Train Tracker Backend – Phase 3 Documentation

1. Phase 3 Overview

Phase 3 focuses on making the backend system real-time, scalable, and production-ready. This phase introduces WebSockets, refresh tokens, background schedulers, Redis caching, and system-level scalability design.

2. Key Objectives

- Enable real-time train updates
- Prevent frequent database polling
- Support long-lived user sessions
- Prepare system for high traffic and concurrency

3. Real-Time Architecture (WebSockets)

WebSockets allow persistent client-server connections. The server actively pushes live train status updates without repeated client requests. This is essential for real-time tracking systems.

4. Authentication Enhancements

Phase 3 introduces a refresh token mechanism. Short-lived access tokens are refreshed using long-lived refresh tokens stored securely in the database, enabling seamless user sessions.

5. Background Job Scheduler

A background scheduler automatically monitors train statuses and generates notifications. This eliminates manual API triggers and enables event-driven automation.

6. Redis Caching Layer

Redis is used as an in-memory cache to store the latest train state. Database writes occur only when changes are detected, significantly reducing load.

7. Scalability Strategy

The system is designed to scale horizontally using Gunicorn workers. Redis acts as a shared cache and messaging backbone. Future upgrades include Redis Pub/Sub and Celery workers.

8. Production Architecture

Clients communicate with FastAPI running behind Gunicorn. WebSocket connections support live updates. Redis handles caching and message coordination, while PostgreSQL remains the source of truth.

9. Performance & Reliability Considerations

Techniques such as caching, background processing, TTL-based cleanup, and selective database writes ensure consistent performance under load.

10. Phase 3 Summary

Phase 3 transforms the backend into a real-time, scalable platform. The system now supports live updates, automated workflows, efficient caching, and production-level architectural thinking.

Phase 3 – High Level Flow (Textual Diagram)

Client connects via WebSocket
Server pushes live train updates
Scheduler monitors train status
Redis caches last known state
Notifications generated on state change
Users receive updates in real time