ROCKSOFT

EXAMPLE REPORT: STARTER PACKAGE

Code Audit Report: EkoMarket

Project:
EkoMarket - Mobile Marketplace MVP built with Base44

Audit Scope:
High-level code review, structure analysis, and critical risk identification

Auditors:
Rocksoft Team

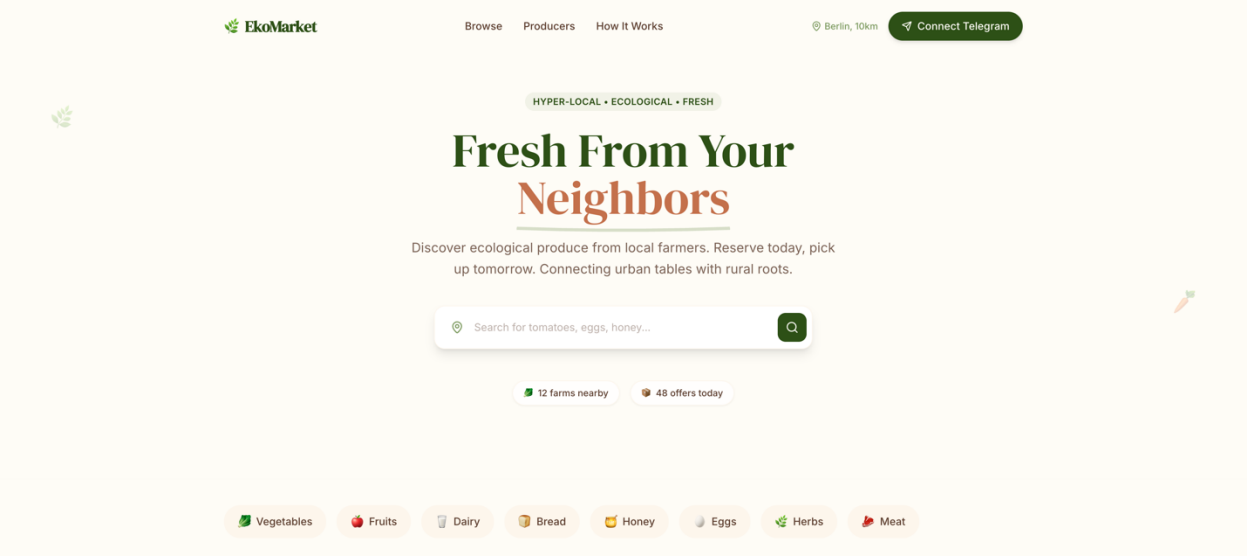
Date: February 13, 2026

Table of Contents

1. Project overview

EkoMarket is a hyper-local marketplace platform designed to connect ecological food producers with urban consumers. The core functionality allows farmers to list daily offers, while users can browse and reserve products.

A key feature of the application is its automated notification system and order management handled via a Telegram Bot.



The application was built using the Base44 AI platform. The technical stack reflects typical Base44 patterns:

- **Frontend/Backend**: Next.js (App Router) with TypeScript.
- **Styling**: Tailwind CSS.
- **Integration**: Telegram Bot API for real-time order alerts.
- **Database/Auth**: Supabase (PostgreSQL).

2. Audit objective

The purpose of this Starter Audit is to perform a quick technical validation of the EkoMarket codebase. We focused on identifying "AI-generated technical debt" typical for Base44 projects and evaluating the safety of external integrations.

We aim to determine:

- if the current codebase provides a **reliable foundation** for further development,
- to identify any **immediate technical risks** that could hinder the project's growth or security.

3. Executive summary

EkoMarket is a well-structured MVP from a visual and functional standpoint. However, the automated code generation by Base44 has introduced a critical security flaw in the Telegram integration.

Current status:
"Let's fix the foundation before you go further"

KEY FINDING:
The Telegram Bot integration exposes sensitive API credentials to the client-side, and the database rules are overly permissive.

4. Code structure and quality

4.1 Base44 architectural patterns

The project follows the standard Next.js App Router structure. While the code is generally readable, there is significant "logic duplication" between different API routes. The AI has generated separate, nearly identical fetching logic for "Products" and "Featured Products" instead of using a shared service layer.

4.2 Type safety

While TypeScript is used, many complex data objects from the Telegram API are typed as 'any'. This negates the benefits of TypeScript and increases the risk of runtime crashes when the external API structure changes slightly.

5. Critical technical red flags

5.1 Insecure Telegram integration

The Telegram Bot Token is currently stored in a client-side constant (constants/telegram.ts) rather than a secure server-side environment variable.

IMPACT:
Anyone visiting the website can extract the bot token and take full control of the EkoMarket Telegram Bot, including reading private messages or spoofing orders.

5.2 Webhook vulnerability

The endpoint receiving updates from Telegram does not verify the 'X-Telegram-Bot-Api-Secret-Token'.

> ⚠ **IMPACT:**
> An attacker can send fake order notifications directly to your backend, bypassing the actual Telegram platform.

6. Security and stability

6.1 Database Row Level Security (RLS)

Typical for rapid Base44 deployments, RLS in Supabase is disabled for the 'Orders' table to speed up development.

> ⚠ **SEVERITY:**
> **Critical**. Any user with the public API key can read or modify orders belonging to other users.

6.2 Hardcoded prompts and metadata

The codebase contains large blocks of commented-out AI prompts and Base44-specific metadata.

> ⚠ **IMPACT:**
> While not a security risk, it bloats the production bundle and makes the code harder to maintain for human developers.

7. Summary and recommendations

The EkoMarket app has high potential, but the current Telegram integration and database setup pose a direct threat to user data and system integrity.

Top 3 recommendations:

1. Move Telegram Bot logic to Next.js Server Actions or Route Handlers to hide the API Token from the browser.
- Enable Supabase RLS and define proper policies for the 'Orders' and 'Users' tables.
- Consolidate redundant API logic into a unified "Service" directory to improve maintainability.

NEXT STEP:
30-minute online call to walk you through the fixes and discuss the "Standard" path for scaling the marketplace.

➔ **Schedule a meeting**