Flex Reviews Dashboard - Technical Documentation

Project Overview

The Flex Reviews Dashboard is a comprehensive review management system designed for vacation rental properties. It provides centralized management for reviews with advanced filtering, analytics, and bulk management capabilities.

Project Hosting & Access

Main Page:	https://flex-reviews-dashboard-mu.vercel.app/
Admin Dashboard:	https://flex-reviews-dashboard-mu.vercel.app/admin
Properties Listing:	https://flex-reviews-dashboard-mu.vercel.app/properties
Property Detail (175160):	https://flex-reviews-dashboard-mu.vercel.app/property/175160
Property Detail (128652):	https://flex-reviews-dashboard-mu.vercel.app/property/128652
Google Reviews:	https://flex-reviews-dashboard-mu.vercel.app/google-reviews

Local setup instructions are available in the README.md file

Technology Stack

Frontend

- Next.js 15.1.6 (App Router)
- React 19
- TypeScript 5
- Tailwind CSS 3.4

- Framer Motion
- Recharts (dashboard charts)
- Lucide React (icons)

Backend

- Next.js API Routes
- Supabase (PostgreSQL)
- Node.js Runtime
- SHA256 Content Hashing

Key Design & Logic Decisions

1. Database Architecture

- Separate tables for different review sources (reviews, google_reviews)
- Content hashing (SHA256) for efficient sync detection
- Normalized structure with foreign key relationships
- Status tracking (pending/approved) for review moderation

2. Sync Mechanism

- Smart sync using content hashes to detect changes
- Only updates modified reviews (reduces API calls)
- Preserves approval status during updates
- Batch processing with error resilience

3. UI/UX Design

- Tab-based navigation for different functions
- Real-time filtering without page reloads
- Bulk selection with "Select All" functionality
- Responsive design with mobile support
- Performance metrics visualization

4. Dashboard Analytics

- Interactive charts using Recharts library

- Review trends visualization over time
- Rating distribution charts
- Property performance metrics
- Response rate analytics
- Real-time data updates

5. State Management

- React hooks for local state
- Server-side rendering for initial data
- Optimistic UI updates for better UX
- Client-side caching for performance

API Behaviors & Endpoints

`GET /api/reviews`

Fetches all reviews with approval status and categories

`POST /api/ingest`

Syncs reviews from fixtures using content hashing
Returns: {processed, updated, skipped, errors}

`POST /api/reviews/[id]/approve`

Toggles review approval status
Body: {approved: boolean, approvedBy: string}

`GET /api/reviews/google`

Fetches Google reviews from database

`POST /api/reviews/google/fetch`

Syncs from external GoSign API

`GET /api/reviews/hostaway` **CRITICAL FOR TESTING**

Returns properly structured and filtered review data. Only passes approved reviews to frontend. Includes property information and metadata. Response format optimized for frontend consumption.

Hostaway API Integration

Authentication Success

Successfully authenticated with Hostaway API. Authentication screenshots are available in the "postman hostaway/" folder demonstrating proper OAuth2 implementation, token management, and refresh handling.

Implementation Steps

1. **Data Source:** Reviews data stored in JSON file (fixtures/hostaway_reviews.json)
2. **Data Ingestion:** POST /api/ingest route reads from source and saves data to database
3. **Database Operations:** Dashboard actions (approve/reject) directly modify data in database
4. **Data Filtering:** GET /api/reviews/hostaway converts to structured format optimized for frontend
5. **Approval Control:** Only approved reviews are passed to frontend through the hostaway endpoint

Google Reviews Integration - Findings

Important Note: Could not obtain Google Places API key. Google Reviews are fetched through an external API service, not through Google Places API directly.

Demonstration Data Source

- **Business:** REDDY Küchen Ansbach (Kitchen Store in Germany)
- **Google URL:** View on Google
- **Purpose:** UI/UX demonstration and functionality testing

External API Integration

Endpoint: `https://dashboard.gosign.de/review/api/{API_KEY}/reviews-list.json`

API Key Location: `.env.local` file (`REVIEWS_API_SECRET_KEY`)

Test Key: `6320d973f681d71fa0a9427d51c5b630` (for verification purposes)

Data Source: Third-party aggregation service

Implementation Details

1. Created dedicated google_reviews table in database
2. Implemented fetch endpoint at /api/reviews/google/fetch
3. Data transformation from German to English
4. Rating conversion (star ratings to numeric values)
5. Pagination with "Load More" functionality
6. Responsive card-based layout

Technical Findings

- Successfully integrated external API with error handling
- Implemented duplicate prevention using review_id
- Created update mechanism for existing reviews
- Added timestamp tracking for data freshness
- Null rating handling for incomplete reviews

Limitations

- Data is from a German business (translated to English)
- Used for demonstration of technical capability only

Critical Implementation Notes

Environment Variables

- `NEXT_PUBLIC_SUPABASE_URL` : Public Supabase endpoint

- `SUPABASE_SERVICE_ROLE_KEY` : Admin access for database
- `REVIEWS_API_SECRET_KEY` : External API authentication

Database Migrations

- Must run SQL migrations before first use
- Content_hash column is required for sync
- Status field cannot be NULL

Error Handling

- All API endpoints have try-catch blocks
- Database errors are logged with context
- User-friendly error messages returned

Performance Optimizations

- Lazy loading for review lists
- Pagination limits (20 items default)
- Client-side filtering for instant response
- Indexed database columns for quick queries