LimeChat MBR Generator — Setup Guide

What it is: A localhost tool that pulls live metrics from LimeChat Analytics + Metabase, runs AI analysis via Groq (Llama 3.3 70B), and generates a structured Monthly Business Review for CSM-to-client sharing. Output: formatted report, PDF export, copy-to-Gmail email.

Files

CSM Complete/

├── limechat_mbr_server.py # Python proxy server (port 3000)

└── limechat_mbr.html # Single-page frontend

No npm, no pip installs. Uses Python 3 stdlib only.

Setup

1. Configure secrets

Open `limechat_mbr_server.py` and set these constants at the top:

```
PORT  = 3000

# LimeChat Analytics API

LC_API_TOKEN = "YOUR_LC_API_TOKEN"

# Metabase (internal BI)

METABASE_KEY = "YOUR_METABASE_API_KEY"

# AI backend — Groq (free tier, get key at console.groq.com)

GROQ_KEY  = "gsk_..."

GROQ_MODEL  = "llama-3.3-70b-versatile"
```

```
# Optional: email sending via Gmail SMTP
SMTP_EMAIL    = "your@gmail.com"
SMTP_PASSWORD = "your_app_password"   # Gmail App Password, not account password
# CRM base URL (do not change)
CRM_BASE = "https://app.limechat.ai"
```

> **Groq free tier:** 14,400 req/day, no credit card needed. Get key at console.groq.com.

2. Run the server

```
python3 limechat_mbr_server.py
```

Expected output:

```
──────────────────────────────────────────────────────────
  🟢 LimeChat MBR Server v7 on port 3000
  📊 Open: http://localhost:3000/limechat_mbr.html
  🧠 AI backend: Groq / Llama 3.3 70B (free tier)
  🔑 Groq key: ✅ Key set
──────────────────────────────────────────────────────────
```

3. Open in browser

```
http://localhost:3000/limechat_mbr.html
```

How It Works

```
Browser (limechat_mbr.html)
    |
```

```
├── GET /limechat_mbr.html          → serves the SPA
│
├── POST /lc-api/{endpoint}         → proxies to LimeChat Analytics API
│       bot_performance, csat, conversation_overview,
│       agent_performance, sales_analytics
│
├── POST /metabase/{card_id}        → proxies to Metabase card API
│       Cards: 3143, 3144, 3262, 3263, 3264, 3267, 3268, 3269, 3270
│
├── POST /crm-sample                → fetches recent conversations from
│       LimeChat CRM (uses Auth0 JWT auto-extracted from browser
│       localStorage — no manual token needed, 24h expiry)
│
└── POST /claude-full               → calls Groq AI with full metrics,
        diagnosis engine output, and LimeChat knowledge base,
        returns structured JSON:
        {executiveSummary, keyObservations, nextSteps}
```

AI Pipeline

1. **Metric diagnosis** (pure Python, no API) — `_diagnose_metrics()` classifies each metric as broken / unused / working before calling AI
2. **LimeChat knowledge base** — embedded constant with features, benchmarks, problem→actionable mappings
3. **Groq call** — sends diagnosis + raw metrics + knowledge → gets structured JSON back

4. **Structured output** — `nextSteps` split into `fix` (what's broken) and `activate` (unused features)

API Dependencies

Service	Auth method	Used for
LimeChat Analytics API	Bearer token (`LC_API_TOKEN`)	Bot, helpdesk, CSAT, sales metrics
Metabase	API key header (`METABASE_KEY`)	Broadcasts, flows, intents, complaints
LimeChat CRM	Auth0 JWT (auto from localStorage)	Conversation sampling for AI context
Groq	Bearer token (`GROQ_KEY`)	AI report generation
Gmail SMTP	App password	Email sending (optional)

Environment Variables (for k8s/prod deployment)

The server currently reads from hardcoded constants. For production, recommend replacing with `os.environ.get()`:

LC_API_TOKEN = os.environ.get("LC_API_TOKEN", "")

METABASE_KEY = os.environ.get("METABASE_KEY", "")

GROQ_KEY   = os.environ.get("GROQ_KEY", "")

SMTP_EMAIL   = os.environ.get("SMTP_EMAIL", "")

SMTP_PASSWORD = os.environ.get("SMTP_PASSWORD", "")

Suggested k8s secret keys:

LC_API_TOKEN

METABASE_KEY

GROQ_KEY

SMTP_EMAIL

SMTP_PASSWORD

Ports & Network

- Server listens on `0.0.0.0:3000` (TCPServer with `allow_reuse_address = True`)
- All external API calls are outbound HTTPS from the server
- The HTML frontend calls only `localhost:3000/*` — no external calls from browser
- CORS headers are set for `*` (safe for internal tooling)

Known Limitations

- **CRM enrichment** requires the CSM to be logged into `app.limechat.ai` in the same browser — Auth0 token is pulled from `localStorage` automatically. Token expires every 24h.
- **No persistent storage** — report data lives in the browser session only
- **Single-threaded server** with `ThreadPoolExecutor` for parallel CRM fetches; not designed for concurrent users

---

Secrets to Remove Before Sharing

The following are currently hardcoded in `limechat_mbr_server.py` — replace with env vars or placeholders before committing to any repo:

- `LC_API_TOKEN`
- `METABASE_KEY`
- `ANTHROPIC_KEY` (unused, out of credits)
- `GEMINI_KEY` (unused, quota issues)
- `GROQ_KEY`

Built by Drishti (CSM Team) · LimeChat AI Initiatives · May 2026