Pump Master Solution Application Test Strategy

**Prepared by**: Santhi Varikuppala

Version :1.0

# Table of Contents

1. Introduction

2. Top Five Critical Areas for Focused Testing

2.1. Authentication & Security

2.2. Pump Data Integrity & CRUD Operations

2.3. Role-Based Access & Permissions

2.4. Data Export, Reporting, and Integration

2.5. Usability & Responsive Design

2.6. Prioritization Rationale

3. Sustainable Testing Strategy & Tools

4. API Testing Plan

5. Automated UI Test Example (Playwright)

6. QA Strategy: Web & Mobile Synchronization

7. Bug Handling: Critical Bug Before Release

# 1. Introduction

# Pump Master Solution Application Test Strategy that covers testing and automation and CICD Pipeline.

# 2. Top Five Critical Areas for Focused Testing

## 2.1. Authentication & Security

Authentication is the gateway to all protected features and data. The solution uses JWT tokens and supports login, token refresh, and logout. Any flaw here could allow unauthorized access or denial of service.  
  
Focus:  
- Valid/invalid login attempts  
- Token expiration and refresh  
- Access control for protected endpoints  
- Logout and token invalidation  
- Security against common vulnerabilities (e.g., brute force, token tampering)

## 2.2. Pump Data Integrity & CRUD Operations

The core of the application is managing pump data (creation, update, deletion, retrieval). Data integrity is essential for operational accuracy and compliance.  
  
Focus:  
- Field validation (required, type, range)  
- Handling of new/updated fields (area, latitude, longitude, flow rate, pressures)  
- Consistency between UI and API  
- Error handling for invalid or missing data

## 2.3. Role-Based Access & Permissions

Different users (e.g., admin, operator) may have different access levels. Ensuring correct permissions prevents data leaks and unauthorized actions.  
  
Focus:  
- Access restrictions for sensitive operations  
- UI and API enforcement of roles  
- Attempted privilege escalation scenarios

## 2.4. Data Export, Reporting, and Integration

Exporting pump data and integrating with other systems is critical for business operations and regulatory compliance.  
  
Focus:  
- Export functionality (format, completeness, accuracy)  
- Handling of large datasets  
- Data privacy in exports  
- API integration points

## 2.5. Usability & Responsive Design

The solution is used across web and mobile devices. Usability and responsive design directly impact user productivity and satisfaction.  
  
Focus:  
- UI consistency and accessibility  
- Mobile and tablet layout verification  
- Error messages and user feedback  
- Keyboard navigation and accessibility

## 2.6. Prioritization Rationale

These areas are prioritized based on risk (security, data loss), business impact (core pump management, reporting), and user experience (usability, access). Focusing automation and manual testing on these domains will maximize defect detection and ensure a robust, secure, and user-friendly solution.

# 3. Sustainable Testing Strategy & Tools

Strategy:  
- Adopt a shift-left approach: integrate testing early in the development lifecycle.  
- Maintain a robust automated regression suite (API + UI).  
- Use CI/CD pipelines for continuous feedback and fast detection of regressions.  
- Regularly review and refactor tests to cover new features and deprecate obsolete ones.  
- Enforce code reviews and test coverage thresholds.  
  
Tools:  
- Playwright for cross-browser UI automation (web & mobile emulation).  
- Postman or RestAPI , Newman for API execution and reporting.  
- Allure or Playwright HTML Reporter for reporting.  
- ESLint/Prettier for code quality.  
- GitHub Actions or Azure Pipelines for CI/CD.  
- TestRail or Xray for test case management (optional).

# 4. API Testing Plan

Key Areas:  
- Authentication (login, token refresh, logout)  
- CRUD operations for pumps (including all new fields)  
- Validation (required fields, data types, boundary values)  
- Error handling (invalid data, unauthorized access, non-existent resources)  
- Rate limiting and concurrency  
- Data export and integration endpoints  
  
Approach:  
- Use data-driven tests to cover positive and negative scenarios.  
- Automate API tests and run them in CI.  
- Validate both response structure and business logic.  
- Monitor API performance and reliability.

# 5. Automated UI Test Example (Playwright)

Scenarios: Simulate user login and verify successful pump creation, updation, edit, delete, search, and filter report.  
  
PUMP-MASTER-AUTOMATION solution leverages a modern, maintainable test stack:  
  
**BDD (Behavior-Driven Development):** Test scenarios are written in Gherkin syntax, ensuring clear communication between business and technical teams and traceability from requirements to tests.  
  
**TypeScript + Playwright**: All automated UI and API tests are implemented in TypeScript using Playwright, enabling robust, fast, and maintainable cross-browser automation for both web and mobile (using device emulation).  
  
**Web & Mobile Cross-Browser**: Automated tests are executed across Chromium, Firefox, and WebKit browsers, as well as mobile emulation (iOS/Android), ensuring consistent user experience and functionality.  
  
**Azure Pipeline & CI/CD**: All tests are integrated into Azure DevOps pipelines, providing automated build, test, and deployment workflows. This ensures every code change is validated for both web and API layers before release.  
  
Approach:  
- **Unified Test Scenarios**: BDD feature files are shared for both web and mobile, ensuring consistent coverage and requirements traceability.  
- **Cross-Platform Automation**: Playwright scripts run on multiple browsers and mobile emulators in parallel, catching platform-specific issues early.  
- **Continuous Integration**: Every pull request triggers the Azure pipeline, running all regression tests (UI and API) automatically.  
- **Continuous Delivery**: Only builds that pass all automated checks are eligible for deployment, reducing manual intervention and risk.  
- **Reporting & Traceability**: Test results are published in Azure DevOps, with links to BDD scenarios and code coverage reports for transparency.  
**- Manual & Exploratory Testing**: Supplement automation with manual checks on real devices for usability and edge cases.  
  
Sample automation framework: <https://github.com/QAtesting8540/pumpmasterapplication>

# 6. QA Strategy: Web & Mobile Synchronization

**Unified Test Data**: Use the same test data and scenarios for both web and mobile.  
**Cross-Platform Automation**: Use Playwright’s device emulation for mobile browsers.  
**Responsive Design Checks**: Automate viewport and device checks for all critical workflows.  
**Manual Exploratory Testing**: Regularly perform manual checks on real devices.  
**Defect Tracking**: Log and track issues by platform, ensuring fixes are validated on both.

# 7. Bug Handling: Critical Bug Before Release

**Immediate Actions**:  
- Triage and reproduce the bug.  
- Assess impact and affected areas.  
- Communicate with stakeholders and halt the release if necessary.  
- Prioritize and assign the fix to the development team.  
- Create or update automated regression tests to cover the bug scenario.  
- Retest the fix and perform a focused regression on related features.  
- Only proceed with the release after confirming the bug is resolved and no regressions are introduced.