```
/**
 * KINETIC CODEX API
 * Role: Tactical Intelligence Server
 * Stack: Cloudflare Workers + D1
 * Mode: Deep Integration (CORS, Caching, Error Handling)
 */

export interface Env {
  DB: D1Database;
}

// STANDARD HEADERS (CORS + JSON)
const corsHeaders = {
  'Access-Control-Allow-Origin': '*', // Locked to specific domain in
prod
  'Access-Control-Allow-Methods': 'GET, OPTIONS',
  'Access-Control-Allow-Headers': 'Content-Type',
  'Content-Type': 'application/json',
};

export default {
  async fetch(request: Request, env: Env, ctx: ExecutionContext):
Promise<Response> {
    const url = new URL(request.url);

    // 1. OPTIONS HANDLER (Pre-flight checks)
    if (request.method === "OPTIONS") {
      return new Response(null, { headers: corsHeaders });
    }

    // 2. ROUTER LOGIC
    try {
      // ENDPOINT: /codex/search?q=term
      if (url.pathname.startsWith('/codex/search')) {
        const query = url.searchParams.get('q');
        if (!query) throw new Error("Search query required");

        const results = await env.DB.prepare(
          `SELECT * FROM concepts WHERE term LIKE ? OR definition LIKE
? LIMIT 10`
        ).bind(`%${query}%`, `%${query}%`).all();

        return jsonResponse(results.results);
      }

      // ENDPOINT: /codex/:term (Deep Lookup with Relationships)
      if (url.pathname.startsWith('/codex/')) {
        const term = decodeURIComponent(url.pathname.split('/').pop()
```

```
|| "");

        // Parallel Query Execution for Speed
        const [conceptInfo, relationships] = await Promise.all([
          env.DB.prepare('SELECT * FROM concepts WHERE term =
?').bind(term).first(),
          env.DB.prepare('SELECT * FROM concept_relationships WHERE
source_term = ? OR target_term = ?').bind(term, term).all()
        ]);

        if (!conceptInfo) return new Response("Concept Not Found", {
status: 404, headers: corsHeaders });

        // Format the "Tactical Graph"
        const payload = {
          identity: conceptInfo,
          tactical_web: relationships.results.map((rel: any) => ({
            role: rel.source_term === term ? "ACTIVE (Source)" :
"PASSIVE (Target)",
            related_entity: rel.source_term === term ? rel.target_term
: rel.source_term,
            type: rel.relationship_type,
            rationale: rel.rationale
          }))
        };

        return jsonResponse(payload);
      }

      // ENDPOINT: / (Index - Return All Categorized)
      const allConcepts = await env.DB.prepare('SELECT term, category
FROM concepts ORDER BY category, term').all();

      // Grouping Logic for Clean Output
      const library = allConcepts.results.reduce((acc: any, curr: any)
=> {
        if (!acc[curr.category]) acc[curr.category] = [];
        acc[curr.category].push(curr.term);
        return acc;
      }, {});

      return jsonResponse(library);

    } catch (err) {
      // Error Boundary
      return new Response(JSON.stringify({ error: (err as
Error).message }), { status: 500, headers: corsHeaders });
    }
```

```
  },
};

// Helper: Response Formatter
function jsonResponse(data: any, status = 200) {
  return new Response(JSON.stringify(data, null, 2), {
    status,
    headers: {
      ...corsHeaders,
      'Cache-Control': 'public, max-age=3600' // Edge Cache for 1 Hour
to save D1 costs
    }
  });
}
```