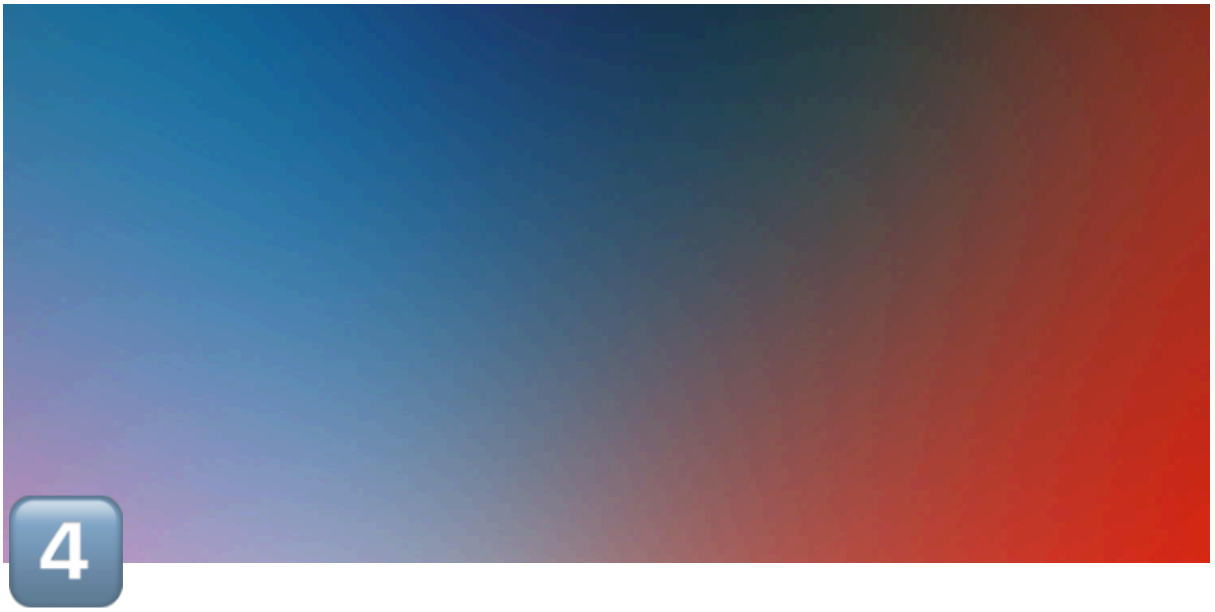

🐐 Phase 4 — Advanced Backend Engineering

Duration: 3 Months (±12 Weeks)

🎯 Goal

Master **scalability, performance, reliability, and security** by learning caching, async processing, messaging systems, file storage, rate limiting, and production-grade API design.

Build a **Video Sharing Platform (Backend-Only)** that behaves like a real system.

🗓️ Roadmap Phase 4 — Weekly Breakdown

📌 Month 1 — Performance, Caching & API Design

🗓️ Week 1: Advanced API Design Principles 🧠

1. Design **clean, consistent REST APIs**:
    - Resource naming
    - Versioning (`/api/v1`)

2. Idempotency (safe retries).
3. Pagination strategies review:
    - Offset vs cursor
4. Error response standardization.
5. API documentation best practices.

🗓️ Week 2: Caching Fundamentals (Redis) ⚡

1. Understand **why caching is needed**.
2. Learn Redis basics:
    - Key–value storage
    - TTL
3. Cache patterns:
    - Cache-aside
    - Write-through (conceptual)
4. Decide **what should / should not be cached**.
5. Integrate Redis into backend project.

🗓️ Week 3: Advanced Caching Strategies 🔥

1. Cache invalidation strategies.
2. Handle stale data.
3. Cache pagination & search results.
4. Prevent cache stampede.
5. Monitor cache hit/miss ratio.

🗓️ Week 4: Performance & Database Optimization 📈

1. Identify performance bottlenecks.
2. Query optimization techniques.
3. Indexing strategies (review).
4. Avoid N+1 queries (deep dive).

5. Measure performance impact.

📌 Month 2 — Async Processing, Messaging & File Storage

🗓️ Week 5: Asynchronous Processing Basics ⏳

1. Understand **sync vs async** workloads.
2. Identify tasks that must be async:
   - Video processing
   - Email sending
3. Background job concepts.
4. Error handling in async jobs.
5. Retry & failure handling strategies.

🗓️ Week 6: Message Queues (RabbitMQ / Kafka Basics) 📪

1. Understand message brokers:
   - Producer
   - Consumer
2. Queue vs pub-sub model.
3. Learn basic RabbitMQ or Kafka concepts.
4. Design async workflows using queues.
5. Avoid duplicate message processing.

🗓️ Week 7: File Storage Systems ☁️

1. Understand file storage challenges.
2. Integrate object storage:
   - AWS S3
   - MinIO
3. Handle file upload metadata.

4. Secure file access (signed URLs).
5. Manage large file workflows.

🗓️ Week 8: Video Processing Simulation 📹

1. Design video upload flow.
2. Store video metadata in database.
3. Trigger async processing job.
4. Simulate transcoding process.
5. Update video status:
    - Uploaded
    - Processing
    - Ready

📌 Month 3 — Security, Rate Limiting & Project Build

🗓️ Week 9: Security Hardening 🔐

1. Input validation & sanitization.
2. Protect against:
    - Injection attacks
    - Broken authentication
3. Secure API headers.
4. TLS & HTTPS awareness.
5. Secrets management best practices.

🗓️ Week 10: Rate Limiting & Abuse Prevention 🚦

1. Understand API abuse patterns.
2. Implement rate limiting:
    - Per user
    - Per IP

3. Use Redis for rate limiting counters.
4. Define fair usage policies.
5. Handle rate-limit errors properly.

🗓️ Week 11: Project Integration — Video Platform Backend 🧩

1. Integrate:
    - Auth
    - Redis caching
    - Async jobs
    - File storage
2. Cache popular videos.
3. Add pagination & search.
4. Improve performance under load.
5. Add structured logging.

🗓️ Week 12: Reliability & Final Review 🏁

1. Graceful error handling.
2. Retry strategies for failed jobs.
3. Monitor system behavior.
4. Final code refactoring.
5. Architecture & design review.

🛠️ Phase 4 Project — Video Sharing Platform (Backend Only)

Core Features

- User authentication (JWT)
- Upload video metadata
- Async video processing simulation

- Redis caching for popular videos
- Rate limiting (e.g. 100 requests/hour/user)
- Secure file storage integration
- Clean API design & documentation

🧠 Phase 4 Summary ✨

- **Month 1:** API design, caching, performance
- **Month 2:** Async processing, queues, file storage
- **Month 3:** Security, rate limiting, system integration

✅ Phase 4 Exit Criteria (Critical)

You are **Phase-4 complete** if you can:

- Design scalable backend systems
- Use Redis correctly
- Build async workflows safely
- Handle heavy API traffic
- Secure APIs properly
- Explain system architecture clearly

🔥 After Phase 4, You Are:

- **Advanced Backend Engineer**
- Ready for:
  - DevOps & cloud deployment (Phase 5)
  - Large-scale distributed systems
  - Real production environments