|  |
| --- |
| CARRY1ST |
| ASSOCIATE QA ENGINEER | HANNAH OKYNE |
| QA ASSESSMENT ANSWERS |

|  |
| --- |
| 4-22-2024 |

**Section A:Test Case Writing, Exploratory Test and Defect Reporting.**

[**https://ecommerce-playground.lambdatest.io/**](https://ecommerce-playground.lambdatest.io/)

You are required to perform an exploratory test on the Web app and the Android App of a data collection system.

1. Question: A test plan and a test case for the following futures (features) \*

a. Signup

b. Login

c. Search

d. Add to cart

2. Question: Explore the sign up, login, search and cart functionalities and write down bugs and improvements

**ANSWERS**

**SECTION A QUESTION 1 AND 2 :**

**TEST PLAN**

Table of Contents

1 [Introduction 3](#_Toc109387551)

[1.1 Objectives 3](#_Toc109387552)

[2 Scope 3](#_Toc109387554)

[3 Assumptions / Risks 4](#_Toc109387555)

[3.1 Assumptions 4](#_Toc109387556)

[3.2 Risks 4](#_Toc109387557)

[4 Test APPROACH 4](#_Toc109387558)

[5 Test Environment 5](#_Toc109387560)

[6 Milestones / Deliverables 5](#_Toc109387561)

[6.1 **LINK TO TEST CASES EXCEL SHEET** 5](#_Toc109387563)

[6.2 DELIVERABLES 5](#_Toc109387562)

[6.3 Test Schedule 5](#_Toc109387563)

# 

# Introduction

The purpose of this document is to outline the test plan for the exploratory testing of the web app and Android app of the data collection system. The testing aims to ensure the functionality and usability of the signup, login, search, and cart features. It includes the objectives, scope risks and approach. This document will clearly identify what the test deliverables will be.

## 1.1 Objectives

- To identify defects or issues within the signup, login, search, and cart functionalities.

- To provide recommendations for improvements to enhance the user experience.

- To ensure compatibility across different browsers and devices.

# 2. Scope

The testing will cover the following features of both the web app and Android app:

- Signup

- Login

- Search

- Adding items to the cart

# 3. Assumptions / Risks

## 3.1 Assumptions

* The test environment accurately represents the production environment.
* The provided web app and Android app are the latest versions available.

## 3.2 Risks

## - Potential risks include compatibility issues across different browsers and devices.

## 4. Test Approach

* The project is using an agile approach.
* Exploratory testing will play a large part of the testing as this is part of an assessment .
* Compatibility testing across devices
* Usability testing to ensure intuitive user experience
* Automated unit tests are part of the development process, but no automated functional tests are planned at this time.

# 5. Test Environment

**Web App:**

* Browser: Google Chrome
* Operating System: Windows 10

**Android App:**

* Device: Samsung A13
* Operating System: Android 13.

# 6. Milestones / Deliverables

## 6.1 Test Cases Link

LINK TO  **(Excel document submitted)**

## 6.2 Deliverables

|  |  |
| --- | --- |
| **Deliverable** | **Milestone** |
| Test Plan document | 1 Day |
| Test Cases Creation | 1 Day |
| Bug Report | 1 Day |

## 

## 6.3 Test Schedule

Based on the instructions provided in this assignment .

The test schedule will be as follows :

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Comments** |
| Test Planning and Environment Set up | - | Day 1 | Completed |
| Exploratory testing of signup and login functionalities | - | Day 2 | Completed |
| Exploratory testing of search and cart functionalities | - | Day 3 | Completed |
| Bug triaging, documentation, and reporting | - | Day 3 | Completed |

**QUESTION 2 : BUGS AND IMPROVEMENTS**

**BUGS :**

* Confirmation email is not sent to the user after successful registration.
* User does not receive password reset links and is unable to reset the password successfully.
* Same Products displayed multiple times in the “top products” section on the website
* Selected category displays incorrect items in the Search Bar
* Inconsistent text styles throughout the webpages
* Inconsistent image display and styles throughout website
* Content on blog page not displayed in English
* Item search results are not displayed correctly.

**IMPROVEMENT SUGGESTIONS :**

Sign Up:

* .
* Provide real-time validation feedback for input fields during the sign-up process, highlighting any errors or missing information as users fill out the form, to prevent form submission errors.
* Offer a seamless transition from the sign-up page to onboarding tutorials or guides, helping new users familiarize themselves with the platform's features and functionalities.
* Improve password strength indicator for better user guidance.
* Incorporate social media sign-up options (e.g., Sign up with Google, Sign up with Facebook) to offer users an alternative and faster registration method, leveraging their existing accounts

Login:

* Implement "Remember Me" functionality for convenience in subsequent logins.
* Implement two-factor authentication (2FA) as an optional security feature for user accounts, adding an extra layer of protection against unauthorized access.

Search:

* Integrate autocomplete suggestions in the search bar to assist users in formulating their queries more efficiently, based on popular or relevant search terms.
* Implement natural language processing (NLP) capabilities to interpret and process complex search queries, enabling the system to understand user intent and return more accurate search results.
* Enable users to save their search history or create custom search filters for quick access to frequently used search parameters, enhancing user productivity and customization options.

Add to Cart

* Provide advanced search filters (e.g., by category, price range) for better user experience.
* Implement intuitive feedback upon adding items to the cart, such as a confirmation message or animation, to reassure users that their action was successful.
* Enhance the visibility of the cart icon or button to provide users with easy access to view their cart contents at any time.
* Introduce a "Continue Shopping" option after adding items to the cart, allowing users to seamlessly return to browsing without interrupting their shopping flow. Ensure that items remain in the cart for a reasonable duration, even after users navigate away from the page, to prevent frustration and facilitate smoother checkout processes.

**Section B**

**Game Testing Concepts:**

1. **Define game testing and explain its importance in the development process.**

Game testing is the systematic process of evaluating a game's functionality, performance, usability, and overall quality to ensure it meets the desired standards before release. It involves identifying and documenting defects, ensuring gameplay mechanics work as intended, and verifying that the game performs well across different platforms and devices.

The importance of game testing in the development process cannot be overstated. It serves as a crucial quality assurance measure, helping to identify and rectify issues that could negatively impact the player experience. By conducting thorough testing, developers can catch bugs early, refine gameplay mechanics, optimize performance, and ultimately deliver a polished product that meets player expectations. Moreover, game testing fosters confidence among stakeholders and enhances the reputation of the game and its development team.

1. **How would you approach compatibility testing for a mobile game across different devices and operating systems?**

To ensure compatibility across various devices and operating systems for a mobile game, I would adopt a comprehensive testing approach that includes the following steps:

**Device Selection:** Identify a diverse range of devices representing different screen sizes, resolutions, hardware capabilities, and operating systems commonly used by the target audience.

**Test Plan Creation:** Develop a detailed test plan outlining specific test scenarios and criteria for evaluating compatibility, including functionality, performance, and user experience.

**Execution:** Conduct tests on each selected device, systematically evaluating the game's performance and functionality across different configurations. This includes verifying graphical fidelity, control responsiveness, audio quality, and any device-specific features or limitations.

**Regression Testing**: Perform regression testing on previously tested devices to ensure updates or fixes have not introduced new compatibility issues.

**Automated Testing:** Implement automated testing frameworks where necessary to streamline the compatibility testing process, especially for repetitive tasks and regression testing.

**User Feedback Integration:** Gather feedback from real users across various devices and operating systems through beta testing or early access programs to identify compatibility issues that may not have been detected through traditional testing methods.

By following this approach, I aim to ensure that the mobile game delivers a consistent and enjoyable experience across a wide range of devices and platforms, thereby maximizing its reach and appeal to players.

1. **Provide an example of a challenging bug you discovered during mobile game testing and how you approached reporting it.**

During testing a mobile game, I encountered a particularly elusive bug that caused the game to crash randomly during certain levels. To tackle this challenge, I meticulously documented the steps leading up to each crash, including any patterns or commonalities among them. Additionally, I gathered device information such as model, operating system version, and available memory to provide comprehensive context for the developers.

For example, in one instance, the game consistently crashed when a specific power-up was activated while the player character was in motion. By replicating this scenario across multiple devices and environments, I confirmed it was a reproducible issue rather than an isolated incident.

Once I had compiled all pertinent details, I drafted a detailed bug report. This report included a clear description of the problem, steps to reproduce the issue, device information, and any relevant screenshots or videos. I also categorized the bug based on severity and provided suggestions for potential fixes or workarounds.

By approaching the reporting process with thoroughness and clarity, I ensured that the development team had all the information they needed to address the bug effectively, ultimately leading to its resolution and improving the overall quality of the mobile game.

During mobile game testing, I encountered a challenging bug related to inconsistent touch controls in a fast-paced action game. Players reported that certain swipe gestures were not consistently registering, leading to frustration and decreased gameplay enjoyment.

To address this issue, I followed these steps:

**Reproduction:** Carefully replicated the reported issue on multiple devices with different screen sizes and operating systems to understand its scope and severity.

Isolation: Identified potential factors contributing to the touch control inconsistency, such as device-specific optimizations.

**Documentation:** Thoroughly documented the steps to reproduce the bug, including device specifications, game settings, and specific touch gestures that were affected.

**Communication**: Communicated the findings to the development team, providing detailed insights into the nature of the bug and its potential impact on the player experience.

**Validation:** Conducted rigorous testing to validate the effectiveness of proposed solutions and ensure that the touch controls were consistent and responsive across all supported devices.

**Reporting:** Prepared a comprehensive bug report summarizing the issue, the steps taken to address it, and the final resolution implemented in the game's codebase.

By following this approach, I was able to effectively address the challenging touch control bug, ultimately improving the gameplay experience for players and enhancing the overall quality of the mobile game.

1. **Imagine you encounter a game-breaking bug in a mobile app game during testing. How would you approach troubleshooting and resolving this issue?**

Encountering a game-breaking bug during testing can be a critical challenge, but I would approach it methodically and efficiently using the following steps:

**Immediate Documentation:** Document the issue promptly, including specific details such as the steps to reproduce, affected game features, and any error messages or crash logs.

**Isolation:** Attempt to isolate the root cause of the game-breaking bug by analyzing recent code changes, game assets, or server configurations that may have introduced the issue.

**Severity Assessment:** Evaluate the severity of the bug and its potential impact on the player experience, considering factors such as gameplay progression, usability, and financial implications.

**Temporary Workaround:** If possible, implement a temporary workaround to mitigate the immediate impact of the bug and allow testing to continue while a permanent solution is developed.

**Collaboration:** Collaborate closely with developers, designers, and other stakeholders to prioritize the resolution of the game-breaking bug and brainstorm potential solutions.

**Testing:** Rigorously test proposed solutions in a controlled environment to verify their effectiveness and ensure they do not introduce new issues or regressions.

Communication: Maintain transparent communication with project stakeholders, providing regular updates on the troubleshooting process, proposed solutions, and estimated timelines for resolution.

**Validation:** Once a solution has been implemented, conduct thorough validation testing to confirm that the game-breaking bug has been successfully resolved and that the affected game features are functioning as intended.

**Post-Mortem Analysis:** Conduct a post-mortem analysis to review the root cause of the game-breaking bug, identify any process or workflow improvements that could prevent similar issues in the future, and capture lessons learned for continuous improvement.

By following this troubleshooting approach, I aim to efficiently address the game-breaking bug, minimize its impact on the development timeline, and ensure the timely delivery of a stable and high-quality mobile game.

1. **Given a scenario of a simple mobile game Mancala Adventures (**[**Android**](https://play.google.com/store/apps/details?id=com.qenegames.gebeta&hl=en&gl=US) **or** [**iOS**](https://apps.apple.com/gb/app/mancala-adventures-board-games/id1518590369)**), create test cases covering basic functionality and edge cases.**

**Test Case Type : Basic Gameplay Functionality**

**Description(s):**

* Verify that the game starts with the correct initial board configuration.
* Confirm that players can select a pit and distribute its seeds in a counter-clockwise direction.
* Ensure that the game accurately updates the number of seeds in each pit after each move.
* Validate that players cannot make an invalid move (e.g., selecting an empty pit or opponent's pit).
* Ensure matches synchronize correctly between devices.
* Test for any issues with disconnects, reconnections, or synchronization errors.
* Verify that players can communicate with each other via in-game chat or messaging.

**Test Case Type : Controls**

**Description(s):**

* Test touch controls on both Android and iOS devices to ensure responsiveness.
* Check for consistency in controls across different screen sizes and orientations.
* Test for any lag or delay in responding to user input.

**Test Case Type : User Interface Testing**

**Description(s):**

* Test the layout and design of the game interface for clarity and ease of use.
* Ensure that all game elements such as pits, stones, and score counters are displayed correctly.
* Verify that the user interface adapts well to different screen sizes and resolutions.
* Test for accessibility features such as text size adjustment and color contrast.

**Test Case Type : Winning Conditions**

**Description(s):**

* Test for the correct determination of game end conditions, such as when one player has no more valid moves.
* Confirm that the game correctly identifies the winner based on the number of seeds captured.

**Test Case Type :** Performance Testing:

**Description(s):**

* Test the game's performance on devices with varying hardware specifications.
* Check for memory leaks or excessive resource usage during extended gameplay sessions.
* Test loading times for different game modes and levels.
* Verify that the game runs smoothly without crashes or freezes under normal and stress conditions.

**Test Case Type : AI Opponent Behavior**

**Description(s):**

* Evaluate the behavior of AI opponents across different difficulty levels (easy, medium, hard).
* Verify that AI opponents make logical moves based on the current board state and strategic considerations.

**Test Case Type : UI/UX Testing**

**Description(s):**

* Test the responsiveness of touch controls on various screen sizes and resolutions.
* Verify that the game's interface elements (buttons, menus, etc.) are properly displayed and functional on both portrait and landscape orientations.
* Test the layout and design of the game interface for clarity and ease of use.
* Ensure that all game elements such as pits, stones, and score counters are displayed correctly.
* Verify that the user interface adapts well to different screen sizes and resolutions.
* Test for accessibility features such as text size adjustment and color contrast.

**Test Case Type : Edge Cases**

**Description(s):**

* Test scenarios where the game board is nearly empty or completely full to ensure correct handling of edge cases.
* Validate the game's behavior when encountering unusual player

1. **Explain 5 common game mechanics and elements (e.g., gameplay, controls, user interface) and how they would test these aspects.**

**Gameplay Mechanics:**

Gameplay mechanics refer to the rules and interactions that govern how players engage with the game world. Testing gameplay mechanics involves ensuring that they function as intended and provide an engaging and balanced experience.

**Testing Approach**: Develop test cases to verify each gameplay mechanic individually and in combination with others. For example, test combat mechanics in a role-playing game to ensure attacks, defenses, and status effects work correctly and create enjoyable gameplay dynamics.

**Controls:**

Controls dictate how players interact with the game, including movement, actions, and navigation. Testing controls is crucial to ensure they are responsive, intuitive, and compatible with various input devices.

**Testing Approach:** Conduct usability testing to evaluate the intuitiveness of controls across different input methods (e.g., keyboard, mouse, touch screen, gamepad). Verify that control mappings are customizable and that players can perform complex maneuvers accurately and efficiently.

**User Interface (UI):**

The user interface encompasses all visual and interactive elements through which players interact with the game, including menus, HUD elements, and on-screen prompts. Testing the UI ensures it is visually appealing, functional, and accessible to players of all skill levels.

**Testing Approach:** Perform UI testing to validate layout consistency, font readability, color contrast, and button responsiveness across various screen resolutions and aspect ratios. Conduct focus group testing to gather feedback on the intuitiveness of menu navigation and the clarity of on-screen instructions.

**Artificial Intelligence (AI):**

AI governs the behavior of non-player characters (NPCs) and enemies within the game world, including pathfinding, decision-making, and adaptive behaviors. Testing AI ensures that NPCs behave realistically, challenge players appropriately, and contribute to immersive gameplay experiences.

**Testing Approach:** Create test scenarios to evaluate AI behavior under different conditions, such as combat encounters, puzzle-solving, or environmental interactions. Use AI debugging tools to monitor NPC pathfinding algorithms, decision trees, and response times to player actions.

**Progression Systems:**

Progression systems reward players for their achievements and motivate continued engagement with the game. This includes experience points, leveling up, unlocking new abilities, and acquiring in-game rewards. Testing progression systems ensures they provide meaningful incentives and a sense of accomplishment.

**Testing Approach:** Design test cases to verify that progression systems are balanced and paced appropriately throughout the game. Monitor player progression data to identify potential bottlenecks or exploits that could disrupt the intended gameplay experience. Conduct A/B testing to compare the effectiveness of different reward structures and pacing strategies.

**Section C: API Automation (Any of Postman, REST Assured, Python, or Cypress):**

**ANSWERS : USING REST Assured IN JAVA**

1. **Question**: https://jsonplaceholder.typicode.com/

Requirements:

* Write a method to fetch all posts.
* Write a method to fetch app post made with userId 1 and return the number of posts made by the userId

import io.restassured.RestAssured;

import io.restassured.response.Response;

public class JsonPlaceholderAPI {

public static void main(String[] args) {

fetchAllPosts();

countUserPosts(1);

}

public static void fetchAllPosts() {

Response response = RestAssured.get("https://jsonplaceholder.typicode.com/posts");

System.out.println("All posts:");

System.out.println(response.asString());

}

public static void countUserPosts(int userId) {

Response response = RestAssured.get("https://jsonplaceholder.typicode.com/posts?userId=" + userId);

System.out.println("Number of posts made by user " + userId + ": " + response.jsonPath().getList("").size());

}

}

1. **Question**: https://jsonplaceholder.typicode.com/

Requirements:

* Write a method to fetch all users.
* Using a random user id, write a method to print a user’s name, email, and address to console
* Using this userID, get this user’s associated post

import io.restassured.RestAssured;

import io.restassured.response.Response;

public class JsonPlaceholderAPI {

public static void main(String[] args) {

fetchAllUsers();

printUserInfo(3); // Replace 3 with any random user id

fetchUserPosts(3); // Replace 3 with any random user id

}

public static void fetchAllUsers() {

Response response = RestAssured.get("https://jsonplaceholder.typicode.com/users");

System.out.println("All users:");

System.out.println(response.asString());

}

public static void printUserInfo(int userId) {

Response response = RestAssured.get("https://jsonplaceholder.typicode.com/users/" + userId);

System.out.println("User info:");

System.out.println("Name: " + response.jsonPath().getString("name"));

System.out.println("Email: " + response.jsonPath().getString("email"));

System.out.println("Address: " + response.jsonPath().get("address").toString());

}

public static void fetchUserPosts(int userId) {

Response response = RestAssured.get("https://jsonplaceholder.typicode.com/posts?userId=" + userId);

System.out.println("User's posts:");

System.out.println(response.asString());

}

}

1. **Question**: https://restful-booker.herokuapp.com/apidoc/index.html

Requirements:

* Create a auth Token.
* Get all bookingIDs
* Create a Booking
* Update a booking

import io.restassured.RestAssured;

import io.restassured.response.Response;

public class RestfulBookerAPI {

public static void main(String[] args) {

String token = createAuthToken("admin", "password");

System.out.println("Auth token: " + token);

getAllBookingIDs(token);

// Example data for creating a booking

String bookingData = "{\"firstname\": \"John\", \"lastname\": \"Doe\", \"totalprice\": 200, \"depositpaid\": true, \"additionalneeds\": \"Breakfast\"}";

String newBookingId = createBooking(token, bookingData);

System.out.println("New booking ID: " + newBookingId);

// Example data for updating a booking

String updateData = "{\"totalprice\": 300, \"depositpaid\": false}";

updateBooking(token, newBookingId, updateData);

}

public static String createAuthToken(String username, String password) {

Response response = RestAssured.given()

.contentType("application/json")

.body("{\"username\": \"" + username + "\", \"password\": \"" + password + "\"}")

.post("https://restful-booker.herokuapp.com/auth");

return response.jsonPath().getString("token");

}

public static void getAllBookingIDs(String token) {

Response response = RestAssured.given()

.header("Authorization", "token " + token)

.get("https://restful-booker.herokuapp.com/booking");

System.out.println("All booking IDs:");

System.out.println(response.jsonPath().getList("bookingid"));

}

public static String createBooking(String token, String bookingData) {

Response response = RestAssured.given()

.header("Authorization", "token " + token)

.contentType("application/json")

.body(bookingData)

.post("https://restful-booker.herokuapp.com/booking");

return response.jsonPath().getString("bookingid");

}

public static void updateBooking(String token, String bookingId, String updateData) {

Response response = RestAssured.given()

.header("Authorization", "token " + token)

.contentType("application/json")

.body(updateData)

.put("https://restful-booker.herokuapp.com/booking/" + bookingId);

System.out.println("Updated booking:");

System.out.println(response.asString());

}

}