[**Test Plan**](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.30j0zll)[**1**](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.30j0zll)

[Objective](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3znysh7) 1-4

[Scope](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2et92p0) 4-5

[Inclusions](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.tyjcwt) 5

[Test Environments](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3dy6vkm) [6](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3dy6vkm)-8

[Defect Reporting Procedure](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.1t3h5sf) 8-10

[Test Strategy](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.4d34og8) 10

[Test Schedule](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2s8eyo1) 11

[Test Deliverables.](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.17dp8vu) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.17dp8vu)2

[Entry and Exit Criteria](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3rdcrjn) 12

[Entry Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.26in1rg) 12

[Exit Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.lnxbz9) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.lnxbz9)2

[Test Execution](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.35nkun2) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.35nkun2)2

[Entry Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.1ksv4uv) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.1ksv4uv)3

[Exit Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.44sinio) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.44sinio)3

[Test Closure](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2jxsxqh) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2jxsxqh)3

[Entry Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.z337ya) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.z337ya)3

[Exit Criteria:](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3j2qqm3) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.3j2qqm3)3

[Tools](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.1y810tw) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.1y810tw)4

[Risks and Mitigations](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.4i7ojhp) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.4i7ojhp)4

[Approvals](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2xcytpi) [1](https://docs.google.com/document/d/1JDF1xfkYxu0gPoyyf_ZstLLOoAgjCxu5/edit#heading=h.2xcytpi)5

1. **Objective**

This document outlines the test plan for the **app.demodbi.com** which is that there are so many microservices in website applications. The objective is to ensure that all features and functionalities work as expected for the target audience, whoever wants to use this website for their business.

Track both leading and lagging indicators for your experiment's impact

Keep an eye on guardrail metrics to stop bad experiments early on

Component Version

* React 18.2.0
* jQuery 2.1.1
* JavaScript ES6+
* PostgreSQL Latest
* Web Server Apache (latest)

1. **Scope**

This test plan defines the testing scope for **app.demodbi.com**, a microservices-based web application designed for business users. The primary goal is to ensure that all integrated services, user-facing features, and backend components function reliably, securely, and efficiently under various conditions.

2.1 Features and Functionality to be Tested

* **User Interface (UI):** Usability, layout consistency, and responsiveness across devices.
* **Authentication and Authorization Services:** User login, role-based access control, session handling.
* **Business Logic Microservices:** Data processing, analytics, task automation, or business-specific operations.
* **API Integrations:** Internal and external service communication via RESTful APIs.
* **Notifications and Alerts:** Real-time updates, system messages, and email/SMS notifications.
* **Data Handling and Reporting:** Dashboard views, report generation, and database interactions.
* **Mobile and Cross-Platform Compatibility:** Accessibility on various devices and screen sizes.

2.2 Types of Testing

* **Functional Testing:** Validation of business workflows and service logic.
* **Integration Testing:** Ensuring seamless interaction between multiple microservices.
* **Manual Testing:** UI and exploratory testing across all modules.
* **Automated Testing:** Regression, smoke, and critical path testing using automation tools.
* **Performance and Load Testing:** Assess system behavior under high user load and data volume.
* **Security Testing:** Authentication, authorization, data encryption, and input validation checks.
* **Accessibility Testing:** Compliance with accessibility standards (WCAG 2.1).

2.3 Test Environments

* **Operating Systems:** Windows, macOS, Linux (Ubuntu).
* **Browsers:** Chrome, Firefox, Safari, Edge (latest versions).
* **Devices:** Desktop, tablet, and mobile.
* **Network Conditions:** High latency, low bandwidth, and standard corporate networks.

2.4 Evaluation Criteria

* Number and severity of defects discovered
* Stability and performance of microservices under load
* Accuracy and reliability of data across modules
* Test execution coverage and completion rates
* Time to detect, fix, and retest issues
* Stakeholder and end-user feedback during UAT

2.5 Roles and Responsibilities

* **Test Lead:** Oversees test planning, execution, defect tracking, and reporting.
* **Manual Testers:** Execute test cases and exploratory tests.
* **Automation Engineers:** Maintain automated test frameworks and pipelines.
* **Developers:** Address reported defects and assist with integration debugging.
* **DevOps Team:** Manage test environment setup, deployment pipelines, and logging.
* **Product Owners:** Validate requirement coverage and approve test outcomes.

2.6 Schedule and Milestones

The testing schedule will follow a phased approach:

* Test Planning & Scenario Creation
* Test Case Development & Review
* Smoke & Functional Testing
* Integration and Regression Testing
* Performance & Security Testing
* Final Validation & Test Closure

2.7 Tools and Resources

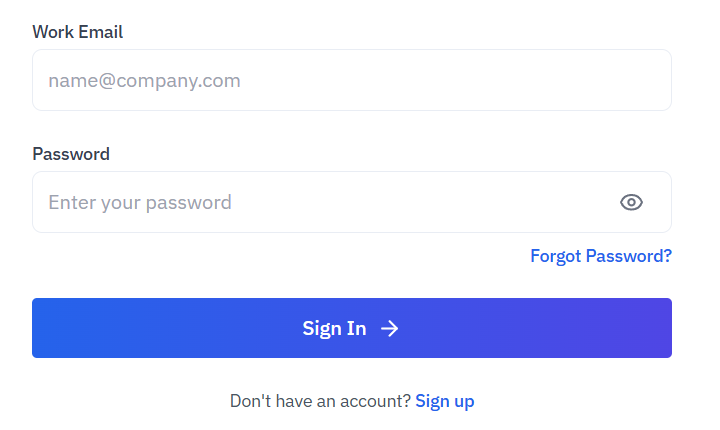
* **Test Management:** TestRail, Zephyr, or spreadsheets
* **Defect Tracking:** JIRA or equivalent
* **Automation Framework:** Cypress, Postman (API), Selenium
* **Performance Tools:** JMeter, k6
* **Security Tools:** OWASP ZAP, Burp Suite
* **CI/CD Integration:** Jenkins, GitHub Actions, or GitLab CI
* **Monitoring and Logs:** Grafana, ELK Stack (for microservice logs)

1. **Inclusions**

**Introduction: -** This section is provide a overview on the Test plane, purpose, Scope & goals

**Test Objective: -** This section is an outline of the section which is testing such as identifying and fixing defects, improving the user experience, or achieving a certain level of performance.

* Login
* Dashboard Page
* Create Account



**Exclusion**

1. Support Page.
2. Support All over the world.
3. **Test Environments**

Testing will be executed across multiple environments:

* **Operating Systems:** Windows, macOS, Linux (Ubuntu).
* **Browsers:** Chrome, Firefox, Safari, Edge (latest versions).
* **Devices:** Desktop, tablet, and mobile.
* **Network Conditions:** High latency, low bandwidth, and standard corporate networks.
* **Authentication and Authorization Services:** User login, role-based access control, session handling.

**5. Defect Reporting Procedure**

This section outlines the procedure for identifying, logging, triaging, resolving, and tracking defects discovered during the testing of **app.demodbi.com**. The objective is to ensure all defects are systematically recorded, addressed, and resolved within the expected timelines and quality standards.

5.1 Criteria for Identifying a Defect

A defect is defined as any deviation from the expected behavior, including but not limited to:

* **Functional issues:** Feature not working as per requirement.
* **User experience (UX) flaws:** Poor interaction flow, inconsistent design elements.
* **UI defects:** Alignment, responsiveness, or accessibility problems.
* **Security vulnerabilities:** Unintended access or lack of input sanitization.
* **Integration errors:** Failure in communication between microservices.
* **Performance problems:** Excessive load times or resource consumption.
* **Browser/device-specific issues.**

5.2 Steps for Reporting a Defect

Defects will be reported using a structured and traceable method to ensure quick understanding and resolution. Each defect should include:

1. **Title:** Clear and concise summary of the defect.
2. **Description:** Detailed explanation of the observed issue.
3. **Steps to Reproduce:** Step-by-step guide to replicate the defect.
4. **Expected Result vs. Actual Result.**
5. **Environment Details:** Browser, OS, device, test environment.
6. **Severity & Priority:** Defined per guidelines (see 6.3).
7. **Screenshots/Logs:** Visual evidence or console/network logs.
8. **Test Case Reference (if applicable).**

Defects must be logged in the defect tracking tool (e.g., JIRA) using the designated template.

5.3 Triage and Prioritization Process

Defects will be reviewed daily during triage meetings. Triage criteria include:

* **Severity Levels:**
  + *Critical:* System crash, data loss, complete blocker.
  + *High:* Major functionality not working; no workaround.
  + *Medium:* Partial impact with available workaround.
  + *Low:* Minor issue, cosmetic or non-critical.
* **Priority Levels:**
  + *P1:* Must fix before release.
  + *P2:* Fix in the current sprint or patch.
  + *P3:* Deferred or fixed in future releases.

Defects will be assigned to appropriate developers or service owners based on module ownership.

5.4 Tools and Systems

The following tools will be used for defect management:

* **Defect Tracking:** JIRA (or alternative selected by the team)
* **Test Case Management:** TestRail / Zephyr / Excel
* **Screenshot/Recording Tools:** Snagit, Loom, or built-in browser tools
* **Log Capture:** Browser DevTools, API tools (Postman), or backend log aggregators (e.g., ELK)

5.5 Roles and Responsibilities

* **Testers:** Identify, reproduce, and log defects with complete details.
* **Test Lead:** Review and triage reported defects, validate fixes, ensure defect lifecycle adherence.
* **Developers:** Investigate assigned defects, provide root cause analysis, and implement fixes.
* **Product Owners (optional):** Validate business-critical defects or UAT blockers.

5.6 Communication Channels and Updates

* **Daily Standups:** Include defect summary/status.
* **Defect Triage Meetings:** Held on alternate days or as required.
* **Slack/MS Teams Channels:** For instant collaboration and clarifications.
* **Weekly QA Status Report:** Sent to all stakeholders with defect metrics and critical blockers.

5.7 Metrics for Measuring Effectiveness

* Total Number of Defects Logged
* Defect Distribution by Severity and Priority
* Average Time to Resolve (TTR)
* Percentage of Reopened Defects
* Fix Success Rate (First-time fix %)
* Defect Leakage Rate (to production)
* Automation Defect Catch Rate (vs. manual)

These metrics will be reviewed in retrospectives and used to improve test coverage, defect detection speed, and product quality.

**6. Test Strategy**

The first step to create a test scenario & the Test case of the all feature which is present in the scope.

6.1: Test Case Design Approach

Test Design Techniques

To ensure thorough and efficient test coverage, the following black-box test design techniques will be applied:

* **Equivalence Class Partitioning (ECP):** Dividing input data into valid and invalid classes to minimize test cases while maximizing coverage.
* **Boundary Value Analysis (BVA):** Focusing on values at the edges of input ranges where defects are more likely to occur.
* **Decision Table Testing:** Handling scenarios with multiple input combinations that affect business logic.
* **State Transition Testing:** Applied where system behavior changes depending on state (e.g., login status, feature availability).
* **Use Case Testing:** Designed around real-world workflows and user interactions.

Experience-Based Techniques

* **Error Guessing:** Leveraging prior experience and defect trends to anticipate high-risk areas.
* **Exploratory Testing:** Performed without predefined steps, allowing testers to uncover unexpected issues through creative exploration.

Test Case Prioritization

Test cases are prioritized based on risk, business impact, and frequency of use:

* **High Priority:** Business-critical workflows, integrations, and security-related components.
* **Medium Priority:** Supporting functionalities and edge scenarios.
* **Low Priority:** Cosmetic issues and rarely used options.

6.2: Testing Execution Procedure

Initial Verification - Smoke Testing

* Upon receiving a new build, **smoke testing** is performed to validate critical functionalities and stability.
* If smoke testing fails, the build is **rejected** and returned to the development team. Testing is paused until a stable build is provided.

Functional & In-Depth Testing

* Once a build passes smoke testing, **in-depth testing** begins using previously prepared test cases.
* **Multiple testers** may execute test cases in parallel across supported environments (browsers, OS, and devices) to ensure compatibility and scalability.

Bug Reporting

* All identified defects are logged in a **bug tracking tool** (e.g., JIRA), complete with reproduction steps, environment details, screenshots, and severity/priority levels.
* A **daily status report** summarizing newly found and open defects is shared with the development and project management teams.

Types of Testing Performed

* Smoke Testing
* Sanity Testing
* Regression Testing
* Retesting
* Usability Testing
* Functionality Testing
* UI Testing

6.3: Testing Best Practices

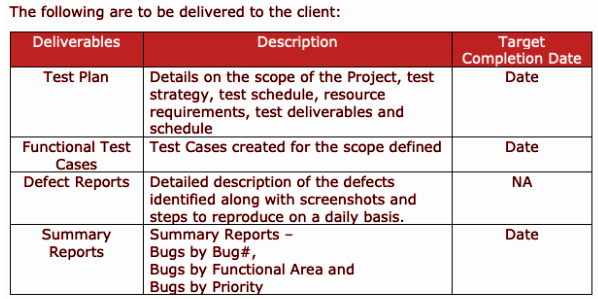
* **Context-Driven Testing:** Test effort is adapted to the application’s domain, use cases, and business context.
* **Shift-Left Testing:** Testing activities will begin early in the development lifecycle, including reviewing requirements, writing unit-level tests, and involving QA in design decisions.
* **Exploratory Testing:** Testers will use their domain expertise to identify issues outside scripted test cases.
* **End-to-End Flow Testing:** Testing of complete user journeys across multiple modules and services will be performed to simulate real user scenarios.

**7. Test Schedule**

The following table outlines the high-level test schedule planned for the project. This schedule includes key phases such as test planning, test case design, execution, and reporting. Specific timelines may be adjusted based on sprint planning, build readiness, or project constraints.

| **Task** | **Time Duration** | **Planned Dates** |
| --- | --- | --- |
| **Creating Test Plan** | 2 working days |  |
| **Test Case Creation** | 4 working days |  |
| **Test Case Review & Signoff** | 1 working day |  |
| **Test Case Execution** | 6 working days (2 cycles) |  |
| **Defect Reporting & Retesting** | Concurrent with execution |  |
| **Test Summary Report Submission** | 1 working day |  |

**8. Test Deliverable**



**9. Entry and Exit Criteria**

Entry and exit criteria define the conditions that must be met before moving in or out of a testing phase within the Software Testing Life Cycle (STLC).

9.1 Requirement Analysis

* **Entry Criteria:**
  + Requirements documentation (Business Requirements Document, Functional Specifications) has been received.
  + Stakeholders have walked the QA team through the scope and objectives of the application.
* **Exit Criteria:**
  + All functional and non-functional requirements have been understood and clarified.
  + Open questions or ambiguities are resolved and documented.
  + Initial test scenarios are drafted and reviewed.

9.2 Test Execution

* **Entry Criteria:**
  + Test Scenarios and Test Cases are reviewed and signed off by stakeholders or the client.
  + A stable and testable build of the application is deployed and available.
  + Required test environments are configured and accessible.
* **Exit Criteria:**
  + All planned test cases are executed.
  + Defects are logged, tracked, and either resolved or documented.
  + Test Execution Report and Defect Report are completed.

**10. Test Closure**

* **Entry Criteria:**
  + Final Test Case Execution Reports and Defect Reports are complete.
  + All high and critical severity defects are resolved or deferred with sign-off.
* **Exit Criteria:**
  + Test Summary Report is completed and reviewed.
  + Lessons learned and retrospective notes are documented (if applicable).
  + QA sign-off is provided.

**11. Tool**

The following tools will be utilized throughout the testing lifecycle:

| **Tool** | **Purpose** |
| --- | --- |
| **JIRA** | Bug tracking, task assignment, and defect lifecycle management |
| **Mind Mapping Tools** | Visual planning and requirement analysis |
| **Snipping Tool / Loom** | Capturing screenshots or screen recordings of test results or defects |
| **MS Word and Excel** | Test documentation, reporting, and traceability matrices |

**12. Risks and Mitigations**

The following table outlines potential risks that may impact the testing process, along with corresponding mitigation strategies:

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Non-availability of QA resources | Maintain a backup resource pool or cross-train team members |
| Build URL not accessible or stable | QA will focus on documentation, test case review, or non-blocking tasks |
| Limited time for test execution | Scale the team dynamically, prioritize critical test cases |
| Changing requirements during execution | Implement change control process and update impacted test cases |
| Third-party integration unavailability | Use stubs/mocks or simulate API responses temporarily |

**13. Approvals**

Testing activities are governed by milestone-based approvals from stakeholders. The QA team will seek formal approval on the following deliverables before proceeding to the next phase:

* ✅ **Test Plan**
* ✅ **Test Scenarios**
* ✅ **Test Cases**
* ✅ **Test Reports (Execution and Summary)**