Test Plan – Web Application QA Project

# 1. Introduction

This test plan outlines the testing strategy for the Web Application QA Project, which includes modules such as Login, Signup, Dashboard, and API Integration. The objective is to ensure the functionality, reliability, and usability of the application through manual and automated testing approaches.

# 2. Scope

The scope includes testing the following functionalities:  
• User Authentication (Login & Signup)  
• Dashboard components and user navigation  
• Backend API integrations and responses  
Testing will be performed using manual test cases as well as Selenium-based automation for regression and repetitive flows.

# 3. Objectives

• Validate critical workflows including login, signup, and navigation.  
• Identify defects and inconsistencies in modules.  
• Ensure application is secure, reliable, and performs well under expected load.

# 4. Test Strategy

• Manual Testing: Functional, UI, and exploratory testing of all modules.  
• Automation Testing: Selenium + TestNG used for automating login, signup, and dashboard flows.  
• Regression Testing: Executed after each build or bug fix.  
• API Testing: Using Postman to verify all RESTful endpoints.

# 5. Test Deliverables

• Test Plan Document  
• Test Cases (Excel format)  
• Bug Report (Excel format)  
• Selenium Automation Scripts  
• Final Test Summary Report

# 6. Resources

• Tools: Selenium, TestNG, Eclipse/IntelliJ, GitHub, Postman, MS Excel, MS Word  
• Team: 1 QA Engineer (Your Name)

# 7. Entry and Exit Criteria

Entry Criteria:  
• Development complete for target modules  
• Test environment is set up and stable  
• Test data is available  
  
Exit Criteria:  
• All critical test cases executed and passed  
• No high/critical severity bugs open  
• Test summary report reviewed and approved

# 8. Schedule

• Test Planning – 1 day  
• Test Case Design – 2 days  
• Test Execution – 3 days  
• Bug Reporting – Concurrent with execution  
• Test Closure – 1 day

# 9. Risks & Mitigations

Risk: Delay in build availability or unstable environment  
Mitigation: Early coordination with the dev team  
  
Risk: Incomplete requirements  
Mitigation: Clarify with product team early  
  
Risk: Script failure during automation  
Mitigation: Manual fallback and logging mechanism