CONTEXT.md — LogSaaS Lite

■ Purpose

LogSaaS Lite is a multi-tenant log monitoring + alerting system.
Think tiny Logtail / Sentry built for teams:

- Capture logs from applications via API key
- Store logs (structured JSON) in MongoDB
- Query/filter logs in dashboard (React)
- Define rules → alerts (email, Slack webhook)
- Multi-tenant (each user/team has their own logs)

■■ Architecture Overview

High-Level
Client (React + Firebase)
|
v
API Gateway (Express + TS)
|
v
MongoDB (Atlas) — Logs, Users, API Keys, Rules
|
v
Worker (Node cron/queue) → Alerts (email/Slack)

Low-Level (Sequence: log ingestion)
App → [POST /logs w/ API_KEY] → Express → Auth middleware → Mongo (logs collection)

Auth Flow
React → Firebase Auth (Google/Email) → JWT → Express API → Verify Token → Mongo

■ Repository Layout

```
logsaas-lite/
 ■■ apps/
 ■   ■■ api/   # backend
 ■   ■   ■■ src/
 ■   ■   ■   ■■ index.ts         # entrypoint
 ■   ■   ■   ■■ config/          # env, db connect
 ■   ■   ■   ■■ middlewares/     # auth, error
 ■   ■   ■   ■■ routes/          # /auth, /logs, /apikeys, /rules
 ■   ■   ■   ■■ models/          # mongoose schemas
 ■   ■   ■   ■■ services/        # business logic
 ■   ■   ■   ■■ workers/         # alert worker
 ■   ■   ■   ■■ utils/           # helpers (logger, mail, slack)
 ■   ■■ web/   # frontend
 ■       ■■ src/
 ■       ■   ■■ pages/          # dashboard, settings, login
 ■       ■   ■■ components/     # charts, tables, alert forms
 ■       ■   ■■ hooks/          # useLogs, useApiKeys
 ■       ■   ■■ theme/          # shadcn/ui + dark mode config
 ■■ packages/
 ■   ■■ config/    # eslint, prettier
 ■   ■■ tsconfig/
 ■■ CONTEXT.md     # this file
 ■■ turbo.json
 ■■ pnpm-workspace.yaml
```

■■ README.md

■ Workflows

1. Authentication
- User signs in (Firebase) → gets JWT
- React stores token → attaches to Axios requests
- API validates Firebase token → attaches req.user

2. API Key Management
- User creates API key in dashboard
- API key stored in Mongo (hashed, tied to userId)
- Logs must include header x-api-key → backend resolves user

3. Log Ingestion
- POST /logs (body = { level, message, meta })
- Middleware → validate API key → attach tenant
- Insert into Mongo time-series collection

4. Query Logs
- GET /logs?level=error&since;=1h
- Query Mongo with filters, return JSON
- React shows in table + chart

5. Alerts
- User defines rule: { level: error, frequency: 5 in 10m }
- Worker runs cron every 1m → scans logs → triggers email/Slack

■ Backend Skeleton

Example index.ts

```
import express from 'express';
import cors from 'cors';
import { connectDB } from './config/db';
import authRoutes from './routes/auth';
import logRoutes from './routes/logs';
import apiKeyRoutes from './routes/apikeys';
import ruleRoutes from './routes/rules';

const app = express();
app.use(cors());
app.use(express.json());

app.use('/auth', authRoutes);
app.use('/logs', logRoutes);
app.use('/apikeys', apiKeyRoutes);
app.use('/rules', ruleRoutes);

connectDB().then(() => {
  app.listen(4000, () => console.log('API running on :4000'));
});
```

Example models/Log.ts

```
import { Schema, model } from 'mongoose';

const LogSchema = new Schema({
  tenantId: { type: String, required: true },
  level: { type: String, enum: ['info','warn','error'], required: true },
  message: { type: String, required: true },
  meta: { type: Object },
```

```
  createdAt: { type: Date, default: Date.now }
});

export default model('Log', LogSchema);
```

■ Frontend Plan

UI Stack
- React + Vite + TS
- shadcn/ui (buttons, tables, modals)
- framer-motion (page transitions, toast animations)
- dark mode (tailwind + next-themes pattern)
- react-query (API hooks)
- Chart.js (line graph of log counts per min/hour)

Example Dark Mode Toggle

```
import { useTheme } from 'next-themes';

export function ThemeToggle() {
  const { theme, setTheme } = useTheme();
  return (
    <button onClick={() => setTheme(theme === 'dark' ? 'light' : 'dark')}>
      {theme === 'dark' ? '■' : '■■'}
    </button>
  );
}
```

■ Contribution Guide

```
- Always run pnpm lint && pnpm format before PR
- Do not rename existing files/functions without agreement
- New features → add test stubs if possible
- Write commit messages in present tense (“add alerts route”)
```