# **System Design Document: Nationwide Toll Hub**

## **1. Executive Summary**

### **1.1 System Overview**

The Nationwide Toll Hub is a unified toll management platform that aggregates toll accounts from multiple state and regional agencies, providing users with consolidated billing, dispute management, and comprehensive toll tracking across the United States. The system serves as an intermediary layer between users and disparate toll agencies, normalizing data formats, reconciling transactions, and providing a seamless multi-platform experience.

### **1.2 Key Design Principles**

* **Eventual Consistency**: Tolling events may arrive with delays; the system embraces asynchronous processing
* **Fault Isolation**: Agency connector failures must not impact other integrations
* **Data Sovereignty**: Respect agency data ownership while maintaining unified user experience
* **Security First**: PCI compliance for payments, encrypted credential storage, comprehensive audit logging
* **Scalability**: Handle 100k+ concurrent users, 1M+ daily transactions, 50+ agency integrations

## **2. System Architecture**

### **2.1 High-Level Architecture**

┌─────────────────────────────────────────────────────────────┐

│ Client Layer │

├──────────────┬──────────────┬──────────────┬────────────────┤

│ iOS App │ Android App │ Web App │ Partner API │

└──────────────┴──────────────┴──────────────┴────────────────┘

│

▼

┌─────────────────────────────────────────────────────────────┐

│ API Gateway (Kong) │

│ Rate Limiting | Auth | Load Balancing │

└─────────────────────────────────────────────────────────────┘

│

┌──────────────────────┼──────────────────────┐

▼ ▼ ▼

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ Core API │ │ Notification │ │ Webhook │

│ Services │ │ Service │ │ Service │

└──────────────┘ └──────────────┘ └──────────────┘

│ │ │

└──────────────────────┴──────────────────────┘

│

┌──────────┴──────────┐

▼ ▼

┌──────────────────┐ ┌──────────────────┐

│ Event Backbone │ │ Cache Layer │

│ (Kafka/Pub/Sub) │ │ (Redis) │

└──────────────────┘ └──────────────────┘

│

┌──────────────┼──────────────┬────────────────┐

▼ ▼ ▼ ▼

┌────────┐ ┌────────┐ ┌────────┐ ┌────────┐

│Connector│ │Statement│ │Dispute │ │Analytics│

│Manager │ │Processor│ │Handler │ │Pipeline │

└────────┘ └────────┘ └────────┘ └────────┘

│ │ │ │

└──────────────┴──────────────┴────────────────┘

│

┌──────────────┼──────────────┐

▼ ▼ ▼

┌────────────┐ ┌────────────┐ ┌────────────┐

│ PostgreSQL │ │ S3/GCS │ │ Snowflake │

│ (OLTP) │ │ (Object) │ │ (OLAP) │

└────────────┘ └────────────┘ └────────────┘

### **2.2 Component Architecture**

#### **2.2.1 Client Applications**

* **Mobile Apps**: React Native with shared TypeScript SDK
* **Web Application**: React SPA with responsive design
* **Partner API**: REST/GraphQL endpoints for B2B integrations

#### **2.2.2 API Gateway Layer**

* **Kong Gateway**: Centralized entry point with plugins for:
  + Rate limiting (per-user and global)
  + JWT validation and API key management
  + Request/response transformation
  + Circuit breaking for downstream services

#### **2.2.3 Core Services**

**Auth Service**

* JWT token issuance and validation
* MFA orchestration (TOTP/SMS)
* Session management with Redis
* Password policy enforcement and breach checking

**User Service**

* Profile management
* Vehicle CRUD operations
* Payment method tokenization interface
* Consent and privacy preference management

**Agency Service**

* Agency metadata and capability management
* Link state machine orchestration
* Credential vault interface
* OAuth2 flow handling

**Toll Service**

* Event ingestion and normalization
* Deduplication logic
* Rate calculation and class mapping
* Evidence artifact management

**Statement Service**

* Daily statement generation
* Reconciliation engine
* Export generation (CSV, PDF)
* Payment orchestration

**Dispute Service**

* Case management workflow
* Evidence collection
* Agency communication interface
* SLA tracking

### **2.3 Integration Layer Architecture**

┌─────────────────────────────────────────────────────────┐

│ Connector Manager │

├─────────────────────────────────────────────────────────┤

│ Connector Registry | Scheduler | Health Monitor │

└────────────┬────────────────────────────────────────────┘

│

┌────────────┼─────────────────────────────────────────────┐

│ ▼ │

│ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ │

│ │ E-ZPass │ │ SunPass │ │ FasTrak │ │

│ │ Connector │ │ Connector │ │ Connector │ │

│ └──────────────┘ └──────────────┘ └──────────────┘ │

│ │

│ ┌──────────────┐ ┌──────────────┐ ┌──────────────┐ │

│ │ TxTag │ │ MTA │ │ Generic │ │

│ │ Connector │ │ Connector │ │ Connector │ │

│ └──────────────┘ └──────────────┘ └──────────────┘ │

└──────────────────────────────────────────────────────────┘

Each connector implements:

* **Adapter Pattern**: Standardized interface regardless of agency protocol
* **Retry Logic**: Exponential backoff with jitter
* **Circuit Breaker**: Fail fast when agencies are down
* **Data Mapping**: Agency-specific to normalized format transformation
* **Rate Limiting**: Respect agency-specific limits

## **3. Data Architecture**

### **3.1 Database Design**

#### **3.1.1 Primary Database (PostgreSQL)**

**Core Tables**

-- Users and Authentication

users (

id UUID PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

email\_verified BOOLEAN DEFAULT FALSE,

password\_hash VARCHAR(255),

mfa\_enabled BOOLEAN DEFAULT FALSE,

mfa\_secret VARCHAR(255),

created\_at TIMESTAMP,

updated\_at TIMESTAMP,

last\_login\_at TIMESTAMP,

status VARCHAR(50) -- active, suspended, deleted

)

-- Vehicles

vehicles (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

plate VARCHAR(20) NOT NULL,

plate\_state VARCHAR(2) NOT NULL,

vehicle\_type VARCHAR(50),

axle\_count INTEGER,

class VARCHAR(50),

nickname VARCHAR(100),

active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP,

updated\_at TIMESTAMP,

UNIQUE(user\_id, plate, plate\_state)

)

-- Payment Methods (tokenized)

payment\_methods (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

type VARCHAR(20), -- card, ach

processor VARCHAR(50), -- stripe, plaid

processor\_token VARCHAR(255), -- encrypted

last4 VARCHAR(4),

brand VARCHAR(50),

is\_default BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP,

updated\_at TIMESTAMP

)

-- Agency Registry

agencies (

id VARCHAR(50) PRIMARY KEY,

name VARCHAR(255),

region VARCHAR(100),

states TEXT[], -- array of state codes

protocol VARCHAR(50), -- ezpass, 6c, proprietary

capabilities JSONB, -- {read: true, write: true, topup: true, evidence: true}

connector\_config JSONB, -- encrypted connection details

status VARCHAR(50),

created\_at TIMESTAMP,

updated\_at TIMESTAMP

)

-- Agency Account Links

agency\_account\_links (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

agency\_id VARCHAR(50) REFERENCES agencies(id),

external\_account\_id VARCHAR(255), -- agency's account ID

status VARCHAR(50), -- pending, active, failed, revoked

auth\_method VARCHAR(50), -- oauth, credentials

auth\_tokens JSONB, -- encrypted tokens/credentials

last\_sync\_at TIMESTAMP,

next\_sync\_at TIMESTAMP,

sync\_status VARCHAR(50),

created\_at TIMESTAMP,

updated\_at TIMESTAMP,

UNIQUE(user\_id, agency\_id)

)

-- Toll Events (partitioned by month)

toll\_events (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

vehicle\_id UUID REFERENCES vehicles(id),

agency\_id VARCHAR(50) REFERENCES agencies(id),

external\_event\_id VARCHAR(255), -- agency's event ID

plate VARCHAR(20),

plate\_state VARCHAR(2),

event\_timestamp TIMESTAMP NOT NULL,

gantry\_id VARCHAR(255),

location JSONB, -- {lat, lon, direction, road\_name}

vehicle\_class VARCHAR(50),

raw\_amount DECIMAL(10,2),

rated\_amount DECIMAL(10,2),

fees DECIMAL(10,2),

currency VARCHAR(3) DEFAULT 'USD',

evidence\_uri TEXT,

source VARCHAR(50), -- agency\_feed, plate\_pay, manual

status VARCHAR(50), -- pending, posted, disputed, voided

created\_at TIMESTAMP,

updated\_at TIMESTAMP,

UNIQUE(agency\_id, external\_event\_id)

) PARTITION BY RANGE (event\_timestamp);

-- Statements

statements (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

period\_start TIMESTAMP,

period\_end TIMESTAMP,

timezone VARCHAR(50),

subtotal DECIMAL(10,2),

fees DECIMAL(10,2),

credits DECIMAL(10,2),

total DECIMAL(10,2),

status VARCHAR(50), -- draft, open, closed, paid, overdue

payment\_method\_id UUID REFERENCES payment\_methods(id),

payment\_transaction\_id VARCHAR(255),

paid\_at TIMESTAMP,

breakdown JSONB, -- detailed breakdown by vehicle/agency

created\_at TIMESTAMP,

updated\_at TIMESTAMP

)

-- Statement Line Items

statement\_items (

id UUID PRIMARY KEY,

statement\_id UUID REFERENCES statements(id),

toll\_event\_id UUID REFERENCES toll\_events(id),

amount DECIMAL(10,2),

created\_at TIMESTAMP

)

-- Disputes

disputes (

id UUID PRIMARY KEY,

user\_id UUID REFERENCES users(id),

toll\_event\_id UUID REFERENCES toll\_events(id),

type VARCHAR(50), -- wrong\_plate, wrong\_class, duplicate, other

status VARCHAR(50), -- submitted, in\_review, resolved, rejected

description TEXT,

evidence\_urls TEXT[],

agency\_reference VARCHAR(255),

resolution TEXT,

submitted\_at TIMESTAMP,

resolved\_at TIMESTAMP,

sla\_deadline TIMESTAMP

)

-- Audit Log

audit\_logs (

id UUID PRIMARY KEY,

user\_id UUID,

entity\_type VARCHAR(50),

entity\_id UUID,

action VARCHAR(50),

changes JSONB,

ip\_address INET,

user\_agent TEXT,

created\_at TIMESTAMP

)

#### **3.1.2 Cache Layer (Redis)**

**Key Patterns**

session:{user\_id} → JWT token and session data (TTL: 24h)

user:{user\_id} → User profile cache (TTL: 1h)

vehicle:{user\_id} → User's vehicles list (TTL: 1h)

agency:capabilities → Agency capability matrix (TTL: 24h)

rate\_limit:{user\_id}:{endpoint} → Rate limiting counters (TTL: 1m)

statement:draft:{user\_id}:{date} → In-progress statement (TTL: 48h)

connector:health:{agency\_id} → Connector status (TTL: 5m)

### **3.2 Event Stream Architecture**

**Kafka Topics**

toll.events.raw → Raw events from agencies

toll.events.normalized → Processed and normalized events

toll.events.matched → Events matched to users/vehicles

statements.generate → Statement generation triggers

statements.closed → Completed statements

disputes.submitted → New dispute cases

disputes.updates → Dispute status changes

payments.initiated → Payment attempts

payments.completed → Successful payments

connector.health → Connector status updates

### **3.3 Object Storage Structure**

**S3/GCS Buckets**

toll-evidence/

├── {year}/{month}/{day}/{agency\_id}/{event\_id}.jpg

├── metadata/{event\_id}.json

statements/

├── {year}/{month}/{user\_id}/{statement\_id}.pdf

├── exports/{year}/{month}/{user\_id}/{export\_id}.csv

disputes/

├── {dispute\_id}/

│ ├── user\_evidence/

│ └── agency\_response/

audit-logs/

├── {year}/{month}/{day}/audit\_{timestamp}.jsonl

## **4. Processing Flows**

### **4.1 Toll Event Processing Pipeline**

Agency Feed → Connector → Raw Events Topic → Event Processor

│

▼

┌──────────────────┐

│ Deduplication │

│ (Redis) │

└────────┬─────────┘

▼

┌──────────────────┐

│ Normalization │

│ & Validation │

└────────┬─────────┘

▼

┌──────────────────┐

│ Vehicle/User │

│ Matching │

└────────┬─────────┘

▼

┌──────────────────┐

│ Rate & Class │

│ Calculation │

└────────┬─────────┘

▼

┌──────────────────┐

│ Store Event │

│ (PostgreSQL) │

└────────┬─────────┘

▼

┌──────────────────┐

│ Update Statement │

│ Draft │

└──────────────────┘

### **4.2 Statement Generation Flow**

Daily Cron (Local Midnight) → Statement Generator

│

┌─────────────────┼─────────────────┐

▼ ▼ ▼

Query Events Check Pending Get User Prefs

by Date Events (Timezone)

│ │ │

└─────────────────┴─────────────────┘

│

▼

┌──────────────────┐

│ Calculate Totals │

│ by Vehicle │

└────────┬─────────┘

▼

┌──────────────────┐

│ Apply Discounts │

│ and Fees │

└────────┬─────────┘

▼

┌──────────────────┐

│Create Statement │

│ Record │

└────────┬─────────┘

▼

┌──────────────────┐

│Generate PDF/CSV │

│ Store in S3 │

└────────┬─────────┘

▼

┌──────────────────┐

│Send Notification │

│ (Email/Push) │

└──────────────────┘

### **4.3 Reconciliation Process**

Nightly Job → Reconciliation Engine

│

▼

For Each Agency:

│

├→ Pull Agency Invoice/Summary

├→ Query Our Records

├→ Compare Totals

│

└→ If Variance > Threshold:

├→ Generate Discrepancy Report

├→ Create Investigation Tasks

└→ Alert Operations Team

## **5. API Design**

### **5.1 API Standards**

* **REST**: Primary API style with JSON payloads
* **GraphQL**: Optional for complex queries (P2)
* **Versioning**: URL path versioning /v1/
* **Pagination**: Cursor-based for large datasets
* **Rate Limiting**: Token bucket algorithm
* **Authentication**: Bearer JWT tokens

### **5.2 Core API Endpoints**

Authentication:

POST /v1/auth/signup

POST /v1/auth/login

POST /v1/auth/refresh

POST /v1/auth/logout

POST /v1/auth/mfa/enable

POST /v1/auth/mfa/verify

POST /v1/auth/password/reset

Users:

GET /v1/users/me

PATCH /v1/users/me

DELETE /v1/users/me

GET /v1/users/me/preferences

PUT /v1/users/me/preferences

Vehicles:

GET /v1/vehicles

POST /v1/vehicles

GET /v1/vehicles/{id}

PATCH /v1/vehicles/{id}

DELETE /v1/vehicles/{id}

Payment Methods:

GET /v1/payment-methods

POST /v1/payment-methods

GET /v1/payment-methods/{id}

DELETE /v1/payment-methods/{id}

POST /v1/payment-methods/{id}/set-default

Agencies:

GET /v1/agencies

GET /v1/agencies/{id}

GET /v1/agencies/coverage

POST /v1/agencies/{id}/link

DELETE /v1/agencies/{id}/link

GET /v1/agencies/{id}/accounts

POST /v1/agencies/{id}/accounts/{accountId}/topup

Tolls:

GET /v1/tolls

GET /v1/tolls/{id}

GET /v1/tolls/timeline

Statements:

GET /v1/statements

GET /v1/statements/{id}

GET /v1/statements/{id}/items

GET /v1/statements/{id}/export

POST /v1/statements/{id}/pay

Disputes:

GET /v1/disputes

POST /v1/disputes

GET /v1/disputes/{id}

PATCH /v1/disputes/{id}

POST /v1/disputes/{id}/evidence

Coverage:

GET /v1/coverage/map

GET /v1/coverage/agencies

GET /v1/coverage/check

Webhooks (Partner API):

GET /v1/webhooks/subscriptions

POST /v1/webhooks/subscriptions

DELETE /v1/webhooks/subscriptions/{id}

POST /v1/webhooks/test

### **5.3 Error Response Format**

{

"error": {

"code": "VALIDATION\_ERROR",

"message": "Invalid request parameters",

"details": [

{

"field": "plate",

"message": "Invalid plate format for state TX"

}

],

"request\_id": "req\_abc123",

"timestamp": "2025-09-27T12:00:00Z"

}

}

## **6. Security Architecture**

### **6.1 Authentication & Authorization**

**Multi-Factor Authentication**

* TOTP (Time-based One-Time Password)
* SMS backup with rate limiting
* Backup codes (10 single-use codes)
* Risk-based authentication triggers

**JWT Token Structure**

{

"sub": "user\_uuid",

"email": "user@example.com",

"roles": ["user"],

"agencies": ["md\_driveez", "pa\_turnpike"],

"iat": 1234567890,

"exp": 1234571490,

"jti": "token\_uuid"

}

### **6.2 Encryption Strategy**

**Data at Rest**

* PostgreSQL: Transparent Data Encryption (TDE)
* Field-level encryption for:
  + Payment tokens
  + Agency credentials
  + SSN/Tax IDs (if collected)
* KMS: AWS KMS or Google Cloud KMS
* Key rotation: Quarterly for master keys

**Data in Transit**

* TLS 1.3 minimum for all connections
* Certificate pinning for mobile apps
* mTLS for agency connections where supported

### **6.3 Credential Vault Design**

┌──────────────┐ ┌──────────────┐ ┌──────────────┐

│ User App │────▶│ Vault Service│────▶│ HSM/KMS │

└──────────────┘ └──────────────┘ └──────────────┘

│

▼

┌──────────────────┐

│ Encrypted Store │

│ (PostgreSQL) │

└──────────────────┘

**Vault Operations**

* Store: Encrypt with user-specific DEK
* Retrieve: Require MFA for first access
* Rotate: Automated key rotation
* Audit: All access logged

### **6.4 PCI Compliance**

**Scope Reduction**

* Use payment processor tokens only
* No raw card data storage
* Network segmentation for payment services
* Regular security scans and penetration testing

**Required Controls**

* WAF for public endpoints
* DLP for preventing card data leakage
* Access controls with principle of least privilege
* Comprehensive audit logging

## **7. Scalability & Performance**

### **7.1 Scaling Strategy**

**Horizontal Scaling Targets**

* API Services: Auto-scale 5-50 instances
* Event Processors: 10-100 workers
* Database: Read replicas for queries
* Cache: Redis Cluster mode

**Load Testing Targets**

Concurrent Users: 100,000

Daily Active Users: 500,000

Toll Events/Hour: 100,000

Peak Events/Second: 5,000

Statement Generation: 1M in 30 min

API Response Time (P99): <500ms

Event Processing Lag: <60 seconds

### **7.2 Database Optimization**

**Indexing Strategy**

-- Primary lookup patterns

CREATE INDEX idx\_toll\_events\_user\_date ON toll\_events(user\_id, event\_timestamp);

CREATE INDEX idx\_toll\_events\_vehicle ON toll\_events(vehicle\_id, event\_timestamp);

CREATE INDEX idx\_toll\_events\_agency ON toll\_events(agency\_id, external\_event\_id);

CREATE INDEX idx\_statements\_user\_period ON statements(user\_id, period\_end DESC);

-- Partitioning

CREATE TABLE toll\_events\_2025\_09 PARTITION OF toll\_events

FOR VALUES FROM ('2025-09-01') TO ('2025-10-01');

**Query Optimization**

* Materialized views for common aggregations
* Prepared statements for frequent queries
* Connection pooling (PgBouncer)
* Read/write splitting

### **7.3 Caching Strategy**

**Cache Layers**

1. **CDN**: Static assets, coverage maps
2. **Application Cache**: Redis for hot data
3. **Database Cache**: PostgreSQL buffer cache
4. **Client Cache**: Mobile app SQLite

**Cache Invalidation**

* TTL-based for reference data
* Event-based for user data
* Cache-aside pattern for statements

## **8. Monitoring & Observability**

### **8.1 Metrics Collection**

**Application Metrics (Prometheus)**

# Business Metrics

toll\_events\_processed\_total

statements\_generated\_total

disputes\_submitted\_total

payment\_success\_rate

agency\_sync\_success\_rate

# Performance Metrics

http\_request\_duration\_seconds

database\_query\_duration\_seconds

event\_processing\_lag\_seconds

cache\_hit\_ratio

**Infrastructure Metrics**

* CPU, Memory, Disk, Network
* Container orchestration metrics
* Database connection pools
* Message queue lag

### **8.2 Logging Architecture**

Application → Fluentd → Elasticsearch → Kibana

↓

S3 Archive

**Log Levels and Retention**

* ERROR: 90 days hot, 2 years cold
* WARN: 30 days hot, 1 year cold
* INFO: 14 days hot, 90 days cold
* DEBUG: 7 days hot (dev/staging only)

### **8.3 Distributed Tracing**

**OpenTelemetry Implementation**

* Trace ID propagation across services
* Span collection for critical paths
* Sampling rate: 1% baseline, 100% for errors

**Critical Traces**

* User authentication flow
* Toll event processing pipeline
* Statement generation
* Payment processing

### **8.4 Alerting Strategy**

**Alert Priorities**

P0 (Page):

- API gateway down

- Database connection failure

- Payment processor unreachable

- Data corruption detected

P1 (Urgent):

- Error rate > 1%

- Response time > 2s (P95)

- Agency sync failure > 3 attempts

- Statement generation delayed > 2 hours

P2 (Warning):

- Disk usage > 80%

- Cache miss rate > 50%

- Connector degraded performance

- Reconciliation variance > 0.5%

## **9. Disaster Recovery & Business Continuity**

### **9.1 Backup Strategy**

**Database Backups**

* Full backup: Daily at 2 AM ET
* Incremental: Every 4 hours
* Point-in-time recovery: 30-day window
* Cross-region replication: Real-time

**Object Storage**

* Versioning enabled
* Cross-region replication
* Lifecycle policies for archival

### **9.2 High Availability Design**

Region 1 (Primary) Region 2 (DR)

┌──────────────┐ ┌──────────────┐

│ Load Balancer│ │Load Balancer │

└──────┬───────┘ └──────┬───────┘

│ │

┌───▼───┐ ┌───▼───┐

│ Apps │◄───Replication────▶│ Apps │

└───┬───┘ └───┬───┘

│ │

┌───▼───┐ ┌───▼───┐

│ DB │◄───Streaming──────▶│ DB │

└───────┘ Replication └───────┘

**RTO/RPO Targets**

* RPO (Recovery Point Objective): 15 minutes
* RTO (Recovery Time Objective): 2 hours
* Availability Target: 99.95% (4.38 hours/year)

### **9.3 Incident Response**

**Runbook Examples**

1. Database failover procedure
2. Agency connector circuit breaker reset
3. Payment processor fallback
4. Statement generation recovery
5. Cache cluster rebuild

## **10. Deployment Architecture**

### **10.1 Container Orchestration**

**Kubernetes Configuration**

Namespaces:

- toll-hub-prod

- toll-hub-staging

- toll-hub-dev

Deployments:

- api-gateway (3-10 replicas)

- auth-service (3-5 replicas)

- toll-service (5-20 replicas)

- statement-service (5-10 replicas)

- connector-manager (2-5 replicas)

Services:

- ClusterIP for internal

- LoadBalancer for external

- NodePort for health checks

### **10.2 CI/CD Pipeline**

Code Push → GitHub Actions → Build & Test → Container Registry

│

▼

Staging Deployment

│

▼

Integration Tests

│

▼

Manual Approval

│

▼

Production Deploy

(Blue/Green)

**Deployment Stages**

1. **Build**: Compile, unit tests, security scan
2. **Test**: Integration tests, load tests
3. **Stage**: Deploy to staging, smoke tests
4. **Approve**: Manual review for production
5. **Deploy**: Blue/green with automated rollback

### **10.3 Infrastructure as Code**

**Terraform Modules**

modules/

├── networking/

│ ├── vpc.tf

│ ├── subnets.tf

│ └── security\_groups.tf

├── compute/

│ ├── eks\_cluster.tf

│ └── node\_groups.tf

├── data/

│ ├── rds.tf

│ ├── elasticache.tf

│ └── s3.tf

└── monitoring/

├── cloudwatch.tf

└── datadog.tf

## **11. Technology Stack Summary**

### **11.1 Core Technologies**

| **Component** | **Technology** | **Justification** |
| --- | --- | --- |
| Mobile Apps | React Native | Code reuse, TypeScript support |
| Web App | React + TypeScript | Consistent with mobile |
| Backend Services | Node.js (NestJS) | TypeScript, good ecosystem |
| API Gateway | Kong | Plugin ecosystem, performance |
| Primary Database | PostgreSQL 15 | JSONB, partitioning, reliability |
| Cache | Redis Cluster | Performance, data structures |
| Message Queue | Apache Kafka | Throughput, durability |
| Object Storage | AWS S3 / GCS | Scalability, lifecycle management |
| Analytics | Snowflake | Separation of OLAP/OLTP |
| Container Orchestration | Kubernetes (EKS/GKE) | Industry standard |
| Monitoring | Datadog + Prometheus | Comprehensive observability |
| CI/CD | GitHub Actions | Integration with repository |

### **11.2 Third-Party Services**

| **Service** | **Provider** | **Purpose** |
| --- | --- | --- |
| Payment Processing | Stripe | PCI compliance, tokenization |
| SMS | Twilio | MFA, notifications |
| Email | SendGrid | Transactional emails |
| CDN | CloudFlare | Static assets, DDoS protection |
| Error Tracking | Sentry | Real-time error monitoring |
| Feature Flags | LaunchDarkly | Progressive rollouts |

## **12. Development Guidelines**

### **12.1 Code Organization**

toll-hub/

├── apps/

│ ├── mobile/

│ ├── web/

│ └── api/

├── packages/

│ ├── shared/ # Shared TypeScript models

│ ├── connectors/ # Agency connectors

│ └── ui-components/ # React components

├── services/

│ ├── auth/

│ ├── toll/

│ ├── statement/

│ └── dispute/

└── infrastructure/

├── terraform/

├── k8s/

└── scripts/

### **12.2 Testing Strategy**

**Test Coverage Targets**

* Unit Tests: 80% coverage
* Integration Tests: Critical paths
* E2E Tests: User journeys
* Load Tests: Weekly in staging
* Security Tests: Monthly penetration testing

### **12.3 Documentation Requirements**

* API documentation (OpenAPI/Swagger)
* Architecture Decision Records (ADRs)
* Runbooks for common operations
* Connector integration guides
* Security policies and procedures

## **13. Migration & Rollout Strategy**

### **13.1 Phase A: Pilot (0-90 days)**

**Milestone 1 (Days 1-30)**

* Core infrastructure setup
* Basic auth and user services
* 2-3 agency connector prototypes

**Milestone 2 (Days 31-60)**

* Statement generation MVP
* Payment integration
* Mobile app beta

**Milestone 3 (Days 61-90)**

* 6 agency integrations
* Load testing and optimization
* Limited beta launch (1000 users)

### **13.2 Phase B: Expansion (90-180 days)**

* Add dispute management
* Tag advisor feature
* 15+ agency integrations
* Scale to 50,000 users
* Partner API beta

### **13.3 Phase C: Full Launch (180-365 days)**

* National coverage (30+ agencies)
* Fleet management features
* Advanced analytics
* 500,000+ active users
* B2B partnerships active

## **14. Risk Mitigation**

### **14.1 Technical Risks**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Agency API instability | High | Circuit breakers, fallback to batch processing |
| Data reconciliation errors | High | Multiple checkpoints, manual review process |
| Scalability bottlenecks | Medium | Load testing, auto-scaling, database sharding |
| Security breach | Critical | Defense in depth, regular audits, insurance |
| Payment failures | High | Multiple processors, retry logic, manual fallback |

### **14.2 Operational Risks**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Agency resistance | High | Gradual rollout, clear value proposition |
| Regulatory compliance | Critical | Legal review, state-by-state compliance |
| User adoption | High | Marketing campaign, referral program |
| Support volume | Medium | Self-service options, AI chatbot, tiered support |

## **15. Success Metrics & KPIs**

### **15.1 Technical KPIs**

* **Availability**: >99.95% uptime
* **Performance**: <500ms API response time (P99)
* **Processing Lag**: <60 seconds for toll events
* **Error Rate**: <0.1% for critical paths
* **Security**: Zero critical vulnerabilities

### **15.2 Business KPIs**

* **User Acquisition**: 500K users in Year 1
* **Retention**: >80% monthly active users
* **Coverage**: 30+ integrated agencies
* **Revenue**: $10M ARR by end of Year 1
* **NPS**: >60 Net Promoter Score

## **16. Conclusion**

This system design provides a robust, scalable foundation for the Nationwide Toll Hub platform. The architecture emphasizes modularity, fault tolerance, and security while maintaining the flexibility to integrate with diverse agency systems. The phased rollout approach allows for iterative improvements and risk mitigation while building toward comprehensive national coverage.

Key success factors include:

* Strong agency partnerships and integration quality
* Exceptional user experience across all platforms
* Reliable and accurate toll processing and reconciliation
* Robust security and compliance framework
* Scalable architecture supporting rapid growth

The design prioritizes operational excellence and user trust, positioning the Nationwide Toll Hub as the definitive solution for unified toll management across the United States.