REPUBLIQUE DU CAMEROUN

Paix – Travail – Patrie

-. - . - . –

UNIVERSITE DE YAOUNDE I

Faculté des Sciences

Département d'Informatique

B.P. 812 Yaoundé

REPUBLIC OF CAMEROON

Peace – Work – Fatherland

-. - . - . –

UNIVERSITY OF YAOUNDE I

Faculty of Sciences

Department of Computer Science

P.O. Box 812 Yaoundé

SOFTWARE UNIT TESTING PLAN: LOCAL RESTAURANT MOBILE APPLICATION

Prepared By:

NTABET SALOMON PIERRE

LIOMO SIMEU NAOMI

TIAGOU AZAMBOU YOLLANDE MATHILDE

Sunday, June 16, 2024

Supervised By:

Mr. Atemengue Regis

Dr. Kimbi Xaveria

Semester 4, 2023-2024

Table of Contents

1. Introduction

This document is a level-specific test plan for the mobile application of a local restaurant. The global purpose of this application is to enable all users to create a basket of products, to order takeaway or delivery, and to benefit from exclusives promotions.

1.1 Objectives

Based on the user and stakeholder requirements, our testing team was able to unleash the following objectives for testing:

- ✓ Make sure the Application Under Test (AUT) i.e. local Restaurant, satisfies the functional and non-functional preliminaries elucidated from requirements.
- ✓ Verify that the AUT lives up to the quality conditions described by the client.
- ✓ Bugs or defects are found and then fixed before it goes live.
- ✓ Do performance related evaluation of the application regarding speed, reactivity and resource utilization.
- ✓ There must be maximum code coverage of the application.
- ✓ Identify and correct any potential security holes.

2. Scope of Testing

This test plan focuses on testing all the core functionalities of the local restaurant application. The in-scope modules that are to be tested are:

- ➢ Basket
  - ▪ CalculateTotal function
  - ▪ ShowAdverts function
  - ▪ SearchBasket function
  - ▪ getBasketItem function
  - ▪ CreateBasketItem function
  - ▪ SerializeBasketItemToJson function
- ➢ Error-handling
  - ▪ InvalidEventNameError class
  - ▪ InvalidEventPriceError class

- InvalidReferralCodeError class
  - InvalidUsernameError class
  - UserHasAccountError class
- Events
  - isSoldOut function
  - getTagLine function
  - createEvent function
  - today function
  - next7Days function
  - next30Days function
  - getEvents function
- Promotions
  - CalculatePercentageDiscount
  - CalculateMoneyOff
  - generateReferralCode
  - applyDiscount asynchronous function
  - getExchangeRate asynchronous function
  - getDiscount asynchronous function
- Users
  - Purchase class
  - User class
  - Event class
  - userExists asynchronous function
  - createUserId function
  - isValidUserName asynchronouos function
  - createAccount asynchronous function
  - getPastPurchases function
  - parsePurchaseResponse function
  - getPurchaseHistory function
  - __getPurchaseHistory function

3. Testing Approach

3.1 Testing Methods and Techniques

Upon thorough analysis of the application with the resources and time at our disposal, not forgetting the users and stakeholders specifications, our testing team was able to choose two main approaches for this mobile application. These approaches are:

- White-box testing (precisely **Unit testing**) permitting us to do a thorough review of the internal workings of the software, precisely, the units of code, logic etc. The programming skills of our testing team are going to be on stake to help us imagine the different test cases.
- Automated testing which we will use to execute repetitive and time-consuming tests. In other words, the test cases we imagined are going to be executed automatically using dedicated software tools (Since the code is in JavaScript, the Vitest framework will used). It is very time safety and help us have a greater code coverage.
- Mocking and Stubbing techniques to isolate units of code from their dependencies.

3.2 Tools

- Vitest: A fast unit test framework for JavaScript that is compatible with Vite.

3.3 Roles and Responsibilities

- Test manager tasked with overseeing the overall operations and things to be done.
- Developers had to write and execute the unit tests.
- QA Engineers who had review test cases and ensure comprehensive coverage.

4. Testing Schedule

Phase	**Start Date**	**End Date**
Test Planning	15-06-2024	16-06-2024
Test Case Design	16-06-2024	17-06-2024
Test Execution	17-06-2024	18-06-2024
Test Reporting	18-06-2024	18-06-2024

5. Test Environment

- **Vscode:** Version 1.90.0
- **Nodejs:** Version 20.12.2
- **Vitest:** Version 1.6.0
- **Vite:** Version 5.0.x

6. Test Cases

Basket.js

ID	Description	Expected Result	Actual result
1	should calculate total price for multiple items without discount	475	475
2	should calculate total price for multiple items with discount	50	50
3	should calculate total price for a single item	100	100
4	should return 0 for an empty basket	0	0
5	should not show adverts for premium users	false	false
6	should find items in the basket matching the search query	basketItem1	basketItem1
7	should return an empty array if no items match the search query	Null	null
8	should create a new basket item if it does not exist in the basket	2	2
9	should return null if the basket item already exists	Null	null

Events.js

ID	Description	Expected Result	Actual result

1	should create a new event	id = 1
		name = Test Event
		ticketPrice = 50
		totalTickets = 100
		ticketsRemaining = 50
		date = 2023-06-17
			id = 1
			name = Test Event
			ticketPrice = 50
			totalTickets = 100
			ticketsRemaining = 50
			date = 2023-06-17

2	should return true if the event is sold out	true	true
3	should return false if the event is not sold out	false	false
4	should return "Event Sold Out!" if the event is sold out	Event Sold Out	Event Sold Out
5	should return "Hurry only X tickets left!" if the event has few tickets remaining	Hurry only 5 tickets left!	Hurry only 5 tickets left!
6	should return "This Event is getting a lot of interest. Don't miss out, purchase your ticket now!" if the event is popular	This Event is getting a lot of interest. Do not miss out purchase your ticket now!	This Event is getting a lot of interest. Do not miss out purchase your ticket now!
7	should return "Don't miss out, purchase your ticket now!" if the event is not sold out and not popular	Don't miss out, purchase your ticket now!	Don't miss out, purchase your ticket now!
8	should throw an InvalidEventNameError if the event name is invalid	InvalidEventNameError	InvalidEventNameError

9	should throw an InvalidEventPriceError if the event price is invalid	InvalidEventPriceError	InvalidEventPriceError
10	should throw an InvalidEventPriceError if the available tickets is invalid	InvalidEventPriceError	InvalidEventPriceError

Exchange.js

ID	Description	Expected Result	Actual result
1	should return the correct exchange rate for supported currencies	originalCurrency = GBP newCurrency = USD exchangeRate = 1.25	originalCurrency = GBP newCurrency = USD exchangeRate = 1.25
2	should throw an error for unsupported currencies	Currency not supported	Currency not supported

Promotions.test.js

ID	Description	Expected Result	Actual result
1	should apply percentage discount if currentPrice is above minimumSpend	135	135
2	should not apply percentage discount if currentPrice is below minimumSpend	90	90
3	should apply money off discount if currentPrice is above minimumSpend	130	130
4	should not apply money off discount if currentPrice is below minimumSpend	90	90
5	should generate a referral code with the userId	#FRIEND-#\\d{3}-#	#FRIEND-#\\d{3}-#

6	should apply MONEYOFF discount if discount code is valid	130	130
7	should apply PERCENTAGEOFF discount if discount code is valid	135	135
8	should return the original total if discount code is invalid	150	150

exceptions.test.js

ID	Description	Expected Result	Actual result
1	should create an instance with the provided error message	Invalid event name	Invalid event name
2	should create an instance with the provided error message	Invalid event price	Invalid event price
3	should create an instance with the provided error message	Invalid referral code	Invalid referral code
4	should create an instance with the provided error message	Invalid username	Invalid username
5	should create an instance with the provided error message	User already has an account	User already has an account

Filter.js

ID	Description	Expected Result	Actual result
1	should return true for an event happening today	true	true
2	should return false for an event not happening today	false	false
3	should return true for an event happening within the next 7 days	true	true

4	should return false for an event happening after the next 7 days	false	false
5	should return false for an event happening before today	false	false
6	should return true for an event happening within the next 30 days	true	true
7	should return false for an event happening before today	false	false

search.js

ID	Description	Expected Result	Actual result
1	should return all events when the search predicate is not provided	events	events
2	should return events that match the search predicate	events	events
3	should return an empty array when no events match the search predicate	empty array []	empty array []

discount.js

ID	Description	Expected Result	Actual result

1	should fetch discount data from the API	isValid = true,
		type = MONEYOFF,
		value = 20,
		minSpend = 100
			isValid = true,
			type = MONEYOFF,
			value = 20,
			minSpend = 100

2	should return error if API request fails	API request failed	API request failed

7. Test Automation

All unit tests will be automated using Vitest. Test scripts will be created and maintained by the development team. Automated tests will be executed as part of the continuous integration process.

8. Risks and Issues

- **Dependency Changes**: Changes in external dependencies may affect test outcomes.
    - **Mitigation**: Regularly update and review dependencies.
- **Incomplete Test Coverage**: Not all code paths may be tested.
    - **Mitigation**: Ensure comprehensive test case design and peer reviews.

9. Reporting and Communication

- **Reporting Frequency**: Daily during the test execution phase.
- **Report Format**: Test execution reports will be generated in JSON formats.
- **Stakeholders**: Development team, QA team, and project managers, lecturer, teacher.

10. Conclusion

This test plan provides a framework for conducting a comprehensive and rigorous testing phase for the local restaurant's mobile application. By following this plan, the team can ensure that the application meets quality requirements and provides a positive user experience.