PLAN DE TEST

Exchange.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarque
1	Vérifie le taux de change pour USD	currencyCode = USD	{ ‘’originalCurrency ‘’ : ‘’GBP’’, ‘’newCurrency” : ‘’USD” , ‘’exchangeRate” : 1.25 }	Mock de ‘callExchangeRateProvider’ pour retourter 1.25. Vérifier que e callback est appelé avec la sortie correcte.	Validé
2	Vérifie le taux de change pour EUR	currencyCode = EUR	{ ‘’originalCurrency ‘’ : ‘’GBP’’, ‘’newCurrency” : ‘’EUR ‘’ , ‘’exchangeRate” : 1.25 }	Mock de ‘callExchangeRateProvider’ pour retourter 1.18. Vérifier que e callback est appelé avec la sortie correcte.	Validé
3	Gère les devises non supportées	currencyCode = ABC	Erreur Currency not supported	Mock de callExchangeRateProvider pour lancer une erreur. Vérifier que l'erreur est levée et que le callback n'est pas appelé.	Aucune

Détails des Scénarios de Test

ID 1: Vérifie le taux de change pour USD

- **Description**: Vérifie que `getExchangeRate` retourne le taux de change correct pour USD.
- **Entrée**: `currencyCode = 'USD'`
- **Sortie Attendue**:

```
{

  "originalCurrency": "GBP",
  "newCurrency": "USD",
  "exchangeRate": 1.25
}
```

- **Méthode de Test**:
  - Mock de `callExchangeRateProvider` pour retourner 1.25.
  - Vérifier que le callback est appelé avec la réponse correcte.
- **Remarques**: Aucune

ID 2: Vérifie le taux de change pour EUR

- **Description**: Vérifie que `getExchangeRate` retourne le taux de change correct pour EUR.
- **Entrée**: `currencyCode = 'EUR'`
- **Sortie Attendue**:

```
{
  "originalCurrency": "GBP",
  "newCurrency": "EUR",
  "exchangeRate": 1.18}
```

- **Méthode de Test**:
  - Mock de `callExchangeRateProvider` pour retourner 1.18.
  - Vérifier que le callback est appelé avec la réponse correcte.
- **Remarques**: Aucune

ID 3: Gère les devises non supportées

- **Description**: Vérifie que `getExchangeRate` lève une erreur pour une devise non supportée.
- **Entrée**: `currencyCode = 'ABC'`
- **Sortie Attendue**: Erreur `Currency not supported`
- **Méthode de Test**:
  - Mock de `callExchangeRateProvider` pour lancer une erreur.
  - Vérifier que l'erreur est levée et que le callback n'est pas appelé.
- **Remarques**: Aucune

exchangeRateProvider.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie le taux de change pour USD	currencyCode = USD	`1.25`	Appel de `callExchangeRateProvider` avec 'USD' et vérification du taux de change.	Validé
2	Vérifie le taux de change pour EUR	`currencyCode = ‘EUR’`	`1.181	Appel de `callExchangeRateProvider` avec ‘EUR’ et vérification du taux de change.	Validé
3	Vérifie le taux de change pour EUR	`currencyCode = ‘NZD’`	`1.93`	Appel de `callExchangeRateProvider` avec ‘NZD’ et vérification du taux de change.	Validé
4	Gère les devises non supportées	`currencyCode = ‘ABC’`	Erreur `Currency not supported`	Appel de `callExchangeRateProvider` avec 'ABC' et vérification que l'erreur est levée.	Validé

Détails des Scénarios de Test

ID 1: Vérifie le taux de change pour USD

- **Description**: Vérifie que `callExchangeRateProvider` retourne le taux de change correct pour USD.
- **Entrée**: `currencyCode = 'USD'`
- **Sortie Attendue**: `1.25`
- **Méthode de Test**:
  - Appeler `callExchangeRateProvider` avec 'USD'.
  - Vérifier que le taux de change retourné est `1.25`.
- **Remarques**: Aucune

ID 2: Vérifie le taux de change pour EUR

- **Description**: Vérifie que `callExchangeRateProvider` retourne le taux de change correct pour EUR.
- **Entrée**: `currencyCode = 'EUR'`
- **Sortie Attendue**: `1.18`
- **Méthode de Test**:
  - Appeler `callExchangeRateProvider` avec 'EUR'.
  - Vérifier que le taux de change retourné est `1.18`.
- **Remarques**: Aucune

ID 3: Vérifie le taux de change pour NZD

- **Description**: Vérifie que `callExchangeRateProvider` retourne le taux de change correct pour NZD.
- **Entrée**: `currencyCode = 'NZD'`
- **Sortie Attendue**: `1.93`
- **Méthode de Test**:
  - Appeler `callExchangeRateProvider` avec 'NZD'.
  - Vérifier que le taux de change retourné est `1.93`.
- **Remarques**: Aucune

ID 4: Gère les devises non supportées

- **Description**: Vérifie que `callExchangeRateProvider` lève une erreur pour une devise non supportée.
- **Entrée**: `currencyCode = 'ABC'`
- **Sortie Attendue**: Erreur `Currency not supported`
- **Méthode de Test**:
  - Appeler `callExchangeRateProvider` avec 'ABC'.
  - Vérifier que l'erreur `Currency not supported` est levée.
- **Remarques**: Aucune

Promotion.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie le calcul du pourcentage de réduction quand le prix actuel est égal au minimum d'achat	`percentage = 10`, `minimumSpend = 60`, `currentPrice = 60`	`54`	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
2	Vérifie le calcul du pourcentage	`percentage = 10`, `minimumSpend = 50`,	`54`	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune

	de réduction quand le prix actuel atteint le minimum d'achat	`currentPrice = 60`			
3	Vérifie qu'aucune réduction n'est appliquée quand le prix est inférieur au minimum d'achat	`percentage = 10`, `minimumSpend = 50`, `currentPrice = 40`	`40`	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
4	Vérifie que le prix reste inchangé pour une réduction de 0%	`percentage = 0`, `minimumSpend = 50`, `currentPrice = 60`	`60`	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
5	Vérifie que le prix devient 0 pour une réduction de 100%	`percentage = 100`, `minimumSpend = 50`, `currentPrice = 60`	`0`	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
6	Gère les pourcentages invalides	`percentage = -10`, `minimumSpend = 50`, `currentPrice = 60`	Erreur "Percentage cannot be negative"	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
7	Gère les pourcentages supérieurs à 100	`percentage = 110`, `minimumSpend = 50`, `currentPrice = 60`	Erreur "Percentage cannot be greater than 100"	Appel de `calculatePercentageDiscount` et vérification du résultat.	Aucune
8	Vérifie le calcul de la réduction en argent quand le prix actuel est égal au minimum d'achat	`discount = 10`, `minimumSpend = 60`, `currentPrice = 60`	`50`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune

9	Vérifie le calcul de la réduction en argent quand le prix actuel est supérieur au minimum d'achat	`discount = 10`, `minimumSpend = 60`, `currentPrice = 70`	`60`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
10	Vérifie qu'aucune réduction en argent n'est appliquée quand le prix actuel est inférieur au minimum d'achat	`discount = 10`, `minimumSpend = 60`, `currentPrice = 40`	`40`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
11	Gère les réductions en argent supérieures au prix actuel	`discount = 70`, `minimumSpend = 50`, `currentPrice = 60`	`-10`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
12	Vérifie que le prix reste inchangé pour une réduction de 0 en argent	`discount = 0`, `minimumSpend = 50`, `currentPrice = 60`	`60`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
13	Gère le cas où le prix actuel est de zéro	`discount = 10`, `minimumSpend = 50`, `currentPrice = 0`	`0`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
14	Gère le cas où le minimum d'achat est de zéro	`discount = 10`, `minimumSpend = 0`, `currentPrice = 60`	`50`	Appel de `calculateMoneyOff` et vérification du résultat.	Aucune
15	Vérifie la génération correcte d'un code de parrainage valide	`userId1 = "abc123"`	Codes de parrainage valide	Appel de `generateReferralCode` et vérification du format et de l'inclusion de l'ID utilisateur.	Aucune

16	Vérifie que les codes de parrainage sont uniques pour différents utilisateurs	`userId1 = "abc123"` `userID2 = ‘’abc456’’`	Codes de parrainage différents	Appel de `generateReferralCode` avec différents IDs utilisateurs et vérification que les codes sont uniques.	Aucune
17	Applique correctement une réduction de type MONEYOFF	`discountCode = "DISCOUNT10"`, `currentTotal = 100`	`90`	Mocker `axios.get` pour retourner une réduction de type MONEYOFF et vérifier que le montant final est correct.	Utilisation de `vi.mock` pour mocker axios.
18	Applique correctement une réduction de type PERCENTAGEOFF	`discountCode = "DISCOUNT10"`, `currentTotal = 100`		Mocker `axios.get` pour retourner une réduction de type PERCENTAGEOFF et vérifier que le montant final est correct.	Utilisation de `vi.mock` pour mocker axios.
19	Ne pas appliquer la réduction si le code est invalide	`discountCode = "INVALID"`, `currentTotal = 100`	`100`	Mocker `axios.get` pour retourner une réponse invalide et vérifier que le montant final reste inchangé.	Utilisation de `vi.mock` pour mocker axios.
20	Ne pas appliquer la réduction si le minimum d'achat n'est pas atteint	`discountCode = "DISCOUNT10"`, `currentTotal = 40`	`40`	Mocker `axios.get` pour retourner une réduction valide mais avec un minimum d'achat non atteint et vérifier que le montant reste inchangé.	Utilisation de `vi.mock` pour mocker axios.
21	Gérer les erreurs lors de l'application de la réduction	`discountCode = "DISCOUNT10"`, `currentTotal = 60`	`60`	Mocker `axios.get` pour retourner une erreur réseau et vérifier que le montant final reste inchangé.	Utilisation de `vi.mock` pour mocker axios.

Détails des Scénarios de Test

ID 1-7: Tests pour `calculatePercentageDiscount`

- **Description**: Vérifie le comportement de la fonction `calculatePercentageDiscount` pour différentes combinaisons de pourcentage de réduction, prix minimum et prix actuel.
- **Entrée**: Variantes de pourcentages, prix minimum, et prix actuel.
- **Sortie Attendue**: Résultats calculés correctement ou erreurs levées pour les entrées invalides.
- **Méthode de Test**:

- Appeler `calculatePercentageDiscount` avec différentes combinaisons d'entrées.
- Vérifier que les résultats retournés ou les erreurs levées sont conformes aux attentes.

ID 8-14: Tests pour `calculateMoneyOff`

- **Description**: Vérifie le comportement de la fonction `calculateMoneyOff` pour différentes combinaisons de réduction en argent, prix minimum et prix actuel.
- **Entrée**: Variantes de réductions, prix minimum, et prix actuel.
- **Sortie Attendue**: Résultats calculés correctement ou erreurs levées pour les entrées invalides.
- **Méthode de Test**:
  - Appeler `calculateMoneyOff` avec différentes combinaisons d'entrées.
  - Vérifier que les résultats retournés ou les erreurs levées sont conformes aux attentes.

ID 15-16: Tests pour `generateReferralCode`

- **Description**: Vérifie que les codes de parrainage générés sont valides et uniques pour différents utilisateurs.
- **Entrée**: Identifiants utilisateur.
- **Sortie Attendue**: Codes de parrainage valides et uniques.
- **Méthode de Test**:
  - Appeler `generateReferralCode` avec différents IDs utilisateurs.
  - Vérifier le format, la longueur, et l'unicité des codes générés.

ID 17-21: Tests pour `applyDiscount`

- **Description**: Vérifie le comportement de la fonction `applyDiscount` avec différents types de réductions et conditions.
- **Entrée**: Codes de réduction, total actuel.
- **Sortie Attendue**: Montants finaux calculés correctement ou erreurs gérées.
- **Méthode de Test**:
  - Mocker `axios.get` pour retourner différentes réponses de réduction.
  - Appeler `applyDiscount` avec différents codes et vérifier les résultats ou les comportements en cas d'erreurs.

4o

Exception.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie la création de `InvalidEventName Error` avec le bon message	`errorMessage = "Event name cannot exceed 200	Instance de `InvalidEventName Error` avec le message fourni	Création de `InvalidEventName Error` et vérification de l'instance et du message.	Aucune

		characters"`			
2	Vérifie la création de `InvalidEventPriceError` avec le bon message	`errorMessage = "Event price must be more or equal to 0"`	Instance de `InvalidEventPriceError` avec le message fourni	Création de `InvalidEventPriceError` et vérification de l'instance et du message.	Aucune
3	Vérifie la création de `InvalidUsernameError` avec le bon message	`errorMessage = "Username is invalid"`	Instance de `InvalidUsernameError` avec le message fourni	Création de `InvalidUsernameError` et vérification de l'instance et du message.	Aucune
4	Vérifie la création de `InvalidReferralCodeError` avec le bon message	`errorMessage = "Referral code is invalid"`	Instance de `InvalidReferralCodeError` avec le message fourni	Création de `InvalidReferralCodeError` et vérification de l'instance et du message.	Aucune
5	Vérifie la création de `UserHasAccountError` avec le bon message	`errorMessage = "User already has an account"`	Instance de `UserHasAccountError` avec le message fourni	Création de `UserHasAccountError` et vérification de l'instance et du message.	Aucune

Détails des Scénarios de Test

ID 1: Test pour `InvalidEventNameError`

- **Description**: Vérifie que l'exception `InvalidEventNameError` est créée avec le message correct.
- **Entrée**: Message d'erreur "Event name cannot exceed 200 characters".
- **Sortie Attendue**: Une instance de `InvalidEventNameError` avec le message spécifié.
- **Méthode de Test**:
  - Créer une instance de `InvalidEventNameError` avec le message donné.
  - Vérifier que l'instance est du type `InvalidEventNameError`.
  - Vérifier que le message de l'instance est correct.

ID 2: Test pour `InvalidEventPriceError`

- **Description**: Vérifie que l'exception `InvalidEventPriceError` est créée avec le message correct.
- **Entrée**: Message d'erreur "Event price must be more or equal to 0".
- **Sortie Attendue**: Une instance de `InvalidEventPriceError` avec le message spécifié.
- **Méthode de Test**:
  - Créer une instance de `InvalidEventPriceError` avec le message donné.

- Vérifier que l'instance est du type `InvalidEventPriceError`.
- Vérifier que le message de l'instance est correct.

ID 3: Test pour `InvalidUsernameError`

- **Description**: Vérifie que l'exception `InvalidUsernameError` est créée avec le message correct.
- **Entrée**: Message d'erreur "Username is invalid".
- **Sortie Attendue**: Une instance de `InvalidUsernameError` avec le message spécifié.
- **Méthode de Test**:
  - Créer une instance de `InvalidUsernameError` avec le message donné.
  - Vérifier que l'instance est du type `InvalidUsernameError`.
  - Vérifier que le message de l'instance est correct.

ID 4: Test pour `InvalidReferralCodeError`

- **Description**: Vérifie que l'exception `InvalidReferralCodeError` est créée avec le message correct.
- **Entrée**: Message d'erreur "Referral code is invalid".
- **Sortie Attendue**: Une instance de `InvalidReferralCodeError` avec le message spécifié.
- **Méthode de Test**:
  - Créer une instance de `InvalidReferralCodeError` avec le message donné.
  - Vérifier que l'instance est du type `InvalidReferralCodeError`.
  - Vérifier que le message de l'instance est correct.

ID 5: Test pour `UserHasAccountError`

- **Description**: Vérifie que l'exception `UserHasAccountError` est créée avec le message correct.
- **Entrée**: Message d'erreur "User already has an account".
- **Sortie Attendue**: Une instance de `UserHasAccountError` avec le message spécifié.
- **Méthode de Test**:
  - Créer une instance de `UserHasAccountError` avec le message donné.
  - Vérifier que l'instance est du type `UserHasAccountError`.
  - Vérifier que le message de l'instance est correct.

Event.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie la création d'un objet `Event`	`(1, 'Concert', 50, 100, 50, new Date())`	Instance de `Event` avec les propriétés correctes	Créer une instance de `Event` et vérifier ses propriétés	Aucune
2	Vérifie que `isSoldOut` retourne `true` si	`Event(1, 'Concert', 50,	`true`	Créer un évènement complet et	Aucune

	l’événement est complet	100, 0, new Date())`		appeler `isSoldOut`	
3	Vérifie que `isSoldOut` retourne `false` si l’événement n’est pas complet	`Event(1, ‘Concert’, 50, 100, 10, new Date())`	`false`	Créer un évènement non complet et appeler `isSoldOut`	Aucune
4	Vérifie que `getTagLine` retourne ‘’Event Sold Out!’’ si l’événement est complet	`Event(1, ‘Concert’, 50, 100, 0, new Date())`, `(10, true)`	`’’Event Sold Out!’’`	Créer un évènement complet et appeler `getTagLine`	Aucune
5	Vérifie que `getTagLine` retourne ‘’Hurry only X tickets lefts!’’ si les tickets sont inférieurs à minimum TicketCount	`Event(1, ‘Concert’, 50, 100, 2, new Date(‘2024-03-15))`, `(5, false)`	`’’Hurry only 2 tickets left’’`	Créer un évènement avec peu de tickets restants et appeler `getTagLine`	Aucune
6	Vérifie que `getTagLine` retourne la bonne phrase pour un événement populaire	`Event(1, ‘Concert’, 50, 100, 2, new Date())`, `(10, true)`	`"This Event is getting a lot of interest. Don't miss out, purchase your ticket now!"`	Créer un événement populaire et appeler `getTagLine`	Aucune
7	Vérifie que `getTagLine` retourne la bonne phrase pour un événement régulier	`Event(1, ‘Concert’, 50, 100, 20, new Date())`, `(10, false)`	`"Don't miss out, purchase your ticket now!"`	Créer un événement régulier et appeler `getTagLine`	Aucune
8	Vérifie la création réussie d'un événement avec des entrées valides	`’’Concert’, 50, 100`	Instance de `Event` avec les propriétés correctes	Appeler `createEvent` avec des entrées valides et vérifier les propriétés de l'événement créé	Aucune
10	Vérifie que `createEvent` lance une erreur pour un nom d'événement trop long	`’’A’.repeat(201), 50, 100`	Lancer ‘InvalidEventNameError`	Appeler `createEvent `avec un nom d'événement invalide et vérifier l'erreur lancée	Aucune

11	Vérifie que `createEvent` lance une erreur pour un prix d'événement invalide	``'Concert', -10, 100`	Lancer 'InvalidEventPriceError`	Appeler `createEvent `avec un prix d'événement invalide et vérifier l'erreur lancée	Aucune
12	Appeler `createEvent` avec un prix d'événement non numérique et vérifier l'erreur lancée	``'Concert', 'fifty', 100`	Lancer `InvalidEventPriceError`	Appeler `createEvent` avec un prix d'événement non numérique et vérifier l'erreur lancée	Aucune
13	Vérifie que `createEvent` lance une erreur pour un nombre de tickets disponibles invalide	``'Concert', 50, 0`	Lancer `InvalidEventPriceError`	Appeler `createEvent` avec un nombre de tickets disponibles invalide et vérifier l'erreur lancée	Aucune
14	Vérifie que `createEvent` lance une erreur pour un nombre de tickets disponibles non numérique	``'Concert', 50, 'hundered'``	Lancer `InvalidEventPriceError`	Appeler `createEvent` avec un nombre de tickets disponibles non numérique et vérifier l'erreur lancée	Aucune

Détails des Scénarios de Test

ID 1: Test pour la création de la classe `Event`

- **Description**: Vérifie que la classe `Event` est correctement instanciée avec les propriétés spécifiées.
- **Entrée**: `(1, 'Concert', 50, 100, 50, new Date()).`
- **Sortie Attendue**: Une instance de `Event` avec les propriétés correctes.
- **Méthode de Test**:
  - Créer une instance de `Event` avec les entrées spécifiées.
  - Vérifier que l'instance est du type `Event`.
  - Vérifier que les propriétés de l'instance sont correctes.

ID 2: Test pour la fonction `isSoldOut` (événement complet)

- **Description**: Vérifie que la fonction `isSoldOut` retourne `true` si l'événement est complet.

- **Entrée**: `Event(1, 'Concert', 50, 100, 0, new Date())`.
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec 0 tickets restants.
    - Appeler la fonction `isSoldOut` avec cet événement.
    - Vérifier que le résultat est `true`.

ID 3: Test pour la fonction `isSoldOut` (événement non complet)

- **Description**: Vérifie que la fonction `isSoldOut` retourne `false` si l'événement n'est pas complet.
- **Entrée**: `Event(1, 'Concert', 50, 100, 10, new Date())`.
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec des tickets restants.
    - Appeler la fonction `isSoldOut` avec cet événement.
    - Vérifier que le résultat est `false`.

ID 4: Test pour la fonction `getTagLine` (événement complet)

- **Description**: Vérifie que la fonction `getTagLine` retourne "Event Sold Out!" si l'événement est complet.
- **Entrée**: `Event(1, 'Concert', 50, 100, 0, new Date()),(10, true)`.
- **Sortie Attendue**: `"Event Sold Out!"`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec 0 tickets restants.
    - Appeler la fonction `getTagLine` avec cet événement, un minimum de tickets de 10 et un événement populaire.
    - Vérifier que le résultat est `"Event Sold Out!"`.

ID 5: Test pour la fonction `getTagLine` (peu de tickets restants)

- **Description**: Vérifie que la fonction `getTagLine` retourne "Hurry only X tickets left!" si les tickets restants sont inférieurs à minimumTicketCount.
- **Entrée**: `Event(1, 'Concert', 50, 100, 2, new Date('2024-03-15')),(5, false)`.
- **Sortie Attendue**: `"Hurry only 2 tickets left!"`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec 2 tickets restants.
    - Appeler la fonction `getTagLine` avec cet événement, un minimum de tickets de 5 et un événement non populaire.
    - Vérifier que le résultat est `"Hurry only 2 tickets left!"`.

ID 6: Test pour la fonction `getTagLine` (événement populaire)

- **Description**: Vérifie que la fonction `getTagLine` retourne la bonne phrase pour un événement populaire.
- **Entrée**: `Event(1, 'Concert', 50, 100, 20, new Date()),(10, true)`.

- **Sortie Attendue**: `"This Event is getting a lot of interest. Don't miss out, purchase your ticket now!"`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec 20 tickets restants.
    - Appeler la fonction `getTagLine` avec cet événement, un minimum de tickets de 10 et un événement populaire.
    - Vérifier que le résultat est `"This Event is getting a lot of interest. Don't miss out, purchase your ticket now!"`.

ID 7: Test pour la fonction `getTagLine` (événement régulier)

- **Description**: Vérifie que la fonction `getTagLine` retourne la bonne phrase pour un événement régulier.
- **Entrée**: `Event(1, 'Concert', 50, 100, 20, new Date()),(10, false)`.
- **Sortie Attendue**: `"Don't miss out, purchase your ticket now!"`.
- **Méthode de Test**:
    - Créer une instance de `Event` avec 20 tickets restants.
    - Appeler la fonction `getTagLine` avec cet événement, un minimum de tickets de 10 et un événement non populaire.
    - Vérifier que le résultat est `"Don't miss out, purchase your ticket now!"`.

ID 8: Test pour la fonction `createEvent` (entrée valide)

- **Description**: Vérifie la création réussie d'un événement avec des entrées valides.
- **Entrée**: `'Concert', 50, 100`.
- **Sortie Attendue**: Une instance de `Event` avec les propriétés correctes.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec des entrées valides.
    - Vérifier que le résultat est une instance de `Event` avec les propriétés spécifiées.

ID 9: Test pour la fonction `createEvent` (nom d'événement invalide)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un nom d'événement invalide.
- **Entrée**: `123, 50, 100`.
- **Sortie Attendue**: Lancer `InvalidEventNameError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un nom d'événement invalide.
    - Vérifier que l'erreur `InvalidEventNameError` est lancée.

ID 10: Test pour la fonction `createEvent` (nom d'événement trop long)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un nom d'événement trop long.
- **Entrée**: `'A'.repeat(201), 50, 100`.
- **Sortie Attendue**: Lancer `InvalidEventNameError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un nom d'événement trop long.

    - Vérifier que l'erreur `InvalidEventNameError` est lancée.

ID 11: Test pour la fonction `createEvent` (prix d'événement invalide)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un prix d'événement invalide.
- **Entrée**: `'Concert', -10, 100`.
- **Sortie Attendue**: Lancer `InvalidEventPriceError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un prix d'événement invalide.
    - Vérifier que l'erreur `InvalidEventPriceError` est lancée.

ID 12: Test pour la fonction `createEvent` (prix d'événement non numérique)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un prix d'événement non numérique.
- **Entrée**: `'Concert', 'fifty', 100`.
- **Sortie Attendue**: Lancer `InvalidEventPriceError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un prix d'événement non numérique.
    - Vérifier que l'erreur `InvalidEventPriceError` est lancée.

ID 13: Test pour la fonction `createEvent` (tickets disponibles invalides)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un nombre de tickets disponibles invalide.
- **Entrée**: `'Concert', 50, 0`.
- **Sortie Attendue**: Lancer `InvalidEventPriceError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un nombre de tickets disponibles invalide.
    - Vérifier que l'erreur `InvalidEventPriceError` est lancée.

ID 14: Test pour la fonction `createEvent` (tickets disponibles non numériques)

- **Description**: Vérifie que la fonction `createEvent` lance une erreur pour un nombre de tickets disponibles non numérique.
- **Entrée**: `'Concert', 50, 'hundred'`.
- **Sortie Attendue**: Lancer `InvalidEventPriceError`.
- **Méthode de Test**:
    - Appeler la fonction `createEvent` avec un nombre de tickets disponibles non numérique.
    - Vérifier que l'erreur `InvalidEventPriceError` est lancée.

Ce plan de test détaillé couvre tous les scénarios de test définis dans le fichier `event.test.js`, garantissant que les fonctions et les classes sont correctement testées pour des comportements attendus et des cas d'erreurs.

Filter.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie si l'événement a lieu aujourd'hui	`Event(1, 'Concert', 50, 100, 50, new Date())`	`true` si l'événement a lieu aujourd'hui	Appeler `today` avec un événement qui a lieu aujourd'hui et vérifier le résultat retourné	Aucune
2	Vérifie si l'événement n'a pas lieu aujourd'hui	`Event(1, 'Concert', 50, 100, 50, new Date('2023-01-01'))`	`false` si l'événement n'a pas lieu aujourd'hui	Appeler `today` avec un événement qui n'a pas lieu aujourd'hui et vérifier le résultat retourné	Aucune
3	Vérifie si l'événement a lieu dans les 7 prochains jours	`Event(1, 'Concert', 50, 100, 50, futureDate)`	`true` si l'événement a lieu dans les 7 prochains jours	Appeler `next7Days` avec un événement qui a lieu dans les 7 prochains jours et vérifier le résultat retourné	Aucune
4	Vérifie si l'événement n'a pas lieu dans les 7 prochains jours	`Event(1, 'Concert', 50, 100, 50, futureDate)`	`false` si l'événement n'a pas lieu dans les 7 prochains jours	Appeler `next7Days` avec un événement qui n'a pas lieu dans les 7 prochains jours et vérifier le résultat retourné	Aucune
5	Vérifie si l'événement a lieu dans les 30 prochains jours	`Event(1, 'Concert', 50, 100, 50, futureDate)`	`true` si l'événement a lieu dans les 30 prochains jours	Appeler `next30Days` avec un événement qui a lieu dans les 30 prochains jours et vérifier le résultat retourné	Aucune
6	Vérifie si l'événement n'a pas lieu dans les 30 prochains jours	`Event(1, 'Concert', 50, 100, 50, futureDate)`	`false` si l'événement n'a pas lieu dans les 30 prochains jours	Appeler `next30Days` avec un événement qui n'a pas lieu dans les 30 prochains jours et vérifier le résultat retourné	Aucune

Détails des Scénarios de Test

ID 1: Test de la fonction `today` (événement aujourd'hui)

- **Description**: Vérifie si la fonction `today` renvoie `true` si l'événement a lieu aujourd'hui.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, new Date()).`
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
  - Appeler la fonction `today` avec un événement qui a lieu aujourd'hui.

    - Vérifier que le résultat est `true`.

ID 2: Test de la fonction `today` (événement pas aujourd'hui)

- **Description**: Vérifie si la fonction `today` renvoie `false` si l'événement n'a pas lieu aujourd'hui.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, new Date('2023-01-01')).`
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
    - Appeler la fonction `today` avec un événement qui n'a pas lieu aujourd'hui.
    - Vérifier que le résultat est `false`.

ID 3: Test de la fonction `next7Days` (événement dans les 7 prochains jours)

- **Description**: Vérifie si la fonction `next7Days` renvoie `true` si l'événement a lieu dans les 7 prochains jours.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, futureDate).`
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
    - Créer une date future dans les 7 prochains jours.
    - Appeler la fonction `next7Days` avec un événement qui a lieu dans les 7 prochains jours.
    - Vérifier que le résultat est `true`.

ID 4: Test de la fonction `next7Days` (événement pas dans les 7 prochains jours)

- **Description**: Vérifie si la fonction `next7Days` renvoie `false` si l'événement n'a pas lieu dans les 7 prochains jours.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, futureDate).`
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
    - Créer une date future après les 7 prochains jours.
    - Appeler la fonction `next7Days` avec un événement qui n'a pas lieu dans les 7 prochains jours.
    - Vérifier que le résultat est `false`.

ID 5: Test de la fonction `next30Days` (événement dans les 30 prochains jours)

- **Description**: Vérifie si la fonction `next30Days` renvoie `true` si l'événement a lieu dans les 30 prochains jours.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, futureDate).`
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
    - Créer une date future dans les 30 prochains jours.
    - Appeler la fonction `next30Days` avec un événement qui a lieu dans les 30 prochains jours.
    - Vérifier que le résultat est `true`.

ID 6: Test de la fonction `next30Days` (événement pas dans les 30 prochains jours)

- **Description**: Vérifie si la fonction `next30Days` renvoie `false` si l'événement n'a pas lieu dans les 30 prochains jours.
- **Entrée**: `Event(1, 'Concert', 50, 100, 50, futureDate).`
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
    - Créer une date future après les 30 prochains jours.
    - Appeler la fonction `next30Days` avec un événement qui n'a pas lieu dans les 30 prochains jours.
    - Vérifier que le résultat est `false`.

Ce plan de test détaille tous les scénarios de test pour les fonctions de filtre dans le fichier `filter.test.js`, garantissant que chaque fonction est testée pour des comportements attendus dans différentes situations.

Search.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérifie si la fonction renvoie les événements correspondant au prédicat de recherche	Liste d'événements, prédicat de recherche	Tableau d'événements qui correspondent au prédicat de recherche	Appeler `getEvents` avec une liste d'événements et un prédicat de recherche qui correspond à certains événements et vérifier le résultat retourné	Aucune
2	Vérifie si la fonction renvoie un tableau vide si aucun événement ne correspond au prédicat de recherche	Liste d'événements, prédicat de recherche	Tableau vide	Appeler `getEvents` avec une liste d'événements et un prédicat de recherche qui ne correspond à aucun événement et vérifier que le résultat est un tableau vide	Aucune

Détails des Scénarios de Test

ID 1: Test de la fonction `getEvents` (événements correspondant au prédicat de recherche)

- **Description**: Vérifie si la fonction `getEvents` renvoie les événements qui correspondent au prédicat de recherche spécifié.
- **Entrée**: Liste d'événements et un prédicat de recherche qui correspond à certains événements.
- **Sortie Attendue**: Tableau d'événements qui correspondent au prédicat de recherche.
- **Méthode de Test**:
    - Créer une liste d'événements avec certains événements.
    - Définir un prédicat de recherche qui correspond à certains de ces événements.
    - Appeler la fonction `getEvents` avec la liste d'événements et le prédicat de recherche.
    - Vérifier que le résultat est un tableau contenant les événements correspondants.

ID 2: Test de la fonction `getEvents` (aucun événement correspondant au prédicat de recherche)

- **Description**: Vérifie si la fonction `getEvents` renvoie un tableau vide si aucun événement ne correspond au prédicat de recherche spécifié.
- **Entrée**: Liste d'événements et un prédicat de recherche qui ne correspond à aucun événement.
- **Sortie Attendue**: Tableau vide.
- **Méthode de Test**:
  - Créer une liste d'événements avec certains événements.
  - Définir un prédicat de recherche qui ne correspond à aucun de ces événements.
  - Appeler la fonction `getEvents` avec la liste d'événements et le prédicat de recherche.
  - Vérifier que le résultat est un tableau vide.

Users.test.js :

ID	Description	Entrée	Sortie Attendue	Methode de Test	Remarques
1	Vérifie si une instance d'utilisateur est créée avec les propriétés correctes	ID utilisateur, nom d'utilisateur	Instance d'utilisateur avec les propriétés spécifiées	Créer une finstance d'utilisateur avec un ID et un nom d'utilisateur spécifiés et vérifier les propriétés de l'instance résultante	Aucune
2	Vérifie si la fonction `userExists `retourne `true` si le nom d'utilisateur existe	Nom d'utilisateur existant	`true`	Appeler la fonction `userExists` avec un nom d'utilisateur existant et vérifier que la valeur retournée est `true`	Aucune
3	Vérifie si la fonction `userExists `retourne `false` si le nom d'utilisateur n'existe pas	Nom d'utilisateur inexistant	`false`	Appeler la fonction `userExists` avec un nom d'utilisateur inexistant et vérifier que la valeur retournée est `false`	Aucune
4	Vérifie si la fonction `createUserId` retourne un ID utilisateur valide	Aucune entrée	ID utilisateur unique	Appeler la fonction `createUserId` et vérifier que l'ID retourné est unique et valide	Aucune

Détails des Scénarios de Test

ID 1: Test de la création d'une instance d'utilisateur avec les propriétés correctes

- **Description**: Vérifie si une instance d'utilisateur est créée avec les propriétés correctes.
- **Entrée**: ID utilisateur et nom d'utilisateur.
- **Sortie Attendue**: Instance d'utilisateur avec les propriétés spécifiées.
- **Méthode de Test**:
  - Créer une instance d'utilisateur avec un ID et un nom d'utilisateur spécifiés.
  - Vérifier que les propriétés de l'instance créée correspondent aux valeurs spécifiées.

ID 2: Test de la fonction `userExists` (nom d'utilisateur existant)

- **Description**: Vérifie si la fonction `userExists` retourne `true` si le nom d'utilisateur existe.
- **Entrée**: Nom d'utilisateur existant.
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
  - Appeler la fonction `userExists` avec un nom d'utilisateur existant.
  - Vérifier que la valeur retournée est `true`.

ID 3: Test de la fonction `userExists` (nom d'utilisateur inexistant)

- **Description**: Vérifie si la fonction `userExists` retourne `false` si le nom d'utilisateur n'existe pas.
- **Entrée**: Nom d'utilisateur inexistant.
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
  - Appeler la fonction `userExists` avec un nom d'utilisateur inexistant.
  - Vérifier que la valeur retournée est `false`.

ID 4: Test de la fonction `createUserId` (ID utilisateur valide)

- **Description**: Vérifie si la fonction `createUserId` retourne un ID utilisateur valide.
- **Entrée**: Aucune entrée.
- **Sortie Attendue**: ID utilisateur unique.
- **Méthode de Test**:
  - Appeler la fonction `createUserId`.
  - Vérifier que l'ID retourné est unique et valide.

Account.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques

1	Vérification de la création d'un objet d'achat avec les propriétés correctes	Nom de l’événement, nombre de billets, coûts total	Instance d'achat avec les propriétés spécifiées	Créer une instance d'achat avec un nom d'événement, un nombre de billets et un coût total spécifiés, puis vérifier les propriétés de l'instance créée	Aucune
2	Vérification de la fonction `isValidUserName` (noms d'utilisateur invalides)	Noms d'utilisateur invalides	`false`	Appeler la fonction `isValidUserName` avec des noms d'utilisateur invalides et vérifier que la valeur retournée est `false`	Aucune
3	Vérification de la fonction `isValidUserName` (noms d'utilisateur valides)	Noms d'utilisateur valides	`true`	Appeler la fonction `isValidUserName` avec des noms d'utilisateur invalides et vérifier que la valeur retournée est `true`	Aucune
4	Vérification de la création d'un compte pour un nom d'utilisateur valide	Noms d'utilisateur valides	Données du compte utilisateur créé	Appeler la fonction `createAccount` avec un nom d'utilisateur valide et vérifier les données du compte utilisateur créé	Aucune
5	Gestion de l'erreur pour un nom d'utilisateur existant	Noms d'utilisateur existant	Erreur "User already exists"	Appeler la fonction `createAccount` avec un nom d'utilisateur existant et vérifier les données du compte utilisateur créé	Aucune
6	Vérification de la récupération des achats antérieurs pour un ID utilisateur valide	ID utilisateur valide	Historique des achats antérieurs	Appeler la fonction `getPastPurchases` avec un ID utilisateur valide et vérifier que l'historique des achats antérieurs est récupéré avec succès	Aucune
7	Gestion de l'erreur si l'historique des achats ne peut pas être récupéré	ID utilisateur valide	Erreur "Failed to get purchase history"	Appeler la fonction `getPastPurchases` et vérifier que l'erreur "Failed to get purchase history" est levée si l'historique des achats ne peut pas être récupéré	Aucune

Détails des Scénarios de Test

ID 1: Test de la création d'un objet d'achat avec les propriétés correctes

- **Description**: Vérifie si une instance d'achat est créée avec les propriétés correctes.
- **Entrée**: Nom de l'événement, nombre de billets, coût total.
- **Sortie Attendue**: Instance d'achat avec les propriétés spécifiées.

- **Méthode de Test**:
    - Créer une instance d'achat avec un nom d'événement, un nombre de billets et un coût total spécifiés.
    - Vérifier que les propriétés de l'instance créée correspondent aux valeurs spécifiées.

ID 2: Test de la fonction `isValidUserName` (noms d'utilisateur invalides)

- **Description**: Vérifie si la fonction `isValidUserName` retourne `false` pour les noms d'utilisateur invalides.
- **Entrée**: Noms d'utilisateur invalides.
- **Sortie Attendue**: `false`.
- **Méthode de Test**:
    - Appeler la fonction `isValidUserName` avec des noms d'utilisateur invalides.
    - Vérifier que la valeur retournée est `false`.

ID 3: Test de la fonction `isValidUserName` (noms d'utilisateur valides)

- **Description**: Vérifie si la fonction `isValidUserName` retourne `true` pour les noms d'utilisateur valides.
- **Entrée**: Noms d'utilisateur valides.
- **Sortie Attendue**: `true`.
- **Méthode de Test**:
    - Appeler la fonction `isValidUserName` avec des noms d'utilisateur valides.
    - Vérifier que la valeur retournée est `true`.

ID 4: Test de la création d'un compte pour un nom d'utilisateur valide

- **Description**: Vérifie si un compte est créé avec succès pour un nom d'utilisateur valide.
- **Entrée**: Nom d'utilisateur valide.
- **Sortie Attendue**: Données du compte utilisateur

Purchase.test.js :

ID		Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Vérification de la fonction `getPurchaseHistory`	ID utilisateur valide	Historique d'achat pour l'utilisateur	Appeler la fonction `getPurchaseHistory` avec un ID utilisateur valide et vérifier que l'historique des achats est récupéré avec succès	Aucune
2	Vérification de l'analyse correcte de la réponse d'achat	Données d'achat valides	Données d'analysées correctement	Appeler la fonction `parsePurchaseResponse` avec des données d'achat valides et vérifier que les données sont analysées correctement	Aucune

Détails des Scénarios de Test

ID 1: Test de la fonction `getPurchaseHistory`

- **Description**: Vérifie si l'historique d'achat est récupéré avec succès pour un ID utilisateur valide.
- **Entrée**: ID utilisateur valide.
- **Sortie Attendue**: Historique d'achat pour l'utilisateur.
- **Méthode de Test**:
  - Mock de la réponse de l'API Axios pour simuler une réponse réussie.
  - Appeler la fonction `getPurchaseHistory` avec un ID utilisateur valide.
  - Vérifier que l'historique d'achat récupéré contient les données attendues.

ID 2: Test de l'analyse correcte de la réponse d'achat

- **Description**: Vérifie si la fonction `parsePurchaseResponse` analyse correctement les données d'achat.
- **Entrée**: Données d'achat valides.
- **Sortie Attendue**: Données d'achat analysées correctement.
- **Méthode de Test**:
  - Appeler la fonction `parsePurchaseResponse` avec des données d'achat valides.
  - Vérifier que les données d'achat sont analysées correctement et que les instances d'achat sont créées avec les propriétés attendues.

Basket.test.js :

ID	Description	Entrée	Sortie Attendue	Méthode de Test	Remarques
1	Calculer le total pour un panier vide	Tableau d'éléments du panier vide	0	Appeler `calculateTotal` avec un tableau vide et vérifier que le résultat est 0	
2	Calculer le total pour un seul élément	Tableau avec un élément (ticketPrice: 20, quantity: 1)	20	Appeler `calculateTotal` avec un tableau contenant un élément et vérifier que le total est 20	
3	Calculer le total pour plusieurs éléments	Tableau avec plusieurs éléments	40	Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et vérifier que le total est correct (10*3* + *5*2 = 40)	
4	Appliquer une reduction	Tableau avec plusieurs éléments, et une réduction	22	Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et une réduction, vérifier que le total est correct (10 + 5*3 - 3 = 22)	

5	Ne pas appliquer de réduction si null	Tableau avec plusieurs éléments, réduction null	39	Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et une réduction null, vérifier que le total est correct (13*3 = 39)	
6	Ne pas afficher de publicités pour un utilisateur premium	Utilisateur premium	False	Appeler `showAdverts` avec un utilisateur premium et vérifier que le résultat est false	
7	Afficher des publicités pour un utilisateur non-premium	Utilisateur non-premium	True	Appeler `showAdverts` avec un utilisateur non-premium et vérifier que le résultat est true	
8	Trouver des éléments du panier correspondants	Tableau avec des éléments, requête de recherche	Tableau avec les éléments correspond ants	Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche, vérifier que les éléments correspondants sont retournés	
9	Retourner un tableau vide pour aucune correspondance	Tableau avec des éléments, requête de recherche	Tableau vide	Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche sans correspondance, vérifier que le résultat est un tableau vide	
10	Effectuer une recherche insensible à la casse	Tableau avec des éléments, requête de recherche	Tableau avec les éléments correspond ants	Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche insensible à la casse, vérifier que les éléments correspondants sont retournés	
11	Trouver un élément du panier correspondant	Tableau avec des éléments, événement correspondant	Événement correspondar	Appeler `getBasketItem` avec un tableau contenant des éléments et un événement correspondant, vérifier que l'élément correspondant est retourné	
12	Retourner null pour aucune correspondance	Tableau avec des éléments, événement non correspondant	null	Appeler `getBasketItem` avec un tableau contenant des éléments et un événement non correspondant, vérifier que le résultat est null	
13	Gérer plusieurs éléments correspondants	Tableau avec plusieurs	Premier élément correspondar	Appeler `getBasketItem` avec un tableau contenant plusieurs éléments correspondants,	

		éléments correspondants		vérifier que le premier élément correspondant est retourné	
14	Créer un nouvel élément du panier	Tableau vide, événement et nombre de billets requis	Nouvel élément du panier	Appeler `createBasketItem` avec un tableau vide, un événement et un nombre de billets requis, vérifier que le nouvel élément du panier est créé	
15	Retourner null pour un élément existant	Tableau avec un élément existant, événement et nombre de billets requis	null	Appeler `createBasketItem` avec un tableau contenant un élément existant, un événement et un nombre de billets requis, vérifier que le résultat est null	
16	Sérialiser les éléments du panier en JSON	Tableau avec des éléments du panier	Tableau d'objets sérialisés en JSON	Appeler `serializeBasketItemsToJson` avec un tableau contenant des éléments du panier, vérifier que les éléments sont correctement sérialisés	
17	Ne pas modifier les éléments du panier d'origine	Tableau avec des éléments du panier	Éléments du panier d'origine inchangés	Appeler `serializeBasketItemsToJson` avec un tableau contenant des éléments du panier, modifier le résultat, vérifier que les éléments d'origine sont inchangés	

Détails des Scénarios de Test

ID 1: Calculer le total pour un panier vide

- **Description**: Vérifie si le total est correctement calculé pour un panier vide.
- **Entrée**: Tableau d'éléments du panier vide.
- **Sortie Attendue**: 0.
- **Méthode de Test**: Appeler `calculateTotal` avec un tableau vide et vérifier que le résultat est 0.

ID 2: Calculer le total pour un seul élément

- **Description**: Vérifie si le total est correctement calculé pour un panier contenant un seul élément.
- **Entrée**: Tableau avec un élément (ticketPrice: 20, quantity: 1).
- **Sortie Attendue**: 20.

- **Méthode de Test**: Appeler `calculateTotal` avec un tableau contenant un élément et vérifier que le total est 20.

ID 3: Calculer le total pour plusieurs éléments

- **Description**: Vérifie si le total est correctement calculé pour un panier contenant plusieurs éléments.
- **Entrée**: Tableau avec plusieurs éléments.
- **Sortie Attendue**: 40.
- **Méthode de Test**: Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et vérifier que le total est correct (10*3* + *5*2 = 40).

ID 4: Appliquer une réduction

- **Description**: Vérifie si la réduction est correctement appliquée lors du calcul du total.
- **Entrée**: Tableau avec plusieurs éléments et une réduction.
- **Sortie Attendue**: 22.
- **Méthode de Test**: Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et une réduction, vérifier que le total est correct (10 + 5*3 - 3 = 22).

ID 5: Ne pas appliquer de réduction si null

- **Description**: Vérifie si aucune réduction n'est appliquée si la réduction est null.
- **Entrée**: Tableau avec plusieurs éléments, réduction null.
- **Sortie Attendue**: 39.
- **Méthode de Test**: Appeler `calculateTotal` avec un tableau contenant plusieurs éléments et une réduction null, vérifier que le total est correct (13*3 = 39).

ID 6: Ne pas afficher de publicités pour un utilisateur premium

- **Description**: Vérifie si les publicités ne sont pas affichées pour un utilisateur premium.
- **Entrée**: Utilisateur premium.
- **Sortie Attendue**: false.
- **Méthode de Test**: Appeler `showAdverts` avec un utilisateur premium et vérifier que le résultat est false.

ID 7: Afficher des publicités pour un utilisateur non-premium

- **Description**: Vérifie si les publicités sont affichées pour un utilisateur non-premium.
- **Entrée**: Utilisateur non-premium.
- **Sortie Attendue**: true.
- **Méthode de Test**: Appeler `showAdverts` avec un utilisateur non-premium et vérifier que le résultat est true.

ID 8: Trouver des éléments du panier correspondants

- **Description**: Vérifie si les éléments du panier correspondants sont trouvés.
- **Entrée**: Tableau avec des éléments, requête de recherche.
- **Sortie Attendue**: Tableau avec les éléments correspondants.

- **Méthode de Test**: Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche, vérifier que les éléments correspondants sont retournés.

ID 9: Retourner un tableau vide pour aucune correspondance

- **Description**: Vérifie si un tableau vide est retourné pour aucune correspondance.
- **Entrée**: Tableau avec des éléments, requête de recherche.
- **Sortie Attendue**: Tableau vide.
- **Méthode de Test**: Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche sans correspondance, vérifier que le résultat est un tableau vide.

ID 10: Effectuer une recherche insensible à la casse

- **Description**: Vérifie si la recherche est insensible à la casse.
- **Entrée**: Tableau avec des éléments, requête de recherche.
- **Sortie Attendue**: Tableau avec les éléments correspondants.
- **Méthode de Test**: Appeler `searchBasket` avec un tableau contenant des éléments et une requête de recherche insensible à la casse, vérifier que les éléments correspondants sont retournés.

ID 11: Trouver un élément du panier correspondant

- **Description**: Vérifie si l'élément du panier correspondant est trouvé.
- **Entrée**: Tableau avec des éléments, événement correspondant.
- **Sortie Attendue**: Événement correspondant.
- **Méthode de Test**: Appeler `getBasketItem` avec un tableau contenant des éléments et un événement correspondant, vérifier que l'élément correspondant est retourné.

ID 12: Retourner null pour aucune correspondance

- **Description**: Vérifie si null est retourné pour aucune correspondance.
- **Entrée**: Tableau avec des éléments, événement non correspondant.
- **Sortie Attendue**: null.
- **Méthode de Test**: Appeler `getBasketItem` avec un tableau contenant des éléments et un événement non correspondant, vérifier que le résultat est null.

ID 13: Gérer plusieurs éléments correspondants

- **Description**: Vérifie si le premier élément correspondant est trouvé lorsqu'il y a plusieurs éléments correspondants.
- **Entrée**: Tableau avec plusieurs éléments correspondants.
- **Sortie Attendue**: Premier élément correspondant.
- **Méthode de Test**: Appeler `getBasketItem` avec un tableau contenant plusieurs éléments correspondants, vérifier que le premier élément correspondant est retourné.

ID 14: Créer un nouvel élément du panier

- **Description**: Vérifie si un nouvel élément du panier est créé.
- **Entrée**: Tableau vide, événement et nombre de billets requis.

- **Sortie Attendue**: Nouvel élément du panier.
- **Méthode de Test**: Appeler `createBasketItem` avec un tableau vide, un événement et un nombre de billets requis, vérifier que le nouvel élément du panier est créé.

ID 15: Retourner null pour un élément existant

- **Description**: Vérifie si null est retourné pour un élément existant.
- **Entrée**: Tableau avec un élément existant, événement et nombre de billets requis.
- **Sortie Attendue**: null.
- **Méthode de Test**: Appeler `createBasketItem` avec un tableau contenant un élément existant, un événement et un nombre de billets requis, vérifier que le résultat est null.

ID 16: Sérialiser les éléments du panier en JSON

- **Description**: Vérifie si les éléments du panier sont correctement sérialisés en JSON.
- **Entrée**: Tableau avec des éléments du panier.
- **Sortie Attendue**: Tableau d'objets sérialisés en JSON.
- **Méthode de Test**: Appeler `serializeBasketItemsToJson` avec un tableau contenant des éléments du panier, vérifier que les éléments sont correctement sérialisés.

ID 17: Ne pas modifier les éléments du panier d'origine

- **Description**: Vérifie si les éléments du panier d'origine ne sont pas modifiés après la sérialisation.
- **Entrée**: Tableau avec des éléments du panier.
- **Sortie Attendue**: Éléments du panier d'origine inchangés.
- **Méthode de Test**: Appeler `serializeBasketItemsToJson` avec un tableau contenant des éléments du panier, modifier le résultat, vérifier que les éléments d'origine sont inchangés.

Technologie utilisée : Javascript, Vitest, Mocking.

Conclusion :

Ces plans de test couvrent les cas de base de notre application. Il est important de noter ces plans peuvent être étendu pour inclure de tests plus complets, tels que des tests d’intégration ou des tests de performance.