FlexLiving Reviews Dashboard – Project Brief

Overview

A production-ready reviews management system for FlexLiving properties. It aggregates reviews from platforms (Hostaway, Google), normalizes data, provides approval workflows, analytics, and observability. Deployed via Docker/Kubernetes with CI/CD and security scanning.

Tech Stack

- Backend
    - Node.js 20, TypeScript, Express
    - PostgreSQL + Prisma ORM; Redis caching
    - Validation with Zod; security via Helmet and express-rate-limit
    - Logging with Winston; metrics (Prometheus) and dashboards (Grafana); Sentry-ready
    - Auth: JWT (dev-mode bypass for local)
    - Testing: Jest, Supertest; rich mock data
- Frontend
    - Next.js 14 (App Router), React 18, TypeScript
    - Tailwind CSS, Radix UI primitives, shadcn-style components
    - TanStack React Query, next-themes (dark mode), lucide-react icons
- DevOps/Infra
    - Docker + docker-compose; Kubernetes manifests in k8s/
    - GitHub Actions CI/CD and security workflows
    - Postman collections and envs; scripts for setup, deploy, backups

Key Design and Logic Decisions

- Normalized Review Model
    - Unified schema across sources (Hostaway, Google) with ratings, categories, timestamps, approval, and preserved rawJson for audits.
    - Consistent response envelope: reviews, pagination, filters, meta.
- Caching Strategy (Redis)
    - Read-through caching for costly external API calls; default TTL 300s.
    - Background refresh near 80% TTL; deterministic, alphabetized query-string cache keys.
    - Manual and pattern-based invalidation endpoints.
- Approval Workflow & Auditability
    - Single and bulk approve/unapprove endpoints; optional host response text.
    - Audit history endpoint provides chronological entries; legacy approvalHistory maintained for backward compatibility.
- Security & Rate Limiting
    - JWT-based auth with role/permission checks (reviews:approve etc.); dev-mode bypass for

local.
    - IP-based rate limiting; standardized error codes and JSON error shape.
    - Standard headers: X-Request-ID, X-Response-Time, X-Cache-Status, X-Source.

- Reliability & DX
    - Health endpoints (basic/detailed), Prometheus metrics, Grafana dashboard.
    - Mock modes and realistic fixtures for development.

API Behaviors (Representative)

- Reviews
    - GET /api/reviews — rich filters (listingId, approved, channel, reviewType, min/maxRating, date range, guestName, hasResponse, search), pagination/sorting; returns reviews, pagination, filters, meta.
    - GET /api/reviews/:id — single review with listing details.
    - GET /api/reviews/:id/approval-history — current status + audit entries; approvalHistory deprecated alias.
    - PATCH /api/reviews/:id/approve l /unapprove — requires reviews:approve; optional response.
    - POST /api/reviews/bulk-approve — approve/unapprove multiple IDs; optional response.
    - GET /api/reviews/stats — totals, rating distribution, channel distribution, trends.
- Hostaway (Cached)
    - GET /api/reviews/hostaway — normalized feed with cache; format=simple for legacy consumers.
    - Admin: /hostaway/health, /hostaway/metrics, /hostaway/cache/invalidate, /hostaway/cache/stats.
- Listings
    - GET /api/listings (+includeStats), /api/listings/:id, /slug/:slug, /hostaway/:hostawayId, /search, /with-stats.
- Cross-cutting
    - Auth: Bearer token (dev bypass in development). Rate limits with 429 responses and reset hints.
    - Error format: { status, message, code, details }.

Google Reviews Findings

- Integration Paths
    - Places API (API key): search and details with up to 5 most relevant reviews per place.
        - Pros: immediate access; simple key-based auth.
        - Limits: only 5 reviews; no complete history; cannot respond; Text Search API does not support fields, Place Details does.
        - Endpoints: GET /api/reviews/google/places/search, GET /api/reviews/google/places/:placeId, POST /api/reviews/google/import/places.
    - Business Profile API (OAuth/service account): full access for owned, verified locations; can

respond to reviews.

        - Requirements: verified ownership; resource accounts/{accountId}/locations/{locationId}; proper scopes; JWT/OAuth.
        - Common errors: 401/403/404/429.

- Cost & Quotas
    - Places API: first ~1,000/month free, then ~$0.017/request; ~100 requests/100s practical limit.
    - Business Profile API: free for verified businesses; daily quotas apply.
- Compliance & Security
    - Required “Powered by Google” attribution; respect retention (e.g., avoid long-term caching of review text beyond allowed windows).
    - Keep API keys server-side; use backend proxy; validate inputs and rate-limit Google endpoints.
- Normalization & Workflow
    - Reviews normalized with source: 'google', externalId, metadata (placeId, language, author profile URLs, relative time text).
    - Deduplication and audit trail align with existing approval workflow.

Environment Configuration (Highlights)

- Backend: DATABASE_URL, REDIS_URL, JWT_SECRET, HOSTAWAY_*, GOOGLE_PLACES_API_KEY, GOOGLE_BUSINESS_PROFILE_CREDENTIALS, CACHE_DEFAULT_TTL.
- Frontend: NEXT_PUBLIC_API_URL, NEXT_PUBLIC_APP_URL, feature flags such as NEXT_PUBLIC_ENABLE_GOOGLE_REVIEWS.

Deployment & Observability

- Dockerized services; Kubernetes manifests in k8s/ with scaling and health probes.
- Monitoring via Prometheus (/metrics) and Grafana dashboards; Sentry integration optional.
- Health endpoints: /api/health, /api/health/detailed.

References

- Detailed API documentation: backend/API_DOCUMENTATION.md
- Google Reviews guide: docs/GOOGLE_REVIEWS_INTEGRATION.md
- Monitoring configs: monitoring/prometheus.yml, monitoring/grafana-dashboard.json