OrangeHRM End-to-End Automation Testing - Test Plan

1. Introduction

This Test Plan outlines the strategy, scope, approach, resources, and schedule of the end-to-end automation testing for OrangeHRM. The primary goal is to ensure that all HR functionalities such as Login, Employee Management, Leave Management, Recruitment, and Reporting are working as expected through automated testing.

2. Objectives

- Validate core functionalities of OrangeHRM through automation. - Improve test coverage and ensure consistent execution. - Detect defects early in the development cycle. - Provide reliable and repeatable test results.

3. Scope

In Scope: - Login Module Automation - PIM (Employee Management) Automation - Leave Management Automation - Recruitment Module Automation - Dashboard & Reports Verification Out of Scope: - Third-party integrations - Performance & Load Testing (handled separately)

4. Test Approach

- **Framework**: Selenium WebDriver, TestNG, Page Object Model (POM) - **Test Types**: Functional, Regression, Smoke, and End-to-End Testing - **Test Data**: Externalized using Excel/JSON - **Reporting**: Extent Reports with screenshots on failure - **CI/CD Integration**: Jenkins/GitHub Actions

5. Test Deliverables

- Test Plan Document - Automated Test Scripts - Test Execution Reports - Defect Logs - Test Summary Report

6. Roles & Responsibilities

Role	Responsibility
QA Engineer	Develop and execute automated test scripts
Test Lead	Review test cases, monitor progress, and ensure quality
DevOps Engineer	Maintain CI/CD pipeline and test environment

7. Entry & Exit Criteria

Entry Criteria: - Test environment is set up - All required test data is prepared - Automation framework is ready Exit Criteria: - All planned test cases executed - All critical defects fixed and verified - Test summary report delivered

8. Schedule

Week 1: Framework Setup & Test Data Preparation Week 2: Script Development for Login & PIM Week 3: Script Development for Leave & Recruitment Week 4: Full Regression Execution & Reporting

9. Risks & Mitigation

- Test environment downtime → Mitigation: Use backup environment - Flaky tests due to UI changes → Mitigation: Implement robust locators - Delayed builds → Mitigation: Parallel execution & CI pipeline