Your test plan is comprehensive and detailed, covering all essential aspects needed to ensure a structured and efficient testing process for the **app.vwo.com** application. Below is a summary and suggestions for improvements, along with specific additions you might consider.

### **Test Plan Overview**

#### **Objective**

The test plan clearly outlines the goal of measuring the impact of A/B tests across various metrics. Ensuring that both leading and lagging indicators are tracked is a solid approach for evaluating the success of experiments. The inclusion of guardrail metrics is also a good strategy for catching potentially harmful experiments early.

#### **Scope**

The scope section defines the functional areas of **app.vwo.com** that will be tested, such as the UI, checkout process, and mobile compatibility. It also covers the testing types and environments that will be utilized. **Functional, API, Integration, database and regression end to end testings** will be done.

#### **Inclusions and Exclusions**

**Inclusions**: The login, dashboard, and create account pages are listed. You might also want to consider adding other critical flows, such as adding items to the cart or completing a purchase (for checkout).

**Exclusions**: The decision to exclude the support page and ZOHO chat widget is understandable, but it might be worth explaining why these areas were excluded.

#### **Test Environments**

Here’s a brief version of the **Test Environments** section, including the environments you've mentioned:

### **Test Environments**

* **DEV:** Development environment for initial testing.
* **QA:** Quality Assurance environment for feature testing.
* **UAT:** User Acceptance Testing environment for client validation.
* **STAGE/PROD:** Staging and Production environments for final validation before live deployment.

**Browsers and Devices:**

* **Browsers:** Chrome, Firefox, Safari, Microsoft Edge.
* **Devices:** Desktop, Mobile (Android & iOS).

**Other Details:**

Testing will occur across different operating systems and network configurations to ensure compatibility.

This version includes the key details about your test environments and cross-browser/device compatibility. Let me know if you'd like to expand on any areas!

### **Defect Reporting Procedure**

**Defect Identification:**

* Report defects when there's a deviation from requirements, errors, or user experience issues.

**Defect Reporting Tool:**

**Tool:** JIRA

**Include:**

* Defect Title, Description, Steps to Reproduce
* Actual vs Expected Results, Severity, Priority
* Attach Screenshots/Logs.

**Defect Assignment:**

* **Frontend:** Monika
* **Backend:** Sonal
* **DevOps:** Prajeeth

**Defect Triage:**

* Prioritize defects by severity (Critical, Major, Minor) and priority (High, Medium, Low).

**Resolution and Retesting:**

* Once resolved, defects are retested and marked as "Resolved" in JIRA.

**Metrics:**

* Track number of defects, time to resolve, and defect closure rate for continuous improvement.

#### **Test Strategy**

### **Test Strategy**

1. **Test Design Techniques:**

Equivalence Class Partition, Boundary Value Analysis, Decision Table Testing, State Transition Testing, Use Case Testing, Exploratory Testing.

**2: Test Procedure:**

**Smoke Testing** for critical functionalities.

**In-depth Functional Testing** for detailed feature verification.

**Regression Testing** to ensure no existing functionality is broken.

**3: Test Prioritization:**

Prioritize high-risk and high-impact test cases.

**Best Practices:**

**Context-**Driven Testing****, **Shift Left Testing**, and ****End-to-End** Flow Testing**.

#### **Test Schedule**

Your test schedule outlines key milestones such as creating the test plan and executing test cases.

"**Test Case Execution**" into more specific tasks, like initial execution, re-execution after defect fixes, and regression testing. This will help manage resources more effectively.

**Test Deliverables:**

Ensure that the deliverables include not only defect reports and test case reports but also test execution logs and test coverage reports. These are important for tracking progress and ensuring comprehensive testing.

* **Defect Reports**: Detailed descriptions of bugs with status and severity.
* **Test Case Reports**: Summary of executed test cases, their results, and any failed cases.
* **Test Execution Logs**: Records of test execution with timestamps, results, and issues encountered.

**Test Coverage Reports**: Overview of what parts of the system were tested, including coverage by feature or requirement.

These elements ensure comprehensive tracking of testing progress and defect management.

#### **Entry and Exit Criteria**

The entry and exit criteria for each phase are well-outlined. Having clear sign-off on test scenarios and test cases before starting execution is crucial.

**Entry Criteria:**

* 100% of test cases written and approved
* Stable build available, with no major issues flagged during smoke testing
* Test environments fully configured

**Exit Criteria:**

* 95% of test cases executed, with at least 90% passing
* 100% of critical defects resolved or mitigated
* Test documentation (Test Cases, Defects, Coverage Reports) finalized
* Regression tests show no major issues

### **Additional Tool Suggestions:**

**Cucumber** (if you need BDD):

If your project requires behavior-driven development (BDD) or user story-based testing, you might want to integrate **Cucumber** with **Selenium**. This allows you to write tests in plain English (using Gherkin syntax), making it easier for non-technical stakeholders to understand the tests.

**Browser Stack** (for cross-browser testing in the cloud):

For cross-browser testing, Browser Stack allows you to run your **Selenium** scripts on real devices and browsers without needing to maintain your own test infrastructure.

### **Summary of Tools:**

* **JIRA**: For bug tracking and project management.
* **Selenium with Java**: For automating UI testing.
* **Postman**: For API testing and validation.
* **TestNG**: For organizing and executing automated tests.
* **Jenkins**: For automating test execution within CI/CD pipelines.
* **Allure**: For generating test execution reports.
* **Cucumber** (optional): For behavior-driven development (BDD) testing.

### **Risks and Mitigation**

**1: Last-Minute Changes to Test Requirements**

**Risk:** Changes to test requirements after test cases are created.

**Mitigation:** Buffer time in the schedule, clear communication, and version control for test cases.

**2: Integration Issues with Third-Party Tools/Services**

**Risk:** Problems with external integrations (e.g., APIs, payment systems).

**Mitigation:** Pre-test integrations, use mocking tools, and communicate closely with third-party vendors.

**3: Incomplete or Inaccurate Test Data**

**Risk:** Invalid test data leading to missed defects.

**Mitigation:** Develop comprehensive test data plans, use automated data generation, and regularly update test data.

**4: Delayed or Unstable Build**

**Risk:** Delays or instability in the build impacting testing timelines.

**Mitigation:** Continuous integration, smoke testing, and stable build verification.

**5: Limited Resources or Resource Unavailability**

**Risk:** Insufficient manpower or skill gaps.

**Mitigation:** Backup resources, cross-training, and possibly outsourcing during peak times.

**6: Environment or Infrastructure Issues**

**Risk:** Downtime or issues in the testing environment.

**Mitigation:** Automated health checks, backup environments, and regular maintenance.

**7: Unforeseen Changes in Technology or Tools**

**Risk:** Updates to tools or tech stack causing test failures.

**Mitigation:** Monitor changes, implement version control, and conduct compatibility checks before upgrades.

**8: Communication Gaps Between Teams**

**Risk:** Misalignment between development, testing, and business teams.

**Mitigation:** Regular meetings, collaboration tools, and clear documentation.

#### **Approvals**

It's clear that document approvals are needed before moving forward with testing activities. This step is crucial for ensuring alignment between stakeholders and the testing team.

* Will be Approved by **Promod Dutta/ Dipak P**
* Time line :4/7/2025-4/30/2025

### **Final Thoughts**

The test plan is quite thorough and well-organized, offering a good foundation for testing the **app.vwo.com** Implementing the suggestions above will help make the process even smoother, with clear responsibilities and risk mitigation strategies.