QUDIL – SaaS Based Hyperlocal Delivery Management
SYSTEM ARCHITECTURE & MICROSERVICE DOCUMENT

1. Executive Overview

Qudil is a **hyperlocal delivery management SaaS platform** intended to solve coordination, visibility, and cost challenges faced by:

- Independent stores and restaurants
- Store chains and corporates
- Freelance delivery agents
- Small hyperlocal delivery startups

While delivery demand is high, most operators lack a **professional delivery operating system**. Existing solutions are either too expensive or unsuitable for small operators. Qudil is designed to be:

- Affordable
- Modular
- API-first
- Operationally focused
- Suitable for multiple countries

2. Business Problem Statement

2.1 Current Challenges

- No visibility into who delivered an order
- Missing timestamps (pickup, delivery, total duration)
- Poor agent productivity tracking
- No delivery cost, expense, or income clarity
- Chaos caused by unstructured freelancer usage

2.2 Market Gap

- Existing solutions are **cost-prohibitive** for SMEs
- Freelancers and small delivery startups lack professional software
- Merchants need a system that works **with or without accounting**

3. Project Objectives

- Provide a **hyperlocal delivery operating system**
- Support **own staff, licensed partners, and freelancers**
- Enable **quote → booking → delivery** workflow
- Offer **advance / credit-based SaaS usage**
- Integrate with ERP, POS, and retail systems
- Be scalable across cities and countries

4. Scope Of Work

The vendor is expected to deliver:

- Admin Web Application
- Merchant Application (Web / Mobile)
- Agent Mobile Application (Android mandatory)
- Consignor Mobile Application
- Backend APIs & Microservices
- Cloud deployment & documentation

5. Functional Requirements

5.1 Admin Module

System Settings

- Country / State / Province master
- City / District master
- Area span (KM radius)

Consignee Management

- Corporate (white-labelled, multi-store)
- Store Chain (multi-store)
- Single Store (single warehouse)

Freelance Agent Governance

- Independent onboarding
- Association with multiple consignees
- Approval workflow

Payment Management

- Advance payment (credit-based usage)
- Optional credit balance
- Auto service enable/ disable

Accounts & MIS

- Usage ledger
- Credit consumption
- COD summary
- Platform-level reports

5.2 Merchant Application

Warehouses / Stores

- Multiple warehouses per consignee
- Geo-coordinates mandatory

Resources

- Own staff
- Licensed agents
- Freelancers

Delivery Cost Configuration

- Rate per KM
- Volumetric pricing
- Min / Max caps
- Pickup / Delivery / Return rates
- Optional free delivery mode

Orders

- Quote generation (lat/long mandatory)
- Order booking (UI & API)
- Agent assignment priority:
  1. Own staff
  2. Licensed partners

        3. Freelancers
  - Job broadcast fallback

Order Lifecycle

  - Quoted → Booked → Assigned → Accepted → Out for Pickup → Out for Delivery → COD Collected → Delivered

API & Webhooks

  - API key management
  - Booking & quote APIs
  - Delivery status webhooks

Reports

  - Delivery performance
  - Cost & earnings
  - Agent productivity

5.3 Agent Mobile Application

  - Mobile OTP sign-in (first time)
  - Login = clock-in for work
  - Auto timeout on inactivity
  - Vehicle selection per login
  - Dashboard with:
    - Completed jobs (month)
    - Jobs in progress
    - Delivered today
    - Pending jobs
    - Distance travelled

Execution Rules

  - Single pickup – single delivery
  - Only one active job at a time

Pickup & Delivery Flow

- Navigation support
- 50m pickup geofence
- Manual pickup fallback
- OTP / signature / photo POD
- Area span validation before job completion

5.4 Consignor Mobile Application

- Public signup via mobile
- Add orders using GCN
- OTP-based order validation
- Real-time order tracking

Future Scope (Optional)

- Customer-initiated pickup
- Courier booking aggregation

6. System Architecture

6.1 Architectural Principles

- Multi-tenant SaaS
- Hyperlocal-first
- Event-driven but cost-aware
- Accounting-optional
- Offline-capable agent experience

6.2 High-Level Architecture

Clients → API Gateway + Tenant Resolver → Core Microservices → Event Bus → Data Stores

7. Microservice Map

Core Services

1. Identity & Access Service
    - OTP auth

    - RBAC
    - Clock-in / clock-out
2. Geography & Area Service
    - Country / State / City
    - Area span & distance rules
3. Tenant & Consignee Service
    - Corporate / Chain / Single store
    - Freelancer associations
4. Wallet & Credit Service
    - Advance balance
    - Credit balance
    - Usage deduction
5. Quote & Pricing Service
    - Distance calculations
    - Cost estimation
    - Quote validity
6. Order & Job Service
    - Order lifecycle
    - SLA timers
    - Status orchestration
7. Agent & Resource Service
    - Agent onboarding
    - Vehicle registry
    - Availability
8. Dispatch & Matching Service
    - Priority-based assignment
    - Distance filtering
    - Job broadcast
9. Tracking & Geo Service
    - Live GPS ingestion
    - Geofencing
    - Area validation
10. Proof of Delivery Service
    - OTP

    - Signature
    - Photo uploads
11. Notification Service
    - SMS / WhatsApp / Push
    - Event-based alerts
12. Accounting Adapter
    - Expense tagging
    - ERP exports
13. Reporting & MIS Service
    - Productivity
    - Cost & revenue analytics
14. Integration & Webhook Service
    - API keys
    - Webhooks

8. Event-Driven Backbone

Key Events

- QUOTE_CREATED
- ORDER_BOOKED
- JOB_ASSIGNED
- JOB_ACCEPTED
- PICKUP_CONFIRMED
- OUT_FOR_DELIVERY
- POD_CAPTURED
- JOB_COMPLETED
- CREDIT_DEDUCTED

9. Data Storage Strategy

- PostgreSQL (core transactional)
- Redis (live tracking)
- Time-series DB (location history)
- Object storage (POD)
- Analytics DB (MIS)

10. Deployment Expectations

MVP

- Dockerized services
- Single-region cloud
- Managed DB & Redis

Scale Phase

- Multi-region
- Kafka
- Read replicas

11. Vendor Response Expectations

Vendors should submit:

- Technical approach
- Architecture confirmation
- Delivery timeline
- Cost breakup
- Support & SLA model
- Relevant experience

12. Commercial & Legal Notes

- Milestone-based payments preferred
- No vendor lock-in
- Full IP ownership with Qudil
- 90-day post-go-live warranty

Conclusion

This requirement and architecture document defines **what to build and how it should be built,** ensuring Clear vendor understanding, Reduced ambiguity, Faster execution and Scalable long-term foundation

:: End of Document ::