Test Plan (VWO.com)

| Test Plan | 2 |
| --- | --- |
| Objective | 2 |
| Scope | 2 |
| Inclusion | 2 |
| Test Environments | 2 |
| Defect Reporting Procedure | 3 |
| Test Strategy | 3 |
| Test Schedule | 4 |
| Test Deliverables. | 4 |
| Entry and Exit Criteria | 4 |
| Test Execution | 5 |
| Test Closure | 5 |
| Tools | 6 |
| Risks and Mitigations | 6 |
| Approvals | 7 |

# 

# 

# 

# **Test Plan**

## **1. Objective**

The objective of this test plan is to ensure that the VWO Application meets the specified requirements, functions as intended, and is free of critical defects before release. This plan outlines the testing approach, resources, schedule, and deliverables to achieve this goal.

## **2. Scope**

This test plan covers the following:

* Functional testing of [specific modules/features].
* Non-functional testing (performance, usability, security, etc.).
* Regression testing to ensure no existing functionality is broken.
* Testing will be conducted on [platforms/devices/browsers].

**Exclusions:**

* Testing of third-party integrations (if applicable).
* Testing of hardware components (if applicable).

## **3. Inclusions**

The following testing types are included:

* **Functional Testing**: Validate all features against requirements.
* **Regression Testing**: Ensure new changes do not break existing functionality.
* **Performance Testing**: Evaluate system responsiveness and stability under load.
* **Security Testing**: Identify vulnerabilities and ensure data protection.
* **Usability Testing**: Assess user experience and interface design.

## **4. Test Environments**

* **Development Environment**: dev.vwo.com
* **Staging Environment**: qa.vw0.com
* **Production Environment**: app.vwo.com
* **Browsers/Devices**: [List of browsers/devices to be tested, e.g., Chrome, Firefox, Safari, iOS, Android].

## **5. Defect Reporting Procedure**

* Defects will be logged in [Defect Tracking Tool, e.g., Jira, Bugzilla].
* Each defect will include:
  + Defect ID
  + Description
  + Steps to reproduce
  + Expected vs. Actual results
  + Severity (Critical, High, Medium, Low)
  + Priority (P1, P2, P3)
  + Screenshots/Logs (if applicable)
* Defects will be assigned to the respective developer for resolution and retested after fixes.

## **6. Test Strategy**

## **Step 1: Test Scenario and Test Case Creation**

The first step in our testing process is to create detailed **test scenarios** and **test cases** for all features in scope. These test cases will be designed to validate the functionality, performance, and usability of the application.

### **Test Design Techniques**

To ensure comprehensive test coverage, we will use the following test design techniques:

* **Equivalence Class Partitioning**: To divide input data into valid and invalid classes.
* **Boundary Value Analysis**: To test the boundaries of input ranges.
* **Decision Table Testing**: To test combinations of inputs and their corresponding outputs.
* **State Transition Testing**: To validate behavior when the system transitions between states.
* **Use Case Testing**: To ensure the application meets user requirements and workflows.

### **Additional Test Case Creation Approaches**

We will also leverage our expertise to create test cases by applying:

* **Error Guessing**: Anticipating potential errors based on experience.
* **Exploratory Testing**: Unscripted testing to uncover unexpected issues.

### **Test Case Prioritization**

Test cases will be prioritized based on:

* Business impact and criticality of the feature.
* Risk of failure.
* Frequency of use.

## **Step 2: Testing Procedure**

### **Smoke Testing**

* Upon receiving a build, we will first conduct **Smoke Testing** to verify that the core functionalities of the application are working as expected.
* If the build fails Smoke Testing, it will be rejected, and we will wait for a stable build before proceeding with further testing.

### **In-Depth Testing**

* Once a stable build passes Smoke Testing, we will perform **in-depth testing** using the test cases created in Step 1.
* Multiple test resources will execute test cases simultaneously across **multiple supported environments** (e.g., browsers, devices, operating systems).

**Types of Testing**

The following types of testing will be performed:

* **Smoke Testing and Sanity Testing**: To ensure the build is stable and ready for further testing.
* **Regression Testing**: To ensure new changes do not break existing functionality.
* **Retesting**: To verify that fixed defects are resolved.
* **Usability Testing**: To evaluate the user experience and interface design.
* **Functionality & UI Testing**: To validate that the application works as intended and meets design specifications.

### **Test Cycles**

* Testing will be conducted in multiple cycles until the application meets the desired quality standards.
* Each cycle will include regression testing to ensure no new defects are introduced.

## **Step 3: Best Practices for Effective Testing**

### **Context-Driven Testing**

* Testing will be tailored to the specific context of the application, focusing on real-world usage scenarios and user needs.

### **Shift Left Testing**

* Testing activities will begin early in the development lifecycle, allowing defects to be identified and resolved sooner.
* Collaboration between developers and testers will be emphasized to ensure quality from the start.

### **Exploratory Testing**

* In addition to scripted test cases, testers will perform **exploratory testing** to uncover issues that may not be covered by predefined test cases.
* This approach leverages the testers' expertise and intuition to identify potential risks.

### **End-to-End Flow Testing**

* We will test **end-to-end user flows** that involve multiple functionalities to simulate real-world usage.
* This ensures that the application works seamlessly across all integrated components.

## **Step 4: Continuous Improvement**

* **Lessons Learned**: After each release, we will conduct a retrospective to identify areas for improvement in the testing process.
* **Automation**: Where applicable, we will automate repetitive test cases to improve efficiency and coverage.
* **Feedback Loop**: Regular feedback from stakeholders and end-users will be incorporated into the testing strategy to ensure alignment with business goals.

## **7. Test Schedule**

| **Task** | **Start Date** | **End Date** | **Responsible Team** |
| --- | --- | --- | --- |
| Test Plan Preparation | March 5 , 2025 | March 7, 2025 | QA Team |
| Test Case Development | March 8 , 2025 | March 10 2025 | QA Team |
| Test Execution | March 8 , 2025 | March 12 , 2025 | QA Team |
| Defect Reporting | March 13 2025 | March 15 2025 | QA Team |
| Retesting | March 15 2025 | March 17 2025 | QA Team |
| Test Closure | March 17 2025 | March 18 2025 | QA Team |

## 

## **8. Test Deliverables**

* Test Plan Document
* Test Cases and Test Scripts
* Test Data
* Defect Reports
* Test Execution Report
* Test Summary Report

## **9. Entry and Exit Criteria**

### **Entry Criteria:**

* Requirements and design documents are approved.
* Test environment is set up and ready.
* Test data is prepared.
* Test cases are reviewed and approved.

### **Exit Criteria:**

* All planned test cases are executed.
* All critical and high-severity defects are resolved.
* Test summary report is prepared and approved.

## 

## **10. Test Execution**

### **Entry Criteria:**

* Test environment is stable and available.
* Test cases are ready for execution.
* Test data is prepared.

### **Exit Criteria:**

* All test cases are executed.
* Defects are logged and tracked.

## **11. Test Closure**

### **Entry Criteria:**

* All test cycles are completed.
* Defect reports are finalized.

### **Exit Criteria:**

* Test summary report is approved.
* Test artifacts are archived.

## **12. Tools**

* **Test Management**: [e.g., TestRail, Zephyr]
* **Defect Tracking**: [e.g., Jira, Bugzilla]
* **Automation Testing**: [e.g., Selenium, Appium]
* **Performance Testing**: [e.g., JMeter, LoadRunner]
* **Security Testing**: [e.g., OWASP ZAP, Burp Suite]

## 

## **13. Risks and Mitigations**

| **Risk** | **Mitigation** |
| --- | --- |
| Delays in test environment setup | Coordinate early with the infrastructure team. |
| Incomplete requirements | Conduct regular requirement review meetings. |
| Resource unavailability | Maintain a backup resource plan. |
| Defect leakage to production | Conduct thorough regression testing and peer reviews. |

## 

## 

## **14. Approvals**

The testing team will prepare and share the following documents with the client for review and approval at various stages of the testing process:

* **Test Plan**: The overall strategy, scope, and approach for testing.
* **Test Scenarios**: High-level scenarios that outline the functionality to be tested.
* **Test Cases**: Detailed steps and conditions for validating each feature.
* **Reports**: Daily status reports, defect reports, and test summary reports.

## **Approval Process**

* Each document will be shared with the client for review and feedback.
* Testing will only proceed to the next phase once the client provides formal approval for the respective document.
* Any feedback or changes requested by the client will be incorporated, and the updated document will be resubmitted for approval.

## **Key Points**

* **Test Plan Approval**: Required before test case creation and execution begin.
* **Test Scenarios Approval**: Required before detailed test cases are developed.
* **Test Cases Approval**: Required before test execution starts.
* **Reports Approval**: Required at the end of each testing phase (e.g., test cycle, regression testing).

This structured approval process ensures alignment with client expectations and minimizes the risk of misunderstandings or rework.