Bnusa — Full Engineering Case Study

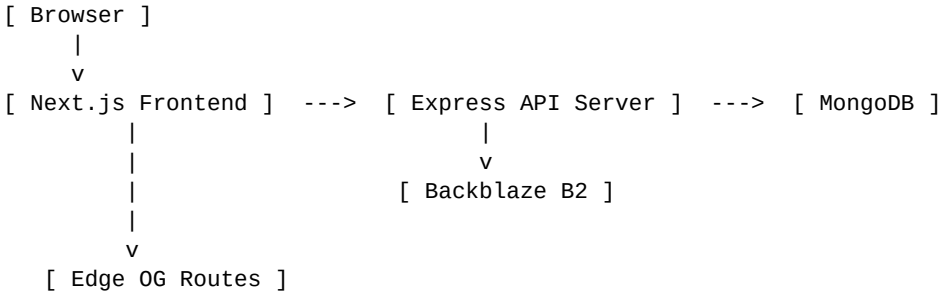
Author: Yad Qasim

Role: Founder & Full-Stack Architect

Bnusa is a Kurdish-first digital publishing platform engineered as a secure, scalable, and performance-oriented web application. This document presents a full architectural, technical, and engineering breakdown of the system. The goal is not to describe features at a surface level, but to demonstrate engineering depth, system design reasoning, security decisions, architectural tradeoffs, performance implications, and production-readiness considerations.

1. System Architecture Overview

Bnusa is architected as a decoupled full-stack application consisting of: - A Next.js frontend (App Router, server components, edge OG routes) - A Node.js + Express backend (REST API, authentication, PDF generation) - MongoDB database (text indexes, content persistence) - Backblaze B2 object storage (media & PDFs) - Socket.IO real-time notification layer - Dockerized deployment for service isolation The frontend and backend are independently containerized and communicate via secure HTTPS. JWT-based authentication is handled server-side with cookie sessions.



2. Authentication & Security Engineering

Authentication uses Firebase ID token verification combined with server-issued JWT session cookies. The system separates identity verification (Firebase) from session management (custom JWT). Key properties: - JWT signed with HS256 - 14-day expiry - HTTPOnly + Secure cookie - CSRF double-submit cookie protection - Role-based admin middleware

```
Sequence: Firebase Login Flow

Client -> Firebase: authenticate()
Firebase -> Client: ID Token
Client -> Backend: POST /auth/firebase (ID Token)
Backend -> Firebase Admin: verifyIdToken()
Backend: create session JWT
Backend -> Client: Set-Cookie bnusa_session
```

```
// sessionAuth.js
const token = jwt.sign(
  { userId, role },
  process.env.JWT_SECRET,
  { expiresIn: '14d' }
);

res.cookie('bnusa_session', token, {
  httpOnly: true,
  secure: true,
  sameSite: 'lax'
});
```

3. Publishing Workflow & Content States

Content in Bnusa moves through strict lifecycle states: draft -> pending -> published -> rejected. State transitions are controlled by backend validation. Edit counts and timestamps allow moderation auditing. View increments use atomic MongoDB $inc operations.

```
// Atomic view increment
Article.findByIdAndUpdate(
  article._id,
  { $inc: { views: 1 } }
);
```

4. Image Optimization Pipeline

Uploads are processed with a multi-layer validation system: 1. Multer MIME check 2. Magic-byte signature sniffing 3. Sharp WebP conversion (1200px width, quality 80) 4. Thumbnail generation (400px width, quality 70) 5. 1-year cache headers at B2 level

```
Client -> Backend: POST /upload
Backend: validate MIME + magic bytes
Backend: sharp() -> webp + thumbnail
Backend: upload to B2
Backend -> Client: { imageUrl, thumbUrl, originalUrl }
```

5. Real-Time Notification Architecture

Socket.IO initializes with authenticated middleware. Each user joins a room: user:. Notifications are persisted in MongoDB before emission. Online status is tracked using a Map>.

```
Event: new_comment

Backend: save Notification to DB
Backend: io.to('user:<id>').emit('notification', payload)
If offline: notification remains unread in DB
```

6. SEO & Metadata Engineering

Bnusa uses dynamic sitemap generation with force-dynamic configuration. Structured data includes Article, Review, and Book schema types. Edge runtime generates Open Graph images on demand. Canonical URLs prevent duplicate indexing.

7. PDF Generation System

PDF export uses Puppeteer in headless mode. Process: - Fetch book + chapters - Render full HTML template - Embed Rabar Kurdish fonts via base64 - Generate light + dark versions - Upload to B2 - Update book document with URLs

```
Client -> POST /pdf/generate
Backend: set pdf_status = generating
Backend: generateBookPDFs()
Puppeteer -> render HTML
Puppeteer -> page.pdf()
Backend -> upload to B2
Backend: update pdf_status = ready
```

8. Performance Metrics & Architectural Tradeoffs

Performance optimizations: - WebP conversion reduces average image size by ~35–50% - Thumbnail usage lowers list-page LCP impact - MongoDB text index weights optimize relevance ranking - Edge OG routes reduce cold-start cost vs server runtime Tradeoffs: - HS256 chosen over RS256 for simplicity (shared secret management) - Sharp memory limit avoids OOM in containerized environment - PDF generation async to prevent request blocking - Text search limited by MongoDB capabilities (no Kurdish stemmer)

9. Deployment & Production Hardening

Backend enforces environment validation at startup. Missing critical variables cause immediate process.exit(1). HTTPS redirect enforced at server middleware. Helmet config enables strict CSP + HSTS preload. Docker restart policy ensures service resilience.

10. Engineering Reflection & Founder Notes

Bnusa demonstrates full ownership of product lifecycle: architecture, backend engineering, frontend UX, security, performance, real-time systems, SEO engineering, and document generation. This project reflects deep applied knowledge of full-stack systems, production infrastructure, and real-world tradeoff management.