En Utilisation' à 'Retiré'. Lors d'un ajout dans la table **'Remplacement' (2pts)**

```
DELIMITER $$

CREATE TRIGGER TR2 AFTER INSERT ON Remplacement
FOR EACH ROW
BEGIN
  UPDATE Batterie
  SET statut = 'Retiré'
  WHERE id = NEW.id_ancienne_batterie;
END$$

DELIMITER ;
```

5- Ecrire une fonction **FCT1** qui retourne le nombre total de batteries consommées par un vélo dont l'id est passé en paramètre **(2pts)**

```
DELIMITER $$

CREATE FUNCTION FCT1(id_trottinette INT) RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE total_batteries INT;
  SELECT COUNT(*) INTO total_batteries
  FROM Remplacement
  WHERE id_trottinette = id_trottinette;
  RETURN total_batteries;
END$$

DELIMITER ;
```

6- Gestion des utilisateurs/Rôles : **(2pts)**

a. Créer le rôle **'RoleReparateur'**
b. Donner les droits d'ajout, suppression et modification sur les tables **'Trottinette'** et **'Remplacement'** au rôle 'Role*Reparateur*'
c. Créer l'utilisateur 'Fatima' avec le mot de passe 1111
d. Attribuer le rôle 'RoleReparateur' à l'utilisateur 'Fatima' déjà crée

```
CREATE ROLE RoleReparateur;

GRANT INSERT, DELETE, UPDATE ON Trottinette TO RoleReparateur;
GRANT INSERT, DELETE, UPDATE ON Remplacement TO RoleReparateur;
```

```
CREATE USER 'Fatima' IDENTIFIED BY '1111';
GRANT RoleReparateur TO 'Fatima';
```

Dossier 2 : Développement Front End (24 pts)

Soit l'état initiale du *store* redux :

```
InitialState = {
Trottinette :{
  id: 14,
  Matricule: "1 A 4321",
  Date_derniere_maintenance: "01/06/2025",
  Date_prochaine_maintenance: "01/08/2025",
  BatterieUtilisee: {
         Id : 135,
         Capacite : 73,
         Numero_serie : "BAT-202",
         sante_batterie: 95,
         nombre_cycles: 2127,
         statut: "En Utilisation",
         }
,Reparateurs :[ {id :…,Nom :…}, {id :…,Nom :…},…],
,Batteries :[ {id :…,Numero_serie :…}, {id :…, Numero_serie:…},],
,StatutBatterie :[ "En Stock","En Utilisation","Retiré"]
};
```

Remarque : Il n'est pas demandé à vous de créer le store Redux (actions, reducer, store).

1. Ecrire la composante **DetailsTrottinette.js** qui lit les informations du depuis le store Redux -En utilisant useSelector- et les affiche comme indiqué dans l'image ci-dessous. **(4pts)**

Trottinette Details

Matricule: 1 A 4321
Capacité Batterie Actuelle: 73%
Santé Batterie Actuelle: 95%
Date Dernière Maintenance: 01-06-2025
Date Prochaine Maintenance: 01-08-2025

Figure 1: Details d'une Trottinette (Composante DetailTrottinette.js)

```
import React from 'react';
import { useSelector } from 'react-redux';

const DetailsTrottinette = () => {
  const trottinette = useSelector(state => state.Trottinette);

  return (
    <div>
      <h1>Details de la Trottinette</h1>
      <p>ID: {trottinette.Id}</p>
      <p>Matricule: {trottinette.Matricule}</p>
      <p>Date dernière maintenance:
{trottinette.Date_derniere_maintenance}</p>
      <p>Date prochaine maintenance:
{trottinette.Date_prochaine_maintenance}</p>
      <h2>Batterie Utilisée</h2>
      <p>ID: {trottinette.BatterieUtilisee.Id}</p>
      <p>Capacité: {trottinette.BatterieUtilisee.Capacite}</p>
      <p>Numéro de série: {trottinette.BatterieUtilisee.Numero_serie}</p>
      <p>Santé de la batterie:
{trottinette.BatterieUtilisee.sante_batterie}</p>
      <p>Nombre de cycles: {trottinette.BatterieUtilisee.nombre_cycles}</p>
      <p>Statut: {trottinette.BatterieUtilisee.statut}</p>
    </div>
  );
};

export default DetailsTrottinette;
```

2. Ecrire le code de la composante **AjouterRemplacement.js** (voir figure 2) qui ajoute un Remplacement de la batterie dans la base de données en les données du store et de l'API Suivante : **(6pts)**

Méthode http	POST
URL de l'API	http://localhost:8000/api/AjouterRemplacement

Structure Objet	{
	"id_trottinette":...,
	"id_ancienne_batterie":...,
	"id_nouvelle_batterie":...,
	"date_Remplacement":...,
	"id_Reparateur":...,
	"raison":...
	}



PS :
-Tous les champs sont obligatoires.
-Les valeurs du champs (Trottinette, ancienne batterie, liste batteries,liste Reparateurs) sont hacs depuis le store redux

Figure 2: Ajout d'une Remplacement d'une batterie

```
import React, { useState } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import axios from 'axios';

const AjouterRemplacement = () => {
  const dispatch = useDispatch();
  const trottinette = useSelector(state => state.trottinette);
  const batteries = useSelector(state => state.Batteries);
  const reparateurs = useSelector(state => state.Reparateurs);

  const [formData, setFormData] = useState({
    id_trottinette: trottinette.id,
    id_ancienne_batterie: trottinette.BatterieUtilisee.Id,
    id_nouvelle_batterie: '',
    date_Remplacement: '',
    id_Reparateur: '',
```



```
    raison: ''
  });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await
axios.post('http://localhost:8000/api/AjouterRemplacement', formData);
      console.log(response.data);
      // Mettre à jour le state Redux si nécessaire
    } catch (error) {
      console.error(error);
    }
  };

  return (
    <div>
      <h1>Ajouter un Remplacement de Batterie</h1>
      <form onSubmit={handleSubmit}>
        <label>
          Nouvelle Batterie:
          <select name="id_nouvelle_batterie"
value={formData.id_nouvelle_batterie} onChange={handleChange}>
            {batteries.map(battery => (
              <option key={battery.Id}
value={battery.Id}>{battery.Numero_serie}</option>
            ))}
          </select>
        </label>
        <label>
          Date de Remplacement:
          <input type="date" name="date_Remplacement"
value={formData.date_Remplacement} onChange={handleChange} />
        </label>
        <label>
          Réparateur:
          <select name="id_Reparateur" value={formData.id_Reparateur}
onChange={handleChange}>
            {reparateurs.map(reparateur => (
              <option key={reparateur.Id}
value={reparateur.Id}>{reparateur.Nom}</option>
            ))}
          </select>
        </label>
        <label>
          Raison:
```

```
          <input type="text" name="raison" value={formData.raison}
onChange={handleChange} />
        </label>
        <button type="submit">Ajouter</button>
      </form>
    </div>
  );
};

export default AjouterRemplacement;
```

3. Créer la composante **ListeBatteries**.js qui lit la liste des batteries depuis le store redux et l'affiche selon le schéma ci-dessous : **(6pts)**

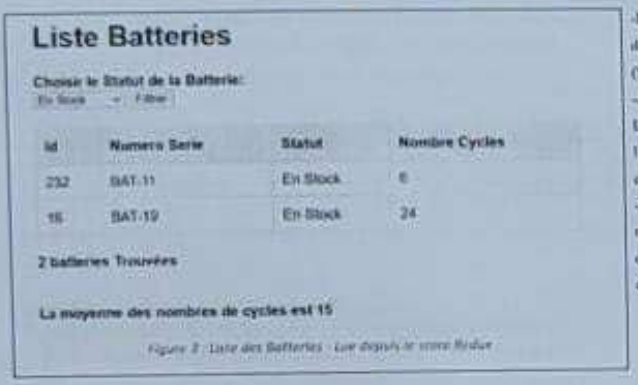


Figure 2 : Liste des Batteries - Lue depuis le store Redux

-La liste déroulante (select) lit les données depuis le store redux (StatutBatterie).
-Après clique sur le bouton 'Filtrer' Les données sont filtrées selon l'option choisie au niveau de la liste déroulante.
-En bas du tableau, on affiche le nombre total d'éléments trouvés ainsi que la moyenne des nombres de cycles

```
import React, { useState } from 'react';
import { useSelector } from 'react-redux';

const ListeBatteries = () => {
  const batteries = useSelector(state => state.Batteries);
  const statutBatterie = useSelector(state => state.StatutBatterie);
  const [filter, setFilter] = useState('');

  const filteredBatteries = batteries.filter(battery =>
    filter ? battery.statut === filter : true
  );

  const totalBatteries = filteredBatteries.length;
  const averageCycles = filteredBatteries.reduce((sum, battery) => sum +
battery.nombre_cycles, 0) / totalBatteries || 0;

  return (
    <div>
      <h1>Liste des Batteries</h1>
```



```
      <select value={filter} onChange={(e) => setFilter(e.target.value)}>
        <option value="">Tous</option>
        {statutBatterie.map(statut => (
          <option key={statut} value={statut}>{statut}</option>
        ))}
      </select>
      <button onClick={() => setFilter('')}>Filtrer</button>
      <table>
        <thead>
          <tr>
            <th>ID</th>
            <th>Numero de série</th>
            <th>Statut</th>
            <th>Nombre de cycles</th>
          </tr>
        </thead>
        <tbody>
          {filteredBatteries.map(battery => (
            <tr key={battery.id}>
              <td>{battery.id}</td>
              <td>{battery.Numero_serie}</td>
              <td>{battery.statut}</td>
              <td>{battery.nombre_cycles}</td>
            </tr>
          ))}
        </tbody>
      </table>
      <p>Total: {totalBatteries}</p>
      <p>Moyenne des cycles: {averageCycles.toFixed(2)}</p>
    </div>
  );
};

export default ListeBatteries;
```

4. Ecrire la composante **Menu.js** qui définit les liens qui mènent vers les composantes : **(4pts)**

Lien	Composante	Titre
/AjouterRempl	AjouterRemplacement.js	Nouveau Remplacement
/ListeBatteries	ListeBatteries.js	Liste Batteries
/DetailsTrottinette	DetailsTrottinette.js	Détails Trottinette

```
import React from 'react';
import { Link } from 'react-router-dom';

const Menu = () => {
  return (
    <nav>
```

```
      <ul>
        <li><Link to="/AjouterRempl">Nouveau Remplacement</Link></li>
        <li><Link to="/ListeBatteries">Liste Batteries</Link></li>
        <li><Link to="/DetailsTrottinette">Détails Trottinette</Link></li>
      </ul>
    </nav>
  );
};

export default Menu;
```

5. Ecrire le code **App.js** qui : **(4pts)**
   - Définit le routage (BrowserRouter, Routes, …)
   - Appel la composante Menu.js crée ci-dessus.
   - Integer le provider du store redux

```
import React from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './store'; // Assurez-vous d'avoir configuré votre store Redux

import Menu from './Menu';
import DetailsTrottinette from './DetailsTrottinette';
import AjouterRemplacement from './AjouterRemplacement';
import ListeBatteries from './ListeBatteries';

const App = () => {
  return (
    <Provider store={store}>
      <BrowserRouter>
        <Menu />
        <Routes>
          <Route path="/AjouterRempl" element={<AjouterRemplacement />} />
          <Route path="/ListeBatteries" element={<ListeBatteries />} />
          <Route path="/DetailsTrottinette" element={<DetailsTrottinette />} />

        </Routes>
      </BrowserRouter>
    </Provider>
  );
};

export default App;
```


Dossier 3 : Développement Back End (24 pts)

1) Donner la commande de la création de migration pour la table Batteries (1pt)

```
php artisan make:migration create_batteries_table
```

2) Donner le code des méthodes ***up*** et ***down*** du fichier de la migration de la table ***Batteries*** en définissant le champ ***numéro_série*** comme unique et le champ ***capacité*** par défaut à «100% » (2pts)

```
    public function up(): void
    {
        Schema::create('batteries', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->string("numero_serie",50)->unique();
            $table->string("capacité",5) )->default("100%");
        $table->string("sante_batterie",50);
        $table->integer("nombre_cycles");
        $table->string("statut",50);

        });
    }

    public function down(): void
    {
        Schema::dropIfExists('batteries');
    }
```

3) Donner la commande pour appliquer la migration (1pt)

```
php artisan migrate
```

4) Donner le code pour créer un fichier ***seeder*** pour la table ***Batterie*** (1pt)

```
php artisan make:seeder batteriesSeeder
```

5) Donner le code de la méthode ***run*** pour assurer l'ajout d'une batterie avec des données de votre choix (2pts)

```
<?php

namespace Database\Seeders;
use Illuminate\Support\Facades\DB;
```

```
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class batteriesSeeder extends Seeder
{
  /**
   * Run the database seeds.
   */
  public function run(): void
  {
    DB::table('clients')->insert
    ([
      "numéro_série"=>"s123",
      "capacité"=>"50%",
      "santé_batterie"=>"bonne",
      "nombre_cycles"=>500,
      "statut"=>"en service"
    ]);
  }
}
```

6) Donner la commande pour appliquer le seeder et peupler la table batteries (1pt)

```
php artisan db:seed --class=batteriesSeeder
```

7) Créer les modèles des tables ***Batteries*** et ***Trottinettes*** avec leurs attributs et fonctions de relations (2pts)

```
class Trottinette extends Model
{
  use HasFactory;
  public function batterie()
  {
    return $this->belongsTo(Batterie::class);
  }
}


class Batterie extends Model
{
  use HasFactory;
  public function trottinettes()
  {
    return $this->hasMany(Trottinette::class);
```



```
  }
}
```

8) Donner la commande de la création du contrôleur ***TrottinetteController*** (2pts)

```
php artisan make:controller TrottinetteController
```

9) Dans le contrôleur ***TrottinetteController*** créer la méthode ***GetBatterie*** qui retourne les informations de la batterie d'une trottinette dont l'id est passé en paramètre (2pts)

```
class TrottinetteController extends Controller
{
 function GetBatterie($trottinetteid)
{
 $trottinette = Trottinette::with('batterie')->find($trottinetteid);
 return $trottinette ->batterie ;
 }
}
```

10) Créer une méthode ***AfficherBatterie*** qui retourne une vue nommée ***InfosBatterie*** avec les données de la méthode ***GetBatterie*** de la question 9 (2pts)

```
class TrottinetteController extends Controller
{
    function GetBatterie($trottinette)
    {
     $trottinette = Trottinette::with('batterie')->find($trottinette);
     return $trottinette ->batterie ;
     }

  function AfficherBatterie($trottinette)
  {
return view("InfosBatterie")->with("infos_batterie",$this->GetBatterie($trottinette));
  }
}
```

11) Créer la vue ***InfosBatterie*** qui affiche les informations de la batterie sous forme d'un tableau comme le montre l'exemple ci-dessous avec les liens modifier et supprimer (2pts)

id	Numéro série	Capacité	Santé batterie	Nombre cycle	Statut	Modifier	Supprimer
1	S2	70	Moyenne	400	En Service	Modifier	Supprimer

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <table>
    <tr>
      <td>Id</td>
      <td>Numéro série</td>
      <td>Capacité</td>
      <td>Santé batterie</td>
      <td>Nombre cycle</td>
      <td>Statut</td>
      <td>Modifier</td>
      <td>Supprimer</td>
    </tr>
    <tr>
      <td>{{$infos_batterie->id}}</td>
      <td>{{$infos_batterie->numero_serie}}</td>
      <td>{{$infos_batterie->capacité}}</td>
      <td>{{$infos_batterie->sante_batterie}}</td>
      <td>{{$infos_batterie->nombre_cycles}}</td>
      <td>{{$infos_batterie->statut}}</td>
      <td>{{$infos_batterie->sante_batterie}}</td>
<td><a href="modifier?id={{ $infos_batterie->id }}">Modifier</a></td>
<td><a href="supprimer?id={{ $infos_batterie->id }}">Modifier</a></td>
    </tr>
  </table>
</body>
</html>
```

12) Dans le contrôleur ***TrottinetteController*** créer la méthode **SuprimerBatterie** qui supprime une batterie dont l'id est passé en paramètre et qui redirige l'utilisateur vers la vue **Welcome** (2pts)


```
class TrottinetteController extends Controller
{
      function GetBatterie($trottinette)
      {
        $trottinette = Trottinette::with('batterie')->find($trottinette);
        return $trottinette ->batterie ;
      }

  function AfficherBatterie($trottinette)
  {
return view("InfosBatterie")->with("infos_batterie",$this->GetBatterie($trottinette));
  }
function SuuprimerBattrie(Request $request)
  {
    $batterie=Batterie::find($request->id);
    $batterie ->Trotinnete()->dissociate();
     Batterie::destroy($request->id);
     return view("welcome");
  }

}
```

13) Donner la commande pour créer un middleware sous le nom «**autorisationSuppression**» (1 pt)

```
php artisan make:middleware autorisationSuppression
```

14) Donner le corps de la méthode ***handle*** pour que le middleware **autorisationSuppression** accepte les requêtes entre 8H30 et 18H30 et dans le cas contraire on affiche un code d'erreur 403 avec le message « Les suppression sont permises entre 8H30 et 18H30 » (2pts)

```
class autorisationSuppression
{
  public function handle(Request $request, Closure $next): Response
  {
      $currentHour = Carbon::now()->format('H:i');
      $startTime = '08:30';
      $endTime = '18:30';

     if ($currentHour < $startTime || $currentHour > $endTime)
     {
      abort(403, ' Les suppressions sont permises entre 8H30 et 18H30');
```

```
            }
            return $next($request);
        }
}
```

15) Donner le code de la route qui mène vers l'action de modification de trottinette en la protégeant par le middleware «autorisationSuppression» (2pts)

```
Route::get('/supprimer',[TrottinetteController::class,'SupprimerBattrie'])
    ->middleware("autorisationSuppression");
```