TEST PLAN

PRADA WEB APPLICATION

Prepared by: Buhaiov Viktor

TABLE OF CONTENTS

1. INTRODUCTION

Customer needs a perfect website, which has passed the full cycle of manual and automation testing. Given the specificity of the site it is very important to have the same quality of the site.

The Test Plan has been created to facilitate communication within the team members. This document describes approaches and methodologies that will apply to the unit, integration and system testing of the https://www.prada.com/. It includes the objectives, test responsibilities, entry and exit criteria, scope, schedule major milestones, entry and exit criteria and approach. This document has clearly identified what the test deliverables will be, and what is deemed in and out of scope.

2. ROLES AND RESPONSIBILITIES

Role	Name	Responsibilities
Scrum Master	Sergey Efremov	1. Acts as a primary contact for development and QA team. 2. Responsible for Project schedule and the overall success of the project. 3.Setting new tasks and reviewing bugs.
QA	Buhaiov Viktor	1. Test plan creation . 2. Test cases creation. 3. Manual test cases executing. 4. Test cases automating. 5. Reporting and tracking defects. 6. Coordinate with QA Lead for any issues or problems.

3. SCOPE

3.1. IN SCOPE

- User authentication.
- Product browsing and search.
- Shopping cart and checkout process.
- Account management.
- Responsive design across devices.
- Performance testing.
- Security testing.

3.2. OUT OF SCOPE

- Appointment management.
- Search usability.
- Pradasphere usability.
- Loading testing.

4. TEST APPROACH

Analytical test approach was used in accordance to requirements - based strategy. An analysis of the requirements specification form is the basis for planning, estimating and designing tests. Test cases will be created during exploratory testing. All test types are determined in Test Strategy.

The project is using an agile approach, with weekly iterations. At the end of each week the requirements identified for that iteration will be delivered to the team and will be tested.

5. ENTRY AND EXIT CRITERIA

5.1. ENTRY CRITERIA

- Proper test data should be available.
- All the necessary documentation, design, and requirements information should be available that will allow testers to operate the system and judge the correct behaviour.
- All the standard software tools including the testing tools must have been successfully installed and functioning properly.
- All test hardware platforms must have been successfully installed, configured, and functioning properly.

5.2. EXIT CRITERIA

- No high priority or severe bugs are left outstanding.
- A certain level of requirements coverage has been achieved.
- All high-risk areas have been fully tested, with only minor residual risks left outstanding.
- Cost – when the budget has been spent.
- The schedule has been achieved.

6. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

6.1. SUSPENSION CRITERIA

- Software/Hardware problems.
- Significant change in requirements suggested by client.
- The build contains many serious defects which seriously or limit testing progress.
- Assigned resources are not available when needed by the test team.

6.2. RESUMPTION CRITERIA

- Resumption will only occur when the problem that caused the suspension has been resolved.

7. TEST STRATEGY

7.1. QA ROLE IN TESTING PROCESS

- Understanding Requirements.
- Requirement specifications will be sent by client.
- Preparing Test Cases:

    - QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.
- Preparing Test Matrix:
    - QA will be preparing a test matrix which maps test cases to respective requirements. This will ensure the coverage for requirements.
- Reviewing test cases and matrix:
    - Review for test cases and test matrix will be conducted by QA Lead.
    - Any comments or suggestions on test cases and test coverage will be provided by the reviewer.
    - Suggestions or improvements will be updated by the preparer and sent to QA Lead for approval.
    - Updates and improvements will be reviewed and approved by the reviewer.
- Creating Test Data:
    - Test data will be created by respective QA based on scenarios and Test cases.
- Executing Test Cases:
    - Test cases will be executed by respective QA based on designed scenarios, test cases and Test data.
    - Test result (Actual Result, Pass/Fail) will be updated in test case document Defect Logging and Reporting: QA will be logging the defect/bugs in Google spreadsheet and JIRA, found during execution of test cases.
- Retesting and Regression Testing:
    - Retesting for fixed bugs will be done by respective QA once issue is resolved by respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.
- Deployment/Delivery:
    - Once all bugs/defects reported after complete testing is fixed and no other bugs are found, a report will be deployed to the client, along with sample output by email to respective lead and Report group Bug life cycle.
    - All the issues found while testing will be logged into JIRA.

7.2. TESTING TYPES

GUI Testing – GUI testing will include testing of the UI part of the report. It covers users Report format, look and feel, error messages, spelling mistakes, GUI guideline violations.

Exploratory testing – includes a type of software testing where Test cases are not created in advance but QA checks system “hands-on”. QA may note down ideas about what to test before test execution.

ADHOC testing – includes an informal testing type with an aim to break the system.

Positive testing – includes the type of testing that can be performed on the system by providing the valid data as input. It checks whether an application behaves as expected with positive inputs.

Negative testing – also known as failure testing or error path testing, is a method of testing an application or system that ensures that the plot of the application is according to the requirements and can handle the unwanted input and user behaviour. Invalid data is inserted to compare the output against the given input.

Boundary testing – testing technique that focuses on testing the boundaries or limits of input values. It aims to uncover any issues or vulnerabilities that may occur at the edges of the input domain, such as minimum and maximum values, as well as values near the boundaries.

7.3. BUG SEVERITY AND PRIORITY DEFINITION

7.3.1. SEVERITY LIST

Severity	Description	Exemples
Critical	The bug or defect will be significant enough to negatively impact the overall functionality of the app, resulting in a complete system failure.	System crashes, data loss, security vulnerabilities.
Major	The bug or defect will impact a large portion of the app's features and functionality.	Major features not working as expected, significant performance issues.
Minor	The bug or defect may interrupt the app, causing it to behave in a less than optimal way, but it will not disrupt the full functionality of the app.	Functional issues that do not require immediate attention, usability problems.
Low	The bug or defect will not significantly impact the overall functionality of the app.	Typos, visual inconsistencies, minor UI glitches.

7.3.2. PRIORITY LIST

Priority	Description	Exemples
Highest	Requires immediate attention. Critical bugs have a severe impact on the system, leading to a complete loss of service, system crash, or critical failure.	Crashes, data corruption, security risks.
High	High priority bugs must be addressed immediately. They often affect the app's functionality and impact customers and their user experience. They must take priority.	Key features malfunctioning, widespread performance issues.
Medium	Typically, medium-priority bugs do not affect customers directly and therefore can be fixed in the normal course of testing and development.	Minor feature issues, UI/UX problems.
Low	The bug does not have to be fixed immediately. High and medium priority bugs should be addressed first.	Cosmetic flaws, typos.

8. RESOURCE AND ENVIRONMENT NEEDS

8.1. TESTING TOOLS

Process	Tools
Test cases creation	Google documents, Google spreadsheets, Jira
Test cases tracking	Jira
Test case execution	Selenium WebDriver, Playwright, Postman API, BrowserStack, Mozilla Observatory, Lighthouse
Test case management	Google spreadsheets, Jira
Test reporting	Google spreadsheets, Jira

8.2. TEST ENVIRONMENTS

Operating Systems	Browsers
Windows 10 (1920 x 1080)	Edge, Google Chrome, Firefox
Windows 10 (2560 x 1440)	Edge, Google Chrome, Firefox
Windows 11 (1920 x 1080)	Edge, Google Chrome, Firefox
Windows 11 (2560 x 1440)	Edge, Google Chrome, Firefox
OS X Sonoma (1920 x 1080)	Google Chrome
OS X Sonoma (2560 x 1440)	Google Chrome

9. APPROVALS

Role	Name	Date	Signature
Scrum master	Sergey Efremov		
QA	Buhaiov Viktor		