# **System Architecture Design for Linka E-Commerce Platform**

## **Overview**

Linka is an e-commerce platform for SMEs in Zambia, utilizing agentic AI and ML for communications, account monitoring, and operations management. It comprises four components: shop-application (Docker container), main server (proxy on Google Cloud), client-application, and delivery system.

## **Components**

### **1. Shop-Application (Docker Container)**

* **Stack**: Node.js (v20.x), Express.js, MongoDB (v7.x).
* **Minimum Requirements**:
  + CPU: 1 vCPU.
  + RAM: 512 MB.
  + Storage: 2 GB.
  + OS: Alpine Linux v3.19.
  + Port: 3000 (HTTP).
* **Endpoints**:
  + GET /catalog: Returns product catalog.
  + POST /order: Processes orders.
  + GET /payment-form?orderId={orderId}: Generates payment form.

### **2. Main Server (Proxy on Google Cloud)**

* **Stack**: Python (v3.12), FastAPI, PostgreSQL (v16.x).
* **Specifications**:
  + Instance: e2-micro (2 vCPUs, 1 GB RAM).
  + Storage: 10 GB SSD.
  + OS: Ubuntu 22.04 LTS.
  + Networking: Public IP, ports 80/443.
  + Autoscaling: 1-5 instances, CPU >70%.
* **Endpoints**:
  + POST /shops/discover: Geospatial query for shops.
  + POST /order/relay: Relays orders.
  + POST /delivery/request: Manages delivery requests.

### **3. Client-Application**

* **Stack**: React.js (v18.x), Vite, Tailwind CSS, hosted on Netlify.
* **Communication**: REST API via Axios to proxy.

### **4. Delivery System**

* **Stack**: Assumed third-party REST API.
* **Endpoints**:
  + POST /delivery: Accepts delivery requests.
  + POST /delivery/status: Updates delivery status.

## **Workflow**

1. **Shop Discovery**:
   1. Client sends location to proxy (POST /shops/discover).
   2. Proxy queries PostgreSQL with PostGIS: SELECT \* FROM shops WHERE ST\_DWithin(location, ST\_MakePoint(:lat, :long), 10000) AND verified = true.
   3. Returns shop list.
2. **Catalog Retrieval**:
   1. Client requests catalog (GET /shops/{shopId}/catalog).
   2. Proxy forwards to shop-application (GET /catalog).
   3. Shop responds, proxy relays to client.
3. **Order Placement**:
   1. Client sends order (POST /order).
   2. Proxy forwards to shop (POST /order).
   3. Shop returns payment form.
4. **Payment Processing**:
   1. Proxy forwards payment form to client.
   2. Client uses MTN Mobile Money API for payment.
   3. Confirmation relayed to shop.
5. **Delivery Coordination**:
   1. Shop sends delivery request (POST /delivery/request).
   2. Proxy confirms with client, re-requests location, forwards to delivery system.
   3. Delivery system updates proxy on completion.

## **Similar Projects**

* **Jumia (Africa)**: E-commerce platform with centralized server and regional proxies for geolocation-based vendor discovery.
* **Glovo (Global)**: Marketplace with proxy API gateway, microservices, and third-party delivery integration.

## **Security and Scalability**

* **Security**: HTTPS, JWT authentication, rate limiting.
* **Scalability**: Google Cloud autoscaling, Kubernetes for Docker, PostgreSQL read replicas.