**HaulHub Regional Pricing Implementation Guide**

**Overview**

This document outlines the implementation strategy for HaulHub's regional pricing system. The goal is to create a flexible, region-aware pricing structure that:

1. Automatically detects a user's region based on geolocation
2. Displays prices in local currency
3. Maintains standardized crypto payments in USDC
4. Adapts pricing parameters to local economies

**Regional Pricing Strategy**

HaulHub's pricing model must be adaptive to maintain competitiveness across vastly different markets while ensuring the platform remains viable for both haulers and users.

**Key Regional Variables**

| **Region** | **Base Rate (USD)** | **Local Currency** | **Equivalent** | **Distance Unit** | **Weight Unit** |
| --- | --- | --- | --- | --- | --- |
| Southeast Asia | $1.75-2.00 | ₱100-110 (PHP) | 3km, 5kg | Kilometers | Kilograms |
| Western Europe | $5.00-5.50 | €4.60 / £4.20 | 5km, 10kg | Kilometers | Kilograms |
| North America | $5.00 | $5.00 USD | 5mi, 10lbs | Miles | Pounds |

**Technical Implementation**

**1. Region Detection**

Implement a multi-layered approach to determine user region:

function detectUserRegion() {

// Try GPS coordinates first (most accurate)

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(position => {

const { latitude, longitude } = position.coords;

// Reverse geocode to get country

reverseGeocode(latitude, longitude)

.then(country => setUserRegion(mapCountryToRegion(country)));

}, error => {

// Fall back to IP-based geolocation

fetchIPGeolocation()

.then(ipData => setUserRegion(mapCountryToRegion(ipData.country)));

});

} else {

// Fall back to IP-based geolocation

fetchIPGeolocation()

.then(ipData => setUserRegion(mapCountryToRegion(ipData.country)));

}

}

**2. Pricing Calculation Flow**

1. User's region is detected on app launch
2. Regional config is loaded with appropriate parameters
3. User inputs (distance, weight, etc.) are processed through regional pricing formulas
4. Prices are displayed in local currency but stored as USD/USDC internally
5. Blockchain transactions occur in USDC regardless of display currency

**3. Database Structure**

regions/

└── ph/

├── name: "Philippines"

├── currency\_code: "PHP"

├── currency\_symbol: "₱"

├── exchange\_rate: 56.50

├── base\_rate: 2.00

├── distance\_increment: 0.50

├── weight\_increment: 0.75

└── ...

└── us/

├── name: "United States"

└── ...

**4. Frontend Implementation**

The UI must adapt to:

* Show correct currency symbols
* Use appropriate distance units (km vs miles)
* Use appropriate weight units (kg vs lbs)
* Display localized pricing

**Integration with Blockchain Payments**

While display prices vary by region, all blockchain transactions remain standardized:

1. Local currency amounts are converted to USD equivalent
2. USD amount is used to determine USDC payment amount (1:1 ratio)
3. Smart contract interactions always use USDC, regardless of display currency
4. Transaction history shows both local currency and USDC amounts

**Regional Badge Systems**

Region-specific badges can be implemented to incentivize adoption:

* **Southeast Asia**: "City Explorer" badges for completing favors in multiple neighborhoods
* **Europe**: "Eco Warrior" badges with higher prominence due to environmental focus
* **North America**: "Rush Champion" badges for completing time-sensitive deliveries

**Implementation Roadmap**

1. **Phase 1**: Implement region detection and basic regional pricing
2. **Phase 2**: Add currency conversion APIs and local currency display
3. **Phase 3**: Develop region-specific promotions and incentives
4. **Phase 4**: Fine-tune regional pricing based on market feedback

**Developer Guidelines**

1. Always store prices in USD internally for consistency
2. Use ISO country codes for region identification
3. Implement fallbacks for when region detection fails
4. Update exchange rates at least daily via an API
5. Round prices to logical increments for each currency (e.g., ₱5 in Philippines, $0.50 in US)
6. Test with VPNs from multiple regions to ensure correct functionality

**Monitoring and Adaptation**

1. Track regional price competitiveness against local alternatives
2. Monitor hauler earnings per region to ensure platform viability
3. Collect region-specific user feedback on pricing
4. Adjust regional parameters quarterly based on economic changes and user behavior

This regional pricing system will allow HaulHub to expand globally while remaining competitive in each local market.