ReelMe Code Review Audit

Cheesecake Labs

24/10/2025

Executive Summary

ReelMe is a location-based dating mobile application with a React Native mock frontend and a Ruby on Rails backend. The backend implements most of the core business logic (authentication, matchmaking, messaging, payments), but the frontend is currently entirely static and disconnected, serving only as a design prototype. As it stands, the project represents a functional backend foundation with a non-functional user interface, equivalent to a pre-MVP stage. To achieve a production-ready version, the team must complete full frontend development, integrate both layers, and implement critical infrastructure and security components.

Technologies used: React Native (App) and Ruby (API)
Current maturity: ~40% of a production MVP
Frontend readiness: Prototype only, no live features or API connectivity
Backend readiness: Core domain implemented; missing production hardening
Estimated time to achieve current state: 10 - 14 weeks
Estimated cost to achieve current state: $84,320

Frontend Main Identified Gaps:

- No backend API integration (all dummy data)
- No test coverage
- Security issues (insecure storage, no input validation)
- Minimal documentation
- Missing core features (real-time chat, payments, location)

Backend Main Identified Gaps:

- Hardcoded credentials and secrets in the code (Sidekiq and Stripe)
- Firebase private key committed
- Hardcoded OTP bypass in registration controller
- Missing authorization checks on user controller
- Missing input validation on several endpoints
- Incomplete error handling
- Magic numbers (should be constants)

ReelMe Code Review Audit

Cheesecake Labs

24/10/2025

Feature Completeness Overview

Feature	Status	Notes
Authentication	Backend-only	Not integrated in frontend Login action doesn't pass payload Sign up form submission doesn't call API - only navigates to next screen Google Sign-In: UI implemented but incomplete error handling Facebook Login: Credentials created but not sent to backend Apple Sign-In: Button present but no implementation
Profiles	Backend-only	Not integrated in frontend All profile data is hardcoded Profile gallery shows static images only
Matching System	Backend-only	Not integrated in frontend
Messaging	Backend-only	Not integrated in frontend Chat uses dummy data only No message persistence Message sending adds to local state only
Payments	Backend-only	Stripe setup, not tested No actual payment processing integration Transaction completion screen exists but no actual transaction processing
Push Notifications	Missing	Firebase placeholder only No actual implementation
Localization Map	Missing	Not implemented in frontend Screen displays a “Location” label instead of the

		feature No location services integration No map implementation No geolocation functionality
Design System	Missing	Inconsistent styling
Misc	Implementation Failures	Duplicate screen definitions in the route file Duplicate *addPhotos* and *addPhotos1* screen registrations AuthStack component defined but never used QR code scanning UI exists but no actual QR code processing logic

Frontend Code Quality Assessment

Inconsistent Naming Conventions

- Mix of camelCase and PascalCase inconsistencies
- File naming: Style.js vs Styles.js (inconsistent)
- Component naming inconsistencies

Dead/Unused Code

- AuthStack component defined but never used (*route.js* line 217-237)
- Commented-out axios interceptor (*App.js* lines 17-33)
- Multiple commented-out code blocks without context

Code Duplication

- Duplicate screen registrations in navigation
- Similar form validation patterns repeated without utilities
- Inline styles mixed with StyleSheet usage

Missing Type Safety

- TypeScript files exist (.tsx, .ts) but most code is JavaScript
- No PropTypes defined for components
- No type checking for API responses

Error Handling

- Poor: Try-catch blocks exist but errors only logged to console
- No user-friendly error messages
- No retry mechanisms
- No offline error handling

Security Concerns

- API base URL should be in environment variables (currently commented out)
- No input sanitization visible
- No rate limiting on API calls
- Token storage in AsyncStorage (should consider secure storage)
- No certificate pinning visible

Effort and Timeline Estimate

To reach a production-ready MVP, ReelMe requires approximately 10 weeks of development distributed as follows:

Frontend Development: full app logic, API integration, map features, authentication
Backend Hardening: security, rate limiting, validation
UI/UX Polish: Refine components, improve accessibility
Testing & QA: E2E and integration coverage

ReelMe Code Review Audit

Cheesecake Labs

24/10/2025

Cost Estimate To Finish

The estimated time to complete the ReelMe project is 10 weeks, covering all remaining development required to reach a production-ready MVP. This includes finalizing backend implementation, integrating the mobile application with the API, developing missing frontend features, and addressing UX and UI inconsistencies identified during review. Additionally, the scope encompasses the creation of an admin panel for moderation and operational control. To achieve this, the recommended team structure consists of one designer, one product manager, one DevOps engineer, and two software engineers working collaboratively to deliver the complete and stable product.

Conclusion

ReelMe has a strong backend foundation but remains an incomplete product due to its static, non-integrated frontend and missing operational components. To reach production readiness, the project requires finalizing backend functionalities, integrating the mobile app, developing missing user features, refining UX and UI, and building an admin panel for moderation. With an estimated development timeline of 10 weeks. The current codebase cannot yet serve users or enter testing.