**AWS SNS Phone Verification Implementation Prompts**

**Phase 1: AWS Infrastructure Setup**

**Prompt 1: AWS SNS Service Configuration**

Set up AWS SNS for phone verification with the following requirements:

1. \*\*AWS Account Setup:\*\*

- Create/configure AWS account with appropriate IAM permissions

- Set up SNS service in your primary region (us-east-1 recommended for cost)

- Enable SMS messaging with production access

- Configure CloudWatch logging for SMS delivery status

2. \*\*IAM Permissions:\*\*

- Create dedicated IAM role for SNS SMS operations

- Include permissions: sns:Publish, sns:GetSMSAttributes, sns:SetSMSAttributes

- Add CloudWatch logging permissions for monitoring

- Set up environment-specific credentials (dev/staging/prod)

3. \*\*SNS Configuration:\*\*

- Set default SMS type to "Transactional" for verification codes

- Configure spending limits to prevent unexpected charges

- Set up delivery status logging to CloudWatch

- Enable SMS usage reports for cost tracking

Include terraform/CloudFormation templates and security best practices.

**Prompt 2: Origination Identity Setup**

Configure AWS SNS origination identities for different regions:

1. \*\*United States (10DLC Setup):\*\*

- Register company information in Amazon Pinpoint console

- Create 10DLC campaign for phone verification use case

- Purchase 10DLC phone number ($1/month)

- Handle registration fees: $4 company + $10 campaign monthly

- Wait for approval process (typically 1-2 weeks)

2. \*\*International Setup:\*\*

- Configure Sender ID for supported countries (free)

- Set up long codes for countries requiring them

- Handle country-specific requirements (India, UK, etc.)

- Document origination identity mapping per country

3. \*\*Fallback Strategy:\*\*

- Implement automatic fallback to shared routes if dedicated fails

- Handle "No origination identity available" errors

- Set up monitoring for origination identity issues

Include code for dynamically selecting appropriate origination identity based on destination country.

**Prompt 3: AWS SDK Integration**

Implement AWS SNS SDK integration for your backend:

\*\*For Node.js/JavaScript:\*\*

```javascript

// Configure AWS SNS client with proper error handling

// Include retry logic with exponential backoff

// Implement phone number validation and formatting

// Add delivery status tracking

// Handle rate limiting and throttling

**For Python:**

# Set up boto3 SNS client with session management

# Implement phone number normalization

# Add comprehensive error handling for SNS-specific errors

# Include cost tracking and logging

**For Other Languages:**

* Include SDK setup for your preferred language
* Implement consistent error handling patterns
* Add logging and monitoring integration
* Include unit tests for SNS operations

Requirements:

* Support international phone number formatting
* Handle SNS-specific error codes and retry logic
* Implement cost tracking per message
* Add delivery status webhook handling

## Phase 2: Backend Implementation

### Prompt 4: SMS Service Layer

Create a robust SMS service layer that abstracts AWS SNS:

1. **Core SMS Service:**
   * Abstract SNS behind a clean interface
   * Support multiple SMS providers (AWS SNS primary, Twilio backup)
   * Implement automatic failover between providers
   * Add message templating system
2. **Phone Number Handling:**
   * Validate and format international phone numbers
   * Detect country codes and select appropriate origination identity
   * Handle special cases (US toll-free, international routing)
   * Cache country detection for performance
3. **Cost Optimization:**
   * Route messages through cheapest available path
   * Implement intelligent routing (domestic vs international)
   * Track costs per message and per campaign
   * Set up spending alerts and limits
4. **Error Handling:**
   * Handle SNS-specific errors (throttling, invalid numbers, etc.)
   * Implement exponential backoff for retries
   * Log failed messages for manual review
   * Provide meaningful error messages to frontend

Include comprehensive error codes, logging, and monitoring integration.

### Prompt 5: Verification Code Management

Implement verification code generation and validation specific to AWS SNS:

1. **Code Generation:**
   * Generate cryptographically secure 6-digit codes
   * Store codes with expiration (5-10 minutes)
   * Track attempts and implement rate limiting
   * Support code resend with progressive delays
2. **Message Composition:**
   * Create branded SMS templates with code injection
   * Keep messages under 160 characters to avoid segmentation
   * Include clear instructions and company identification
   * Add opt-out instructions for compliance
3. **Delivery Tracking:**
   * Parse SNS delivery receipts from CloudWatch
   * Track delivery status (sent, delivered, failed)
   * Handle partial failures and retry logic
   * Update user verification status based on delivery
4. **Security Measures:**
   * Implement code validation with timing attack protection
   * Rate limit verification attempts (3 tries per code)
   * Block suspicious phone numbers automatically
   * Log security events for monitoring

Include database schema, API endpoints, and security considerations.

### Prompt 6: AWS SNS Error Handling

Implement comprehensive error handling for AWS SNS specific scenarios:

1. **SNS Error Types:**
   * InvalidParameter (malformed phone numbers)
   * Throttling (rate limit exceeded)
   * OptedOut (user has opted out of SMS)
   * InvalidParameterValue (unsupported destination)
   * EndpointDisabled (blocked number)
2. **Recovery Strategies:**
   * Automatic retry with exponential backoff
   * Fallback to alternative SMS provider
   * Queue messages for later retry
   * Graceful degradation to email verification
3. **User Experience:**
   * Show appropriate error messages for each failure type
   * Provide alternative verification methods
   * Allow users to update phone numbers easily
   * Include support contact for persistent issues
4. **Monitoring and Alerting:**
   * Set up CloudWatch alarms for high error rates
   * Track delivery success rates by country
   * Monitor spending and usage patterns
   * Alert on unusual failure patterns

Include specific error handling code and user-facing error messages.

## Phase 3: Integration and Optimization

### Prompt 7: Cost Monitoring and Optimization

Implement AWS SNS cost monitoring and optimization:

1. **Real-time Cost Tracking:**
   * Track cost per message by destination country
   * Monitor daily/monthly spending against budgets
   * Calculate cost per successful verification
   * Compare costs between different routing strategies
2. **Optimization Strategies:**
   * Route domestic messages through cheapest paths
   * Use Sender ID where possible (free in many countries)
   * Batch non-urgent messages during off-peak hours
   * Implement intelligent retry strategies to reduce waste
3. **Budget Controls:**
   * Set up AWS billing alerts at multiple thresholds
   * Implement application-level spending limits
   * Automatically switch to backup provider if budget exceeded
   * Generate cost reports for stakeholders
4. **Analytics Dashboard:**
   * Show real-time cost metrics
   * Display success/failure rates by country
   * Track user conversion rates through verification flow
   * Compare costs between SMS providers

Include CloudWatch dashboards, cost allocation tags, and reporting queries.

### Prompt 8: International Compliance and Routing

Handle international SMS compliance and optimal routing:

1. **Country-Specific Requirements:**
   * India: Handle ILDO vs local routing ($0.02171 vs $0.00278)
   * US: Use 10DLC for better deliverability
   * EU: Implement GDPR-compliant opt-in/opt-out
   * China: Use approved SMS gateways only
2. **Intelligent Routing:**
   * Detect user country from phone number
   * Select cheapest compliant routing option
   * Handle timezone considerations for delivery
   * Implement local business hours respect
3. **Compliance Features:**
   * Automatic opt-out handling
   * Maintain suppression lists per country
   * Include required legal disclaimers
   * Track consent and communication preferences
4. **Delivery Optimization:**
   * Choose optimal sending times by timezone
   * Implement retry strategies based on country
   * Handle carrier-specific delivery issues
   * Monitor and improve delivery rates

Include country configuration files, routing logic, and compliance checklists.

### Prompt 9: Monitoring and Performance Optimization

Set up comprehensive monitoring for AWS SNS phone verification:

1. **CloudWatch Integration:**
   * Create custom metrics for verification success rates
   * Track SMS delivery times and failure reasons
   * Monitor API latency and error rates
   * Set up automated alerts for issues
2. **Application Performance Monitoring:**
   * Track user journey through verification flow
   * Monitor frontend/backend response times
   * Measure verification completion rates
   * Identify and resolve bottlenecks
3. **Business Metrics:**
   * Track verification abandonment rates
   * Monitor support ticket volume related to SMS
   * Measure cost per successful user verification
   * Track user satisfaction with verification process
4. **Alerting Strategy:**
   * Critical: High failure rates, service outages
   * Warning: Increased costs, slower response times
   * Info: Daily usage reports, cost summaries
   * Custom: Country-specific delivery issues

Include CloudWatch dashboard JSON, alert configurations, and performance baselines.

## Phase 4: Testing and Deployment

### Prompt 10: Testing Strategy for AWS SNS

Create comprehensive testing for AWS SNS integration:

1. **Unit Tests:**
   * Mock AWS SNS SDK for isolated testing
   * Test phone number validation and formatting
   * Verify error handling for all SNS error types
   * Test cost calculation and tracking logic
2. **Integration Tests:**
   * Test actual SMS delivery to test numbers
   * Verify delivery status tracking works correctly
   * Test failover to backup SMS provider
   * Validate international routing logic
3. **Load Testing:**
   * Test concurrent SMS sending under load
   * Verify rate limiting and throttling handling
   * Test system behavior during SNS outages
   * Validate cost controls under high volume
4. **User Acceptance Testing:**
   * Test complete verification flow end-to-end
   * Verify user experience across different devices
   * Test edge cases (invalid numbers, blocked users)
   * Validate international user experiences

Include test data sets, mock services, and testing environments setup.

### Prompt 11: Production Deployment and Rollback

Plan production deployment with safe rollback strategy:

1. **Deployment Strategy:**
   * Blue-green deployment for zero downtime
   * Feature flags for gradual SMS provider transition
   * A/B testing between email and phone verification
   * Gradual traffic shifting (10% → 50% → 100%)
2. **Environment Configuration:**
   * Separate AWS accounts for dev/staging/prod
   * Environment-specific SNS configurations
   * Secure credential management
   * Infrastructure as code deployment
3. **Rollback Planning:**
   * Automatic rollback triggers (high error rates)
   * Manual rollback procedures
   * Data migration considerations
   * User communication during issues
4. **Post-Deployment Monitoring:**
   * Enhanced monitoring for first 48 hours
   * Daily cost and usage reports
   * User feedback collection
   * Performance comparison with previous system

Include deployment scripts, rollback procedures, and monitoring checklists.

## Implementation Checklist

### Pre-Implementation:

- [ ] AWS account setup with proper permissions

- [ ] 10DLC registration initiated (US users)

- [ ] Origination identities configured

- [ ] Development environment configured

### Core Implementation:

- [ ] AWS SNS SDK integration complete

- [ ] Phone verification API endpoints created

- [ ] Frontend verification components built

- [ ] Error handling and fallbacks implemented

### Pre-Production:

- [ ] Cost monitoring and alerts configured

- [ ] International routing tested

- [ ] Load testing completed

- [ ] Security audit performed

### Production Launch:

- [ ] Gradual rollout plan executed

- [ ] Monitoring dashboards active

- [ ] Support team trained on new system

- [ ] User documentation updated

## Key Success Metrics:

- \*\*Cost Reduction\*\*: 70-95% savings vs current solution

- \*\*Delivery Rate\*\*: >95% successful SMS delivery

- \*\*User Experience\*\*: <2 minute verification completion

- \*\*System Reliability\*\*: 99.9% uptime

- \*\*Support Reduction\*\*: <2% users requiring assistance