Estimation & Risk Matrix

Automation Strategy for ThingsBoard Platform

1. Introduction

This document provides a detailed estimation, resource plan, and risk matrix for implementing a scalable, CI/CD-ready test automation solution for the ThingsBoard IoT platform. The scope of work includes:

- UI automation using Playwright
- REST API validation using Python Requests
- Real-time telemetry ingestion and dashboard verification
- Integration into CI/CD pipelines
- Test data simulation and environment setup

2. Effort Estimation – High-Level Planning

Module	Description	Effort (Person-Weeks)
Framework Setup	Initial Pytest, Playwright, Requests integration, repo structure, test hooks	3
Test Data Layer	Creating fixture schema for device profiles and tenants	2
API Automation (5 Modules)	Auth, Telemetry, Device, Alarm, Rule Engine	4
UI Automation (5 Modules)	Login, Dashboard, Device UI, Alarm UI, Navigation flows	5
Real-Time Validation	Polling/WebSocket-based telemetry assertion	2
Simulation Setup	MQTT/HTTP test scripts for data injection	2
CI/CD Integration	GitHub Actions or Azure pipelines, notifications, basic jobs	2
Regression & Nightly	Categorization, smoke vs. full suite, nightly trigger setup	1
Training & Reviews	Team KT, pairing sessions, feedback cycles	2

Total Effort: 23 Person-Weeks
Assuming 2 team members: ~11–12 calendar weeks, accounting for onboarding, ramp-up, and technical support.

3. Team Composition & Skill Matrix

Role	Key Skills	Responsibility	FTE
Automation Lead	Python, Pytest, Playwright, CI/CD	Strategy, mentoring, framework design	1
Automation Engineer	REST APIs, selectors, curl, Git	Script development, data simulation, debugging	2
DevOps (part-time)	GitHub Actions, YAML, artifact management	Pipeline support, report upload, fail triggers	0.2

4. Tool Selection Rationale

Function	Tool	Reason
UI Testing	Playwright	Fast, auto-wait, headless, cross-browser, easy for beginners
API Testing	Python requests	Lightweight, readable, minimal learning curve
Load Simulation	curl/MQTT scripts	Easy manual and scripted data simulation
Test Runner	Pytest	Extensible, plugin support, familiar to Python users
Reporting	Allure/HTML reports	Human-readable, visual debug traces
CI/CD	GitHub Actions	Free tier available, community-supported, scriptable

5. Automation Pyramid Strategy

Layer	Focus	Weight %	Rationale
Unit + Integration	Parsers, logic, rules	50%	Fast feedback, stable even during UI churn
API / Contract	Devices, telemetry, alarms, tokens	35%	Ensures backend integrity
UI End-to-End	Login, dashboard, critical widgets	15%	Limited but covers high-value paths

6. Test Data Strategy

Aspect	Approach
Tenant Isolation	Each run creates/destroys a dedicated tenant via API
Device Fixtures	JSON/CSV defining device types, telemetry keys
Feature Flag Testing	Use API calls to turn on/off features dynamically
Telemetry Simulation	curl or MQTT scripts simulating devices like valves or temperature sensors
Validation Sync	Check API data after ingestion and compare with dashboard/UI widgets

7. CI/CD Integration Plan

Stage	Step	Time	Details
Stage 1	Lint + Unit Tests	< 5 min	Quick checks to block broken commits
Stage 2	API Tests	3–7 min	Auth, device, telemetry verification
Stage 3	UI Smoke Tests	5–10 min	Dashboard, login, widgets visibility
Stage 4	Full Regression	Overnight	Runs all tests with history
Artifacts	Reports, logs	-	Allure reports + failure screenshots
Notifications	Slack, Email, Jira (auto)	-	Fast alerts and defect traceability

8. Traceability & Versioning

Aspect	Plan
Requirement Mapping	All tests linked to Jira Epic/User Story
Version Tagging	Test branches tagged with Git release versions
Metrics Tracked	Execution time, failure %, skipped count, test coverage
KPI Dashboards	MTTD (detect), MTTR (resolve), test ROI, stability index

9. Risk Identification & Mitigation Matrix

Category	Risk	Impact	Mitigation
Technical	UI locator changes	High	Use test-IDs, fallback XPath, centralized selector files
Product	Shared demo tenants create test interference	Medium	Use dedicated tenant per run with teardown API
Simulation	No real device during phase 1	Medium	Use curl/MQTT as mock publisher, add real HW in phase 2
Team Ramp-up	Team unfamiliar with platform/tools	High	Conduct bootcamps, pair programming, KT sessions
CI/CD	Long pipeline runs slow delivery	Medium	Categorize tests into smoke vs regression
Process	No defined release tagging for tests	Medium	Enforce versioning policy and Git tags in test repos

10. Summary

This estimation and risk plan is designed to ensure high test coverage, low flakiness, and a sustainable CI/CD-integrated automation strategy for the ThingsBoard platform. By balancing quick feedback loops (unit/API) with strategic UI coverage and real-time data validations, the suite ensures quality assurance at scale. The outlined risk mitigations support smooth onboarding and platform adaptability with Automation teams.