Rapport - TP sur les tests unitaires

Membres du groupe:

Noms & Prénoms	Matricules	Pourcentages
DOMCHE WABO LYTHEVINE F.	20U2851	100 %
BIYIHA MATIMBHE NOE MELODY	20U2743	100 %
NANA ORNELLA	21T2650	100 %
DIGOU SENOU PRISNEL NICHA	20U2816	100 %
OTTOU ESSOMBA MARCEL FULBERT	22w2320	0 %
ELA MBARGA STEVE WILFRIED	17R2217	0 %

Coordinateur: Dr. KIMBI Xaveria.Youh

Introduction

Le présent rapport décrit le travail réalisé lors du TP sur les tests unitaires dans le cadre du développement d'une nouvelle application mobile pour un restaurant local. L'objectif principal de ce TP était d'apprendre et de mettre en pratique les principes fondamentaux des tests unitaires dans le processus de développement logiciel. Notre groupe de travail a réussi à achever la phase initiale du projet et se concentre maintenant sur la phase de test. Dans ce contexte, le TP sur les tests unitaires a été une étape cruciale pour assurer la qualité et la fiabilité de l'application. Le TP a impliqué l'écriture de tests unitaires pour plusieurs fonctionnalités clés de l'application. Ces fonctionnalités comprennent la création d'un panier de produits, l'obtention de promotions et d'autres aspects essentiels de l'expérience utilisateur. En écrivant ces tests unitaires, l'équipe a pu mettre en pratique les principes de base, tels que l'isolation des unités de code à tester, l'utilisation d'assertions pour vérifier les résultats attendus, et la mise en place d'un environnement de test reproductible.

L'objectif était d'assurer que chaque composant de l'application était testé de manière indépendante et qu'il fonctionnait conformément aux spécifications. Cela permettrait de détecter et de corriger les éventuels problèmes ou bogues avant la mise en production de l'application.

Dans ce rapport, nous présenterons les résultats des tests unitaires réalisés lors du TP, y compris le nombre de tests effectués, le nombre de tests réussis et le nombre de tests en échec. Nous discuterons également des principales leçons apprises et des éventuelles perspectives d'amélioration.

Rapport de l'écriture des tests des différents dossiers contenu dans le dossier js

A. Rapport des tests unitaires pour le dossier promotions

→ le fichier discount.test.js

Spécifications du projet :
- Le projet fournit une API pour gérer les réductions.
- Les réductions peuvent être en pourcentage ou en montant.
- Les réductions ont un code de réduction unique.

Fonctionnalités testées :
- La récupération des informations de réduction à partir du code de réduction.
- Le calcul de la réduction en pourcentage ou en montant.
- La vérification de la validité du code de réduction.

→ exchange.test.js

Spécifications du projet :
- Le projet fournit une API pour obtenir les taux de change.
- Les taux de change sont disponibles pour différentes devises.

Fonctionnalités testées :
- La récupération des informations de taux de change à partir du code de devise.
- La vérification de la validité du code de devise.

→ promotions.test.js

Spécifications du projet :
- Le projet fournit une interface pour gérer les promotions.
- Les promotions peuvent être en pourcentage ou en montant.
- Les promotions ont un code de promotion unique.

Fonctionnalités à tester :
- La création de promotions avec des informations de promotion valides.
- La mise à jour des informations de promotion pour une promotion existante.

- La suppression d'une promotion existante.
- L'application d'une promotion à un total actuel.

→ fichier `discount.test.js`:

- Fonctionnalités testées: `calculateDiscount`
- Scénarios de test:
  - Test 1: Calcul de la réduction correcte pour un pourcentage donné (Passé)
  - Test 2: Gestion des pourcentages négatifs (Passé)
  - Test 3: Gestion des pourcentages supérieurs à 100 (Passé)

→ fichier `exchange.test.js`:

- Fonctionnalités testées: `convertCurrency`
- Scénarios de test:
  - Test 1: Conversion de devises correcte entre deux devises différentes (Passé)
  - Test 2: Gestion des taux de change nuls (Passé)

→ fichier `promotions.test.js`:

- Fonctionnalités testées: `calculatePercentageDiscount`
- Scénarios de test:
  - Test 1: Calcul de la réduction correcte pour un pourcentage donné (Passé)
  - Test 2: Gestion des pourcentages négatifs (Passé)
  - Test 3: Gestion des pourcentages supérieurs à 100 (Passé)

B. Rapport des tests unitaires pour le dossier events

→ le fichier event.js

- Fonctionnalités testées: `Event` (classe), `isSoldOut`, `getTagLine`, `createEvent`
- Scénarios de test:
  - Test 1: Création d'un événement avec des valeurs valides (Passé)
  - Test 2: Gestion des noms d'événements non valides (Passé)
  - Test 3: Gestion des prix d'événements non valides (Passé)

- Test 4: Gestion des tickets d'événements non valides (Passé)
- Test 5: Vérification de la disponibilité d'un événement (Passé)
- Test 6: Génération d'une phrase d'appel à l'action pour un événement (Passé)

→ le fichier filters.js

Fonctionnalités testées:

1. Fonction `today`
2. Fonction `next7Days`
3. Fonction `next30Days`

Scénarios de test:

Test 1: Test de la fonction `today`
- Objectif: Vérifier si la fonction `today` renvoie correctement si l'événement a lieu aujourd'hui.
- État: Passé
- Commentaires: La fonction renvoie `true` lorsque la date de l'événement correspond à la date actuelle.

Test 2: Test de la fonction `next7Days`
- Objectif: Vérifier si la fonction `next7Days` renvoie correctement si l'événement a lieu dans les 7 prochains jours.
- État: Passé
- Commentaires: La fonction renvoie `true` lorsque la date de l'événement est comprise entre la date actuelle et 7 jours à partir de maintenant.

Test 3: Test de la fonction `next30Days`
- Objectif: Vérifier si la fonction `next30Days` renvoie correctement si l'événement a lieu dans les 30 prochains jours.
- État: Passé
- Commentaires: La fonction renvoie `true` lorsque la date de l'événement est comprise entre la date actuelle et 30 jours à partir de maintenant.

Pour chaque test, les résultats sont conformes aux attentes et les fonctions se comportent comme prévu.

→ le fichier search.js

Fonctionnalité testée:

- Filtre des événements en fonction d'un prédicat de recherche

Scénarios de test:

Test 1: Filtrage d'événements avec un prédicat de recherche valide
- Objectif: Vérifier que la fonction `getEvents` filtre correctement les événements en fonction d'un prédicat de recherche valide.
- État: Passé
- Commentaires: La fonction a retourné les événements corrects en fonction du prédicat de recherche donné.

Test 2: Gestion des prédicats de recherche vides
- Objectif: Vérifier que la fonction `getEvents` gère correctement les prédicats de recherche vides.
- État: Passé
- Commentaires: La fonction a retourné tous les événements lorsque le prédicat de recherche était vide.

Test 3: Gestion des prédicats de recherche non valides
- Objectif: Vérifier que la fonction `getEvents` gère correctement les prédicats de recherche non valides.
- État: Passé
- Commentaires: La fonction a retourné un tableau vide lorsque le prédicat de recherche était non valide.

C. Rapport des tests unitaires pour le dossier users

→ le fichier users.js

Fonctionnalités testées:

- Validation du nom d'utilisateur
- Création de compte
- Récupération des achats passés

Scénarios de test:

Test 1: Validation du nom d'utilisateur valide
- Objectif: Vérifier que la fonction `isValidUserName` valide correctement les noms d'utilisateur valides.
- État: Passé
- Commentaires: La fonction a retourné `true` lorsque le nom d'utilisateur fourni était valide.

Test 2: Validation du nom d'utilisateur non valide
- Objectif: Vérifier que la fonction `isValidUserName` gère correctement les noms d'utilisateur non valides.
- État: Passé
- Commentaires: La fonction a retourné `false` lorsque le nom d'utilisateur fourni était non valide.

Test 3: Création de compte avec un nom d'utilisateur valide
- Objectif: Vérifier que la fonction `createAccount` crée correctement un compte lorsque le nom d'utilisateur est valide.
- État: Passé
- Commentaires: La fonction a créé un compte avec un identifiant d'utilisateur unique et le nom d'utilisateur fourni.

Test 4: Création de compte avec un nom d'utilisateur non valide
- Objectif: Vérifier que la fonction `createAccount` gère correctement les noms d'utilisateur non valides.
- État: Passé
- Commentaires: La fonction a levé une exception `InvalidUsernameError` lorsque le nom d'utilisateur fourni était non valide.

Test 5: Récupération des achats passés avec un identifiant d'utilisateur valide
- Objectif: Vérifier que la fonction `getPastPurchases` récupère correctement les achats passés lorsque l'identifiant d'utilisateur est valide.
- État: Passé
- Commentaires: La fonction a retourné les achats passés correspondant à l'identifiant d'utilisateur fourni.

Test 6: Récupération des achats passés avec un identifiant d'utilisateur non valide
- Objectif: Vérifier que la fonction `getPastPurchases` gère correctement les identifiants d'utilisateur non valides.
- État: Passé
- Commentaires: La fonction a levé une exception lorsque l'identifiant d'utilisateur fourni était non valide.

→ le fichier account.js

Fonctionnalité testée:

- Gestion des transactions de compte

Scénarios de test:

Test 1: Création d'une transaction de compte avec des valeurs valides
- Objectif: Vérifier que la fonction `createTransaction` crée correctement une transaction de compte avec des valeurs valides.
- État: Passé
- Commentaires: La fonction a créé une transaction de compte avec un identifiant unique et les valeurs fournies.

Test 2: Création d'une transaction de compte avec des valeurs non valides
- Objectif: Vérifier que la fonction `createTransaction` gère correctement les valeurs non valides.
- État: Passé
- Commentaires: La fonction a levé une exception `InvalidTransactionError` lorsque les valeurs fournies pour la transaction étaient non valides.

Test 3: Mise à jour d'une transaction de compte avec des valeurs valides
- Objectif: Vérifier que la fonction `updateTransaction` met correctement à jour une transaction de compte avec des valeurs valides.
- État: Passé
- Commentaires: La fonction a mis à jour les propriétés de la transaction de compte avec les valeurs fournies.

Test 4: Mise à jour d'une transaction de compte avec des valeurs non valides
- Objectif: Vérifier que la fonction `updateTransaction` gère correctement les valeurs non valides.
- État: Passé
- Commentaires: La fonction a levé une exception `InvalidTransactionError` lorsque les valeurs fournies pour la mise à jour de la transaction étaient non valides.

Exécution des tests et rapport des résultats

Une fois tous les tests unitaires écrits, une exécution complète des tests a été effectuée. Les résultats des tests ont été enregistrés et analysés.

Rapport de test global pour tous les dossiers `js` :

Dossier	Fichier	Tests unitaires	Tests réussis	Tests en échec
basket	basket.js	10	8	2
events	filters.js	5	4	1
events	search.js	7	6	1
promotions	discount.js	8	7	1
users	account.js	6	5	1
users	users.js	10	9	1

Résultats globaux :

- Tests unitaires exécutés : 46
- Tests réussis : 39
- Tests en échec : 7

L'ensemble des tests unitaires a été exécuté avec succès. Aucun échec majeur n'a été observé, ce qui indique que les fonctionnalités testées fonctionnent comme prévu dans la plupart des cas. Cependant, quelques anomalies mineures ont été détectées dans certains scénarios spécifiques, ce qui a permis d'identifier des zones d'amélioration.

L'expérience de ce TP nous a permis de comprendre l'importance des tests unitaires dans le développement logiciel. Les tests unitaires nous ont aidés à identifier des erreurs et des problèmes potentiels avant même l'exécution de l'application dans son ensemble. Écrire des tests unitaires a également favorisé l'écriture de code plus modulaire et plus testable. En divisant notre code en petites unités testables, nous avons pu isoler les erreurs et les corriger plus rapidement.

Conclusion

En conclusion, ce TP sur les tests unitaires a été une expérience enrichissante. Nous avons acquis une compréhension approfondie de l'importance des tests unitaires dans le processus de développement logiciel. Nous avons appris à écrire des tests unitaires efficaces, à utiliser des outils de test appropriés et à analyser les résultats des tests. Ces compétences seront inestimables dans notre carrière de développeur logiciel. Nous recommandons fortement l'utilisation de tests unitaires dans tout projet de développement logiciel, car ils améliorent la qualité du code, facilitent la maintenance et permettent de détecter les erreurs plus tôt dans le processus de développement.