alpine

CASE STUDY

Money that can’t silently disappear

A Swedish consumer fintech for young users

THE STAKES

A money app for kids and teens: parents watching, regulators watching — the least forgiving audience in fintech. A refund that silently vanishes is a lost family and a compliance case. We ran the backend money infrastructure for five years.

WHAT WE DID — GIT-VERIFIED

The numbers are in the repository history:

- **#1 contributor to the codebase**: 6,836 commits over the engagement — more than any other engineer on the product — sustained at 1,300–2,500 commits per year
- **~560 commits on KYC and payment integrations**: Onfido, Instantor, Klarna, Stripe, Swish, PrepaidFinancialServices
- **202 database migrations** authored on a live financial product
- Designed the **transaction-failure handling**: a terminal CRITICAL_FAIL state, refund retries moved into resilient async tasks, payment-request creation wrapped in transactions — money states engineered so they cannot silently vanish
- Owned databases, cloud infrastructure, deployments and telemetry; ran anomaly response when something moved wrong

OUTCOMES

- Identity and payment rails built and operated to compliance-grade on a consumer money product, across five years of continuous delivery.
- Failure paths designed so the worst case is a visible, recoverable state — never a silent loss.
- The claim is auditable: almost nobody selling engineering services can say “count the commits.” We can.

STACK

Python · AWS · Stripe · Swish · Klarna · Onfido · Instantor · PrepaidFinancialServices

This is the engineering standard behind Alpine’s fixed-price delivery audit: findings with file-and-line evidence, an implementation playbook, and an execution-ready backlog your team can run with. References on request.

Alpine LTD · contact@alpine-engineering.eu · alpine-engineering.eu · linkedin.com/company/138193958