Leadstorm AI Development Handoff Document (Google Places Edition)

## **1. Project Overview**

This document describes requirements and an implementation plan for **Leadstorm AI v2**, a lightweight scraper tool that delivers up to **50 fresh email leads per day** using the Google Maps Places API and straightforward HTML scraping—**no Hunter.io or other enrichment services**. It runs as a forkable Replit template so WarriorPlus buyers can run it in IDE/preview mode at zero hosting cost.

### **Goals**

* **50 unique email leads/day**: count only records where an email is successfully scraped.
* **Zero-cost**: leverage Google’s $200/month Places API credit; no paid email APIs.
* **Simplicity**: one free Replit fork, add API key, click Run (or schedule via UptimeRobot).
* **Reliability**: enterprise-grade Google data with retry/backoff.
* **Local dedupe**: SQLite datastore ensures no repeats across days.
* **User-friendly export**: CSV download of all leads for import into any spreadsheet.

## **2. Tech Stack & Dependencies**

**Runtime:** Node.js 18+ on Replit

**Core Packages:**

****{

"name": "leadstorm-ai-v2",

"version": "2.0.0",

"main": "index.js",

"scripts": { "start": "node index.js" },

"dependencies": {

"express": "^4.x",

"axios": "^1.x",

"better-sqlite3": "^8.x",

"dotenv": "^16.x"

}

}

**Optional:** node-cron (for local scheduling tests).

## **3. Folder Structure**

****/ (root)

|-- index.js # Express server, /run and /download endpoints

|-- leadJob.js # runLeadJob(city, keyword): core orchestration

|-- db.js # SQLite wrapper: leads table, dedupe functions

|-- scraper.js # scrapeHTMLForEmail(url) via regex

|-- places.js # fetchPlaces(city, keyword) via Google Places Text Search

|-- config.js # Load env vars: GOOGLE\_PLACES\_KEY

|-- .env.example # Example: CITY, KEYWORD, GOOGLE\_PLACES\_KEY, DAILY\_CAP

|-- package.json

|-- README.md # Fork, add secret, run instructions

|-- data.sqlite # Auto-created SQLite DB

## **4. Environment & Secrets**

In Replit Secrets (🔑):

* GOOGLE\_PLACES\_KEY – your Google Maps API key with Places API enabled.
* Optional CITY and KEYWORD for default-run mode.
* DAILY\_CAP – default 50 (for testing, set lower).

### **Google Maps Setup Summary**

1. Create Google Cloud project, enable billing (free $200/mo credit).
2. Enable **Places API** under **APIs & Services → Library**.
3. Create **API Key** under **Credentials**; copy to GOOGLE\_PLACES\_KEY.
4. (Optional) Restrict key to HTTP referrers.

## **5. Detailed Implementation Steps**

### **5.1 Database Layer (db.js)**

* Use better-sqlite3 to open or create data.sqlite.
* Create table leads:

CREATE TABLE IF NOT EXISTS leads (

id INTEGER PRIMARY KEY,

place\_id TEXT NOT NULL,

phone TEXT,

name TEXT,

email TEXT,

UNIQUE(place\_id, phone)

);

*   
  Expose:  
  + isExistingLead(placeId, phone) -> boolean
  + addLead({ place\_id, phone, name, email })
  + getAllLeads() -> Array (for CSV export)

### **5.2 Configuration (config.js)**

****require('dotenv').config();

module.exports = {

GOOGLE\_PLACES\_KEY: process.env.GOOGLE\_PLACES\_KEY,

CITY: process.env.CITY,

KEYWORD: process.env.KEYWORD,

DAILY\_CAP: parseInt(process.env.DAILY\_CAP || '50', 10)

};

Validate presence of GOOGLE\_PLACES\_KEY on startup; exit with clear error if missing.

### **5.3 Places Fetcher (places.js)**

****const axios = require('axios');

const { GOOGLE\_PLACES\_KEY } = require('./config');

async function fetchPlaces(city, keyword) {

const query = encodeURIComponent(`${keyword} in ${city}`);

const url = `https://maps.googleapis.com/maps/api/place/textsearch/json?query=${query}&key=${GOOGLE\_PLACES\_KEY}`;

const resp = await axios.get(url);

return resp.data.results.map(r => ({

place\_id: r.place\_id,

name: r.name,

address: r.formatted\_address,

// optional: we can fetch phone & website via Place Details if needed

}));

}

module.exports = { fetchPlaces };

### **5.4 Email Scraper (scraper.js)**

****const axios = require('axios');

const emailRegex = /[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}/gi;

async function scrapeHTMLForEmail(url) {

try {

const html = (await axios.get(url, { timeout: 10000 })).data;

const matches = html.match(emailRegex);

return matches ? matches[0] : null;

} catch {

return null;

}

}

module.exports = { scrapeHTMLForEmail };

### **5.5 Core Job Logic (leadJob.js)**

****const { fetchPlaces } = require('./places');

const { scrapeHTMLForEmail } = require('./scraper');

const db = require('./db');

const { CITY, KEYWORD, DAILY\_CAP } = require('./config');

async function runLeadJob(city = CITY, keyword = KEYWORD) {

let added = 0, dupes = 0, noemail = 0;

const places = await fetchPlaces(city, keyword);

for (const biz of places) {

if (added >= DAILY\_CAP) break;

if (db.isExistingLead(biz.place\_id, biz.phone)) { dupes++; continue; }

const email = biz.website

? await scrapeHTMLForEmail(biz.website)

: null;

if (!email) { noemail++; continue; }

db.addLead({ place\_id: biz.place\_id, phone: biz.phone, name: biz.name, email });

added++;

}

console.log(`Run summary: added ${added}, duplicates ${dupes}, no-email ${noemail}`);

return { added, dupes, noemail };

}

module.exports = { runLeadJob };

### **5.6 Server & Endpoints (index.js)**

****const express = require('express');

const { runLeadJob } = require('./leadJob');

const db = require('./db');

const app = express();

// Trigger run

app.get('/run', async (req, res) => {

const city = req.query.city; const keyword = req.query.keyword;

if (!city || !keyword) return res.status(400).send('Missing city or keyword');

const summary = await runLeadJob(city, keyword);

res.json({ status: 'success', summary });

});

// Download CSV

app.get('/download', (req, res) => {

const leads = db.getAllLeads();

const header = ['Date','Name','Email','Phone','Website','Address'];

const rows = leads.map(l => [l.date, l.name, l.email, l.phone, l.website, l.address]);

const csv = [header, ...rows]

.map(r => r.map(c => `"${(c||'').replace(/"/g,'""')}"`).join(',')).join('\n');

res.setHeader('Content-Disposition', 'attachment; filename=leads.csv');

res.setHeader('Content-Type', 'text/csv');

res.send(csv);

});

app.listen(process.env.PORT || 3000, () => console.log('Leadstorm AI v2 running'));

## **6. Testing Plan**

1. Set Secrets: GOOGLE\_PLACES\_KEY, CITY, KEYWORD, DAILY\_CAP=3 for quick test.
2. Hit /run?city=...&keyword=...: verify added, dupes, noemail counts.
3. Refresh /download: ensure CSV downloads with header + records.
4. Test dedupe: run twice, second run yields 0 added.
5. Simulate missing key: restart without GOOGLE\_PLACES\_KEY, app fails with clear error.

## **7. Deployment & Usage Instructions**

1. **Fork** this Repl template.
2. **Add Secret** GOOGLE\_PLACES\_KEY (no Hunter key).
3. **Run** server, then visit https://<repl>/run?city=City&keyword=Niche.
4. **Download** your CSV at https://<repl>/download.
5. **Schedule** daily with UptimeRobot pinging /run.

## **8. Error Handling & Edge Cases**

* **HTTP errors/timeout** in fetchPlaces or scraping: skip biz, log warning.
* **Missing email**: skip, increment noemail.
* **DB errors**: log and continue.
* **CSV build failure**: catch and return 500 with message.

**End of Handoff Document (Google Places Edition)**