ChatGPT

Productivity Analytics Ecosystem – Development Plan

Current Prototype Status

- **Frontend Extension:**
- Tracks website usage: tab URL, title, scroll depth, duration.
- Categorizes domains (e.g., research, entertainment).

- Detects AFK state via event listeners.

- **Backend:**

- FastAPI-based API: `/track-event`, `/track-afk`, `/afk-events`, `/events`
- Redis queue for ingestion
- Worker service saves to PostgreSQL
- SQLAlchemy models: `Event`, `AFKEvent`

Goal: Scalable Productivity Analytics Ecosystem

1. DSA-Driven Analytics Engine

- Merge events into sessions using interval trees / sorting
- Aggregate active/idle time with:
- Sliding windows
- Prefix sums
- Time bucket rollups
- Use tries/dict trees for advanced domain matching

2. OOP Architecture & Design Patterns

- Refactor backend into modular components:
- **Strategy Pattern**: productivity scoring engines per category
- **Factory Pattern**: AFK/event creation abstraction
- **Repository Pattern**: DB access isolation layer

3. Multi-User Support (For Chrome Store Release)

- Add `user_id` to all tracked events
- Secure endpoints with JWT or OAuth2
- Partition database queries by user

• Design endpoints for concurrent scale

4. Storage Mode Flexibility

• Local-only mode:
• IndexedDB frontend storage
• In-browser summary generation
• Server-sync mode:
• API batching with retry logic
• Toggleable sync state in settings

5. Scalable Ingestion Infrastructure

• Replace Redis with Kafka for scalable ingestion
• Add worker pool for concurrent processing
• Retry/failure recovery support

6. Frontend Expansion

• Extend popup UI:
• Show session summaries & streaks
• Real-time productivity feedback
• (Optional) Full React-based analytics dashboard

Next Recommended Step

Build a `session_engine.py`: - Load web + AFK events - Stitch continuous sessions - Compress and score sessions - Return daily productivity summary