PrecisionPulse – Implementation Plan

1. Purpose of this document:

This document outlines a sprint-wise implementation plan for PrecisionPulse, detailing what to build, in what order, to ensure real-time telemetry streaming, offline resilience, and bi-directional synchronization between desktop and web applications.

Task ownership will be decided after plan approval.

---

2. Assumptions & Scope:

- **Sprint Duration:** 2 weeks per sprint
- **Total Sprints Planned:** 4 sprints
- **Backend:** Python Flask + MQTT integration
- **Frontend:** Next.js with real-time dashboards
- **Desktop Client:** PySide6
- **Databases:** PostgreSQL (web), SQLite (desktop)
- **Protocol:** MQTT (TLS encrypted), WebSocket bridge to web
- **Deployment:** Dockerized services on cloud
- **Testing & Quality:** Pytest, SonarQube, integration tests
- **Documentation:** Confluence per sprint

---

3. Overall Development Workflow:

- Understand requirements & domain modeling
- Design system architecture & database
- Implement authentication, authorization, and RBAC(Role-Based Access Control)
- Build core modules: telemetry, offline buffer, sync
- Implement commands & configuration updates
- Testing, quality checks, and documentation
- Production deployment readiness

---

4. Sprint 1 – Foundation, Architecture & Security:

Sprint Goal:
Establish a strong technical foundation, authentication, and base database models.

Backlog Items:

4.1 Project Setup & Architecture	Owner

Initialize Flask backend & modular architecture	Saniya Chavan
Initialize Next.js frontend with base layout and routing	Prasad Zade
Setup PostgreSQL & SQLite databases	Saniya Chavan
Configure Docker for backend, frontend, and MQTT broker	
Setup MQTT broker (TLS-enabled)	Saniya Chavan

4.2 Authentication & Authorization	**Owner**
Implement JWT login and token refresh	Saniya Chavan
Define roles: Admin, Viewer	Prasad Zade
Enforce RBAC at API & frontend levels	Prasad Zade
Protect web routes based on roles	Prasad Zade

4.3 Base Domain Models	**Owner**
Design tables for users, roles, permissions, clients	Saniya Chavan
Setup audit log and configuration tables	Saniya Chavan

4.4 Testing & Documentation	**Owner**
Setup Pytest for backend & desktop modules	Prasad Zade
Expose initial Swagger/OpenAPI documentation	Prasad Zade

Document architecture & setup in Confluence	Saniya Chavan

---

5. Sprint 2 – Telemetry Streaming & Offline Resilience

Sprint Goal:
Implement real-time telemetry streaming, offline buffering, and live dashboards.

Backlog Items:

5.1 Desktop Telemetry Module	**Owner**
Implement telemetry producer (integers, decimals, booleans)	Saniya Chavan
Send telemetry periodically via MQTT with timestamps	Saniya Chavan

5.2 MQTT Integration & Heartbeat	**Owner**
Setup MQTT subscriptions for telemetry, commands, sync, heartbeat	Saniya Chavan
Implement keep-alive / heartbeat logic	Saniya Chavan
Configure QoS levels: live (0), buffered (1), commands (2)	Saniya Chavan

5.3 Offline Buffer & Auto-Sync	**Owner**
Implement SQLite buffer table for unsent telemetry	Saniya Chavan
Auto-flush buffered data on reconnection	Prasad Zade
Ensure transactional safety and order preservation	Prasad Zade

5.4 Web Dashboard Updates	Owner
Implement Socket.IO bridge for MQTT → Next.js	Prasad Zade
Render live values dynamically without page refresh	Prasad Zade
Show connection status indicators	Prasad Zade

5.5 Testing & Documentation	Owner
Unit tests for telemetry generation & buffering	Saniya Chavan
Integration tests: Desktop ↔ MQTT ↔ Web	Saniya Chavan
Document telemetry & offline logic in Confluence	Prasad Zade

---

6. Sprint 3 – User Management & Bi-Directional Sync

Sprint Goal:
Enable user, role, and permission management, and bi-directional sync between desktop and web.

Backlog Items:

6.1 User & Role Management APIs	Owner
APIs to create, update, delete users, roles, and permissions	Saniya Chavan
Enforce RBAC and validations	Saniya Chavan

6.2 Bi-Directional Sync Engine	Owner
Publish changes via MQTT	Saniya Chavan
Conflict resolution using updated_at timestamp	Saniya Chavan

Desktop client listener applies remote updates	Saniya Chavan

6.3 Frontend Admin UI	Owner
Implement screens for user/role management	Prasad Zade
Display live sync status indicators	Prasad Zade

6.4 Testing & Documentation	Owner
Unit tests for sync engine & conflict resolution	Prasad Zade
Integration tests for desktop-web user sync	Prasad Zade
Update Confluence with API details	Prasad Zade

---

7. Sprint 4 – Commands, Config, Security & Production Readiness

Sprint Goal:
Implement remote commands, configuration management, security hardening, reporting, and prepare system for production.

Backlog Items:

7.1 Remote Commands & Config Management	Owner
Implement command APIs (UpdateConfig)	Saniya Chavan
Publish commands via MQTT, desktop executes with ACK	Saniya Chavan
Config updates via web → auto-sync to desktops	Saniya Chavan

7.2 Security & Encryption	Owner

Enforce TLS for MQTT and HTTP	Saniya Chavan
Sign sensitive MQTT payloads	Saniya Chavan
Validate RBAC across modules	Saniya Chavan

7.3 Logging, Reporting & Dashboard Enhancements	**Owner**
Audit logs for telemetry, commands, and user changes	Saniya Chavan
Reports for audit, config changes, and offline events	Saniya Chavan

7.4 Testing & Deployment	**Owner**
Integration & chaos tests (simulate offline, network loss)	Prasad Zade
Run SonarQube & resolve issues	Prasad Zade
Finalize Docker images & deploy backend, frontend, MQTT broker	Saniya Chavan
Update final Confluence documentation	Prasad Zade

---

8. Post-Sprint Activities

- Full system integration testing
- Cloud deployment validation
- Documentation cleanup
- Backlog refinement for future enhancements