INF 352 : SOFTWARE TESTING

GROUPE 11

NOUBISSIE KAMGA WILFRIED……...20U2671

DEUDJIE MBADI SABINE AXELLE ……19M2144

NDEUNA NGANA OUSMAN SINCLAIR ……21T2433

FOGUE GUEMGNE NELLY SORELLE ……20U2993

Responsable :

M. Régis Atemengue

ANNEE ACADEMIQUE : 2023 - 2024

DESCRIPTION DU PROCESSUS DE CONSTRUCTION DES DIFFERENTS PLAN DE TESTS

1. Définition des objectifs

L'objectif que nous nous fixons pour TP, est de vérifier le bon fonctionnement des principales fonctions et fonctionnalités de l'application mise à notre disposition ; notamment toutes les fonctions se trouvant dans le dossier nommé js.

2. Analyse de la conception

Il s'agissait ici pour nous d'identifier les différents modules de l'application afin d'être mieux informé quant-aux types de test dont on aura besoin.

3. Identification des types de tests

Tout au long de notre projet, nous allons procéder par des tests unitaires sur les différents modules de l'application, et les tests d'intégration pour vérifier les interactions entre les modules.

4. Conception des cas de test

Ici, nous allons donner les différents cas de test pour chaque fonction et chaque fonctionnalité :

a) Test pour le fichier « basket.js »

i. Fonction CalculateTotal

Cas de test :

- ✓ Doit calculer le prix d'une simple basket
- ✓ Doit calculer le prix de plusieurs baskets
- ✓ Doit retourner 0 pour une basket vide
- ✓ Doit appliquer le discount sur le prix total

ii. Fonction ShowAdverts

Cas de test :

- ✓ Doit retourner faux si l'utilisateur est premium
- ✓ Doit retourner vrai si l'utilisateur n'est pas premium

iii. Fonction SearchBasket

Cas de test :

- ✓ Doit retourner un tableau vide si l'élément recherché n'est pas dans le tableau d'éléments
- ✓ Doit retourner un tableau non vide si l'élément recherché est dans le tableau d'éléments

iv. Fonction GetBasketItem

Cas de test :

- ✓ Doit retourner l'objet si l'id de l'élément recherché correspond à un id dans la liste d'éléments
- ✓ Doit retourner null si l'id de l'élément recherché n'est pas trouvé

v. Fonction CreateBasketItem

Cas de test :

- ✓ Doit créer un élément si cet élément n'existe pas déjà
- ✓ Doit retourner null si l'élément existe déjà

vi. Fonction SerializeBasketItemToJson

Cas de test :

- ✓ Doit sérialiser les données si le format est correct

b) Test pour le fichier « exceptions.js »

i. Classes Erreur

Cas de test :

- ✓ Doit créer un InvalidEventNameError
- ✓ Doit créer un InvalidEventPriceError
- ✓ Doit créer un InvalidReferralCodeError
- ✓ Doit créer un InvalidUsernameError
- ✓ Doit créer un UserHasAccountError
- ✓ Doit créer un UserHasAccountError

c) Test pour le fichier « event.js »

i. Fonction IsSoldOut

Cas de test :

- ✓ Doit retourner vrai s'il n'y a plus aucun ticket restant
- ✓ Doit retourner faux s'il y'a encore des tickets restant

ii. Fonction GetTagLine

Cas de test :

- ✓ Doit retourner une expression qui match avec l'expression « Sold Out » s'il n'y a plus aucun ticket
- ✓ Doit retourner une expression qui match avec ticket s'il reste juste 1 ticket et le nombre de ticket restant est inférieur au nombre de ticket minimum
- ✓ Doit retourner une expression qui match avec tickets s'il reste plusieurs tickets et le nombre de ticket restant est inférieur au nombre de ticket minimum
- ✓ Doit retourner "This Event is getting a lot of interest. Don't miss out, purchase your ticket now! " si le film est populaire
- ✓ Doit retourner "Don't miss out, purchase your ticket now" si le film n'est pas populaire

iii. Fonction CreateEvent

Cas de test :

- ✓ Doit créer un nouvel élément si les données entrées sont valides
- ✓ Doit lever une erreur InvalidEventNameError si le nom n'est pas du type string
- ✓ Doit lever une erreur InvalidEventPriceError si le prix est inférieur à 0
- ✓ Doit lever une erreur InvalidEventPriceError si le nombre de ticket disponible est inférieur a 1

d) Test pour le fichier « filters.js »

i. Fonction Today

Cas de test :

- ✓ Doit retourner vrai si la date de l'évènement est la date du jour en cour
- ✓ Doit retourner faux si la date de l'évènement n'est pas la date du jour en cour

ii. Fonction Next7Days

Cas de test :

- ✓ Doit retourner vrai si la date de l'évènement est au plus tard dans 7 jours
- ✓ Doit retourner faux si la date de l'évènement n'est pas dans les 7 jours qui suivent

iii. Fonction Next30Days

Cas de test :

- ✓ Doit retourner vrai si la date de l'évènement est au plus tard dans 30 jours
- ✓ Doit retourner faux si la date de l'évènement n'est pas dans les 30 jours qui suivent

e) Test pour le fichier « search.js »

i. Fonction GetEvents

Cas de test :

- ✓ Doit filtrer les événements avec une fonction de recherche valide
- ✓ Doit retourner un tableau vide si on recherche dans un tableau d'événements vide
- ✓ Doit lever une erreur si on utilise un attribut incorrect pour la recherche
- ✓ Doit retourner tous les événements si la fonction de recherche retourne toujours vrai

f) Test pour le fichier « discount.js »

i. Fonction GetDiscount

Cas de test :

- ✓ Doit retourner les données venues de l'API
- ✓ Doit lever une erreur si l'appel d'API échoue

g) Test pour le fichier « exchange.js »

i. Fonction GetExchangeRate

Cas de test :

- ✓ Doit appeler la fonction exchangeRateProvider et retourner le résultat attendu
- ✓ Doit gérer les erreurs de l'appel de exchangeRateProvider

h) Test pour le fichier « exchangeRateProvider.js »

i. Constante ExchangeRateProvider

Cas de test :

- ✓ Doit retourner l'échange correcte pour l'USD
- ✓ Doit retourner l'échange correcte pour l'EUR
- ✓ Doit retourner l'échange correcte pour l'NZD
- ✓ Doit lever une erreur pour une monnaie non supportée

i) Test pour le fichier « promotions.js »

i. Fonction CalculatePercentageDiscount

Cas de test :

- ✓ Doit retourner le resultat du produit entre le prix et le discount divisé par 100 si le prix est superieur au spend minimum
- ✓ Doit retourner le prix si le prix est inférieur au spend minimum

ii. Fonction CalculateMoneyOff

Cas de test :

- ✓ Doit retourner la différence entre le prix et le discount si le prix est supérieur ou égal au spend minimum
- ✓ Doit retourner le prix si le prix est inférieur au spend minimum

iii. Fonction GenerateReferralCode

Cas de test :

- ✓ Doit retouner une chaine de caractère qui match avec « #FRIEND-# »

iv. Fonction ApplyDiscount

Cas de test :

- ✓ Doit retourner le total actuel si les données sont invalid
- ✓ Doit appeler la fonction calculateMoneyOff si les données sont valides et le type de donnée est MONEYOFF
- ✓ Doit appeler la fonction calculatePercentageDiscount si les données sont valides et le type de donnée est PERCENTAGEOFF

j) Test pour le fichier « users.js »

i. Class User

Cas de test :

- ✓ Doit retourner vrai si l'utilisateur existe déjà
- ✓ Doit retourner faux si l'utilisateur n'existe pas encore
- ✓ Doit créer un nouvel id d'utilisateur
- ✓ Doit créer un nouvel utilisateur

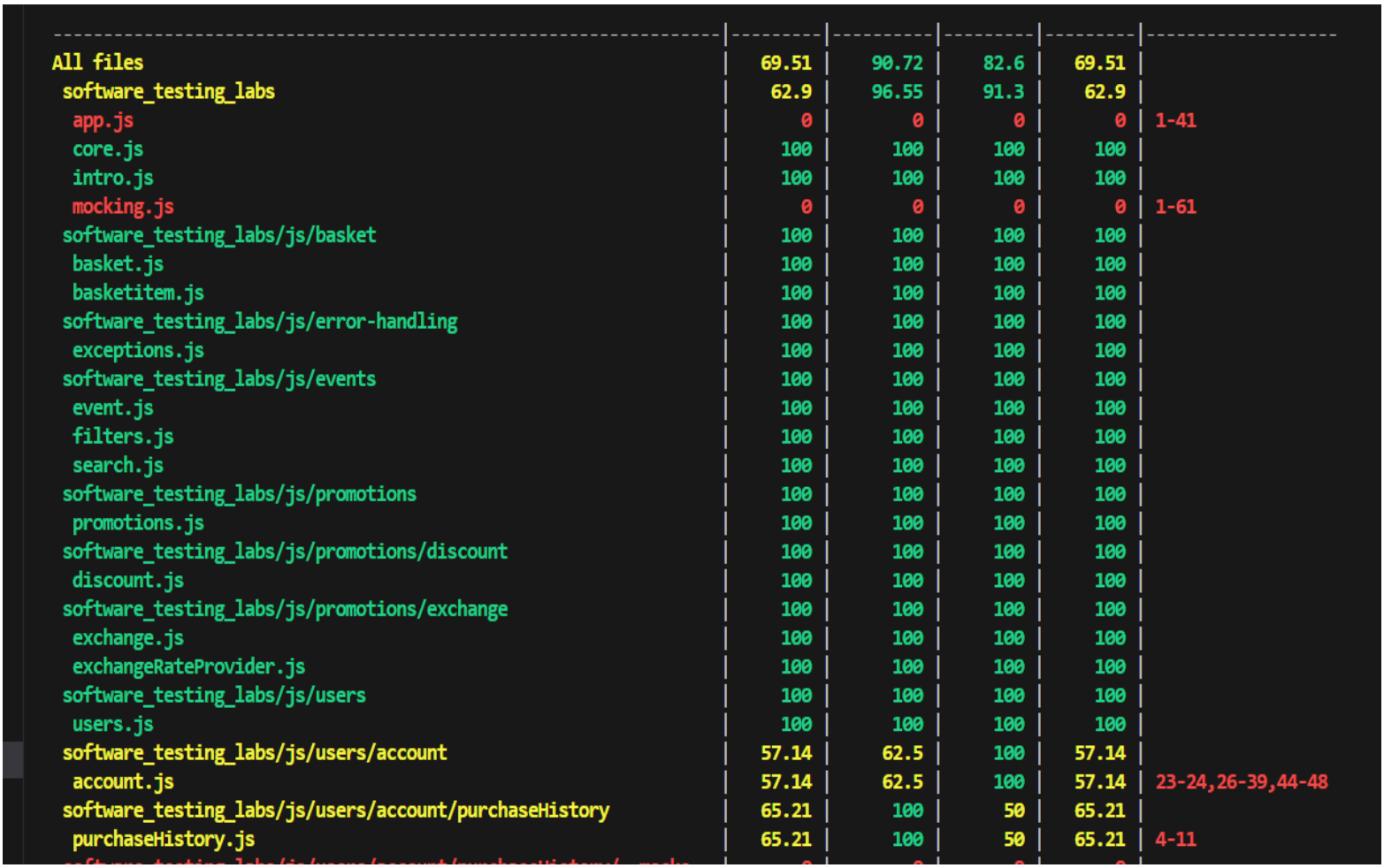
5. Exécution des tests

Nous implémentons les cas de test proprement dit, le tout se trouvant dans un dossier nommé **tests** du projet.

6. Rapports de test et analyse

En définitif, nous avons effectué au total exactement **115 tests réussis**, pour une couverture globale de :

- **100%** pour le dossier basket
- **100%** pour le dossier error-handling
- **100%** pour le dossier events
- **100%** pour le dossier promotions
- **100%** pour le fichier users.js du dossier users
- **57.14%** pour le fichier account.js du dossier account
- **65.21%** pour le fichier purchaseHistory.js
-



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	69.51	90.72	82.6	69.51	
software_testing_labs	62.9	96.55	91.3	62.9	
app.js	0	0	0	0	1-41
core.js	100	100	100	100	
intro.js	100	100	100	100	
mocking.js	0	0	0	0	1-61
software_testing_labs/js/basket	100	100	100	100	
basket.js	100	100	100	100	
basketitem.js	100	100	100	100	
software_testing_labs/js/error-handling	100	100	100	100	
exceptions.js	100	100	100	100	
software_testing_labs/js/events	100	100	100	100	
event.js	100	100	100	100	
filters.js	100	100	100	100	
search.js	100	100	100	100	
software_testing_labs/js/promotions	100	100	100	100	
promotions.js	100	100	100	100	
software_testing_labs/js/promotions/discount	100	100	100	100	
discount.js	100	100	100	100	
software_testing_labs/js/promotions/exchange	100	100	100	100	
exchange.js	100	100	100	100	
exchangeRateProvider.js	100	100	100	100	
software_testing_labs/js/users	100	100	100	100	
users.js	100	100	100	100	
software_testing_labs/js/users/account	57.14	62.5	100	57.14	
account.js	57.14	62.5	100	57.14	23-24,26-39,44-48
software_testing_labs/js/users/account/purchaseHistory	65.21	100	50	65.21	
purchaseHistory.js	65.21	100	50	65.21	4-11