📌Persistent Reference: Backend Skill Upgrade (10-Day Plan)

This document is the **single source of truth** for this conversation and future messages. It exists so context is never lost as the chat grows.

User Profile (Technical Context)

- **Primary focus:** Backend Engineering (Node.js)
- **Secondary:** Backend + DevOps (light, practical)
- **Not focusing on:** Frontend-heavy work, ML/AI, blogs/docs
- **Daily time available:** 4–5 hours
- **Market context:** Bangladesh job market → backend + deployment + infra basics > ML hype

---

Current Skill Baseline

Comfortable With

- Node.js + Express
- REST APIs
- CRUD applications
- JWT authentication
- Role-based access control
- File uploads (Cloudinary)
- MySQL & MongoDB (conceptually + practically)

Partial / Rusty / Theoretical

- Redis (basic concepts, little real usage)
- WebSockets (learned before, rusty)
- Docker (dockerized once, forgot details)
- CI/CD (theory, no ownership-level hands-on)

Missing Experience

- Running a backend system in production
- Async job systems with workers
- Owning deployment on a VM
- Connecting infra pieces into one system

💣Core Project (Locked)

Project Type

Production-Style Backend API Platform

Base

- Use **existing hackathon repository** as a foundation
- Reframe from "hackathon submission" → **owned backend service**

Domain

Async File Processing & Download API

Core Idea

> HTTP is for coordination, not work.

Long-running tasks are handled asynchronously using workers.

🧰Tech Stack (Final)

- Runtime: Node.js
- Framework: Express / Hono (from repo)
- Queue: Redis + BullMQ
- Workers: Separate Node.js worker process
- Storage: MinIO (S3-compatible)
- Caching & Rate Limiting: Redis
- Containerization: Docker + Docker Compose
- CI/CD: GitHub Actions (light, real)
- Deployment: Existing VM (Docker-based)
- Observability: Minimal (metrics > tracing)

🗄️Explicit Constraints (Do NOT Drift)

- ❌No React / frontend UI
- ❌No deep OpenTelemetry / Jaeger rabbit holes
- ❌No SRE-level infra (read replicas, partitioning, etc.)
- ❌No fake scaling claims

🕐End Goal (Day 10 Outcome)

By Day 10, the system should:

- Respond instantly to API requests (<200ms)
- Process long-running jobs via background workers
- Persist job status reliably
- Store results in MinIO and serve via presigned URLs
- Be fully dockerized
- Be deployed on a real VM with a public endpoint
- Have CI checks running on every push

And the user should be able to **explain every design decision calmly in an interview**.

🗓️10-Day Roadmap (High-Level)

- **Day 1:** Remove long HTTP requests → async jobs
- **Day 2:** Job lifecycle, retries, persistence
- **Day 3:** MinIO integration (proper storage usage)
- **Day 4:** Dockerization (API + Worker + Redis + MinIO)
- **Day 5:** Deployment on VM (ownership moment)
- **Day 6:** Redis caching + rate limiting
- **Day 7:** Minimal monitoring (metrics only)
- **Day 8:** Small-scale load testing & tuning
- **Day 9:** CI/CD refinement
- **Day 10:** Polish, README, interview readiness

Guiding Principle (Always Valid)

> Build ONE system. Understand it deeply. Own it end-to-end.

This document should be treated as **authoritative context** for all future guidance in this conversation.