Job Queue System Architecture

Job Enqueuing Flow
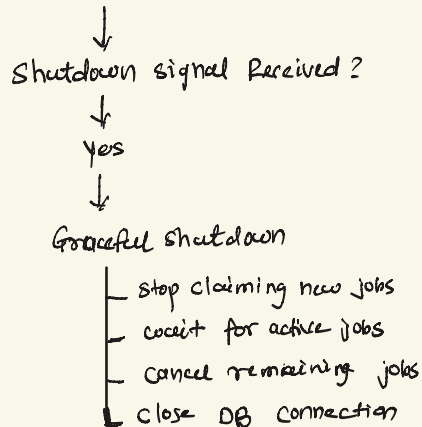
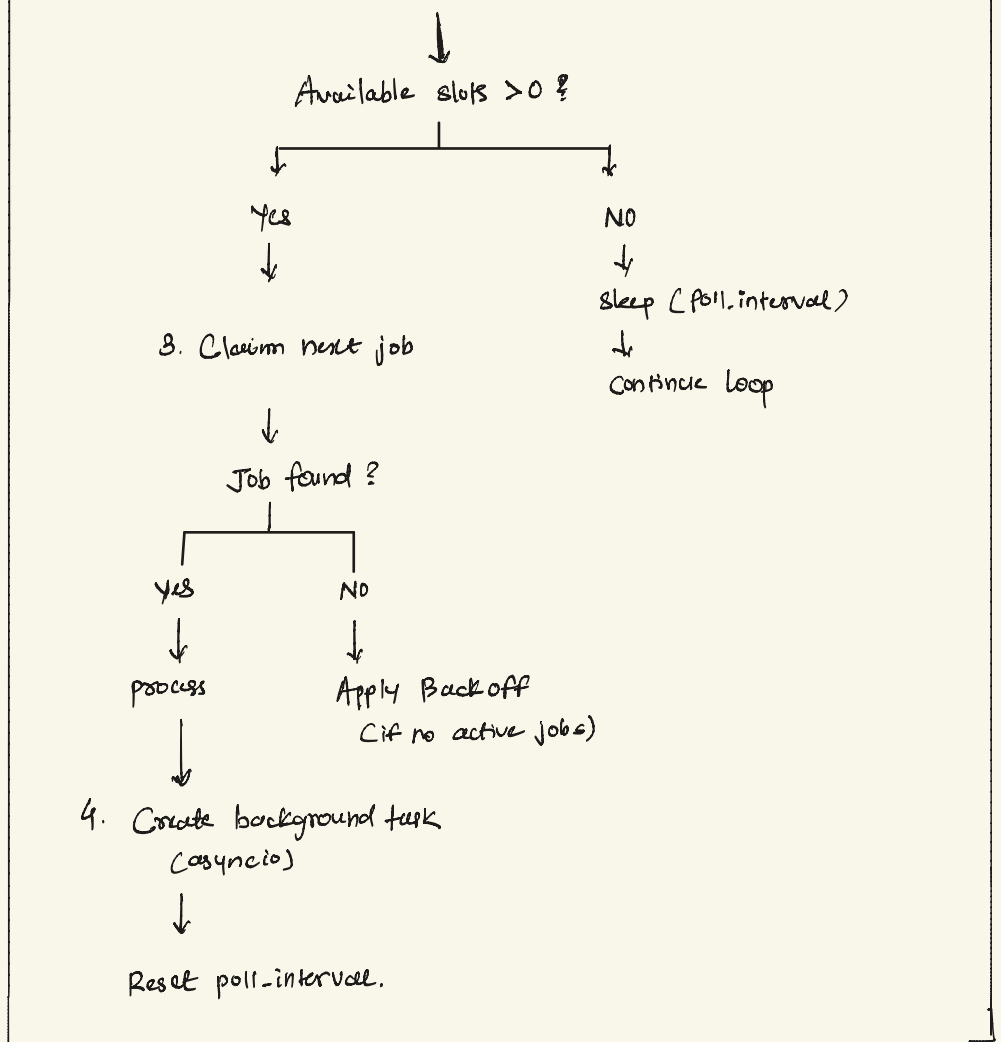API Request → FastAPI Endpoint

↓

Job inserted into DB

↓

Return job Id to client

Worker processing flow

Start worker

↓

Initialize

- Database connection
- Logger setup
- Load configuration
- Register Signal

↓
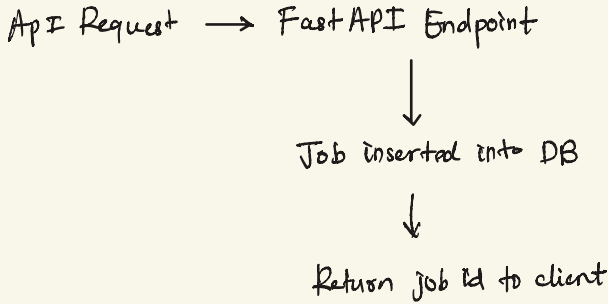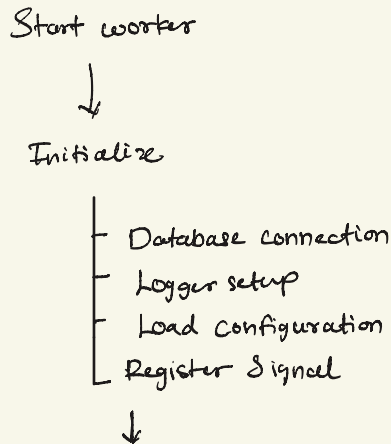
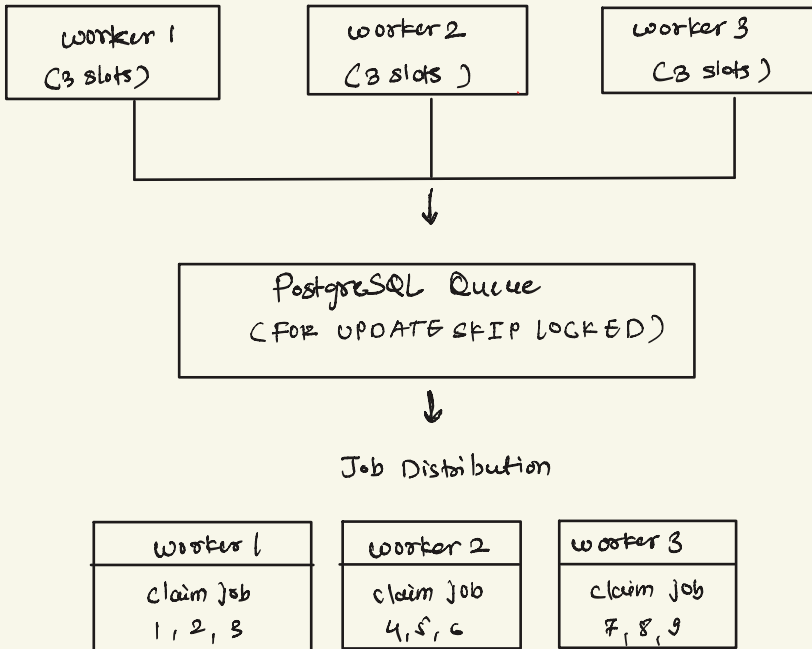## Main Processing Loop

1. Cleanup completed tasks

2. Check available slots
   available = MAX - CONCURRENT - len(active -jobs)

Available slots > 0 ?

Yes → 3. Claim next job

NO → Sleep (poll.interval) → Continue loop

Job found ?

Yes → Process → 4. Create background task (asyncio) → Reset poll-interval.

NO → Apply Back off (if no active jobs)

Shutdown signal Received ?
→ Yes
→ Graceful shutdown
- stop claiming new jobs
- wait for active jobs
- cancel remaining jobs
- close DB connection

worker 1 (3 slots)	worker 2 (3 slots)	worker 3 (3 slots)

↓

PostgreSQL Queue
(FOR UPDATE SKIP LOCKED)

↓

Job Distribution

worker 1	worker 2	worker 3
claim Job 1, 2, 3	claim Job 4, 5, 6	claim Job 7, 8, 9

==CRITICAL DESIGN PATTERNS==

1. Atomic job claiming
2. Background tasks (asyncio)
3. Exponential Backoff
4. Graceful shutdown.