altpay Technical Overview

Internal Technical Documentation

Document Information

Field	Details
Project Name	altpay Payment Gateway
Document Type	Technical Overview
Version	1.0
Date	January 30, 2026
Author	Balogun Abdulsamad (Engineering Team Lead)
Audience	Engineering Team, Technical Staff, Project Managers

Table of Contents

1. Project Overview

altpay is a payment gateway platform that enables Nigerian businesses to accept card payments online. The system processes card transactions through Interswitch, handles 3D Secure authentication, and settles funds to merchant bank accounts.

Core Technical Functions

- Payment initialization and processing
- Card tokenization and secure storage
- 3D Secure (3DS) authentication flow
- Webhook delivery system
- Settlement batch processing

- Merchant dashboard and API access

---

2. Technology Stack

Frontend

Technology	Version	Purpose
Next.js	15.x	React framework for dashboard and checkout
TypeScript	5.7	Type-safe JavaScript
Tailwind CSS	4.x	Utility-first CSS framework
Shadcn/ui	Latest	Component library
Zustand	5.x	State management
TanStack Query	5.x	Server state management

Backend

Technology	Version	Purpose
Laravel	12.x	PHP framework for API
PHP	8.4	Server-side language
Laravel Octane	2.x	High-performance server (Swoole)
Laravel Sanctum	4.x	API authentication
Laravel Horizon	5.x	Queue monitoring

Database & Storage

Technology	Version	Purpose
PostgreSQL	16.x	Primary relational database
Redis	7.x	Cache, sessions, queue driver
Meilisearch	1.x	Full-text search
AWS S3	-	File storage

External Integrations

Service	Purpose
Interswitch	Primary card processor
Flutterwave	Backup card processor

Service	Purpose
SendGrid / AWS SES	Transactional emails
Termii	SMS notifications
Smile Identity	KYC verification

Observability

Tool	Purpose
Sentry	Error tracking and monitoring
Grafana + Prometheus	Metrics and dashboards
PostHog	Product analytics
Segment	Data pipeline
ELK Stack	Centralized logging
AWS CloudWatch	Infrastructure logs

3. System Architecture

Architecture Pattern

The system follows a **Modular Monolith** architecture - organized into distinct modules but deployed as a single application. This approach suits the team size while maintaining clear boundaries for potential future extraction.

Core Modules

Module	Responsibility
Payment	Transaction processing, card tokenization, 3DS handling
Merchant	Merchant management, onboarding, API key generation
Settlement	Daily payouts, reconciliation, batch processing
Notification	Email, SMS, webhook delivery
Analytics	Reporting, metrics, audit logging

Data Flow

1. **Payment Initiation**: Merchant server calls altpay API to create payment intent
2. **Checkout**: Customer redirected to hosted checkout or uses embedded form
3. **Card Capture**: altpay.js SDK captures card in secure iframe, sends to tokenization service
4. **Processing**: Backend sends token to Interswitch for authorization
5. **3D Secure**: If required, customer completes bank verification

6. **Confirmation**: Transaction result stored, webhook dispatched to merchant
7. **Settlement**: Daily batch job calculates merchant payouts and initiates bank transfers

API Design

- RESTful API with JSON responses
- Versioned endpoints (`/api/v1/`)
- Bearer token authentication (API keys)
- Idempotency key support for safe retries
- Rate limiting per merchant

---

4. Infrastructure

Cloud Provider

Amazon Web Services (AWS) - Lagos region (af-south-1) for low latency

Phase 1 Infrastructure (MVP)

Component	Service	Configuration
Compute	Laravel Forge / EC2	2x t3.large instances
Database	RDS PostgreSQL	db.r6g.large, Multi-AZ
Cache	ElastiCache Redis	cache.r6g.large cluster
Storage	S3	Standard storage class
CDN	CloudFront	Global edge locations
DNS	Route 53	Hosted zone
Load Balancer	ALB	Application load balancer

Phase 2 Infrastructure (Scale)

Component	Service	Configuration
Container Orchestration	EKS	Kubernetes cluster
Auto Scaling	EKS Node Groups	Min 2, Max 10 nodes
Database	RDS PostgreSQL	db.r6g.xlarge, read replicas

Environment Configuration

Environment	Purpose	Database
Development	Feature development	Shared dev DB
QA	Testing	Isolated test DB

Environment	Purpose	Database
Staging	Pre-production	Production mirror
Production	Live system	HA cluster

5. Security

Compliance

PCI DSS Level 1 compliance is required. Key requirements:

- Network segmentation and firewall rules
- Card data tokenization (never store raw PAN)
- TLS 1.3 for all data transmission
- AES-256-GCM encryption at rest
- Quarterly penetration testing
- Comprehensive audit logging

Encryption

Data Type	Method
Card Numbers	Tokenized (never stored)
CVV	Never stored
API Keys	Hashed with Argon2id
Passwords	Hashed with Argon2id
PII	AES-256-GCM encrypted
Database	AWS RDS encryption

Key Management

AWS KMS manages encryption keys with automatic 90-day rotation. Separate Data Encryption Keys (DEKs) for card tokens, PII, API secrets, and webhook secrets.

Authentication

- Merchant API: Bearer tokens (API keys)
- Dashboard: JWT tokens with refresh flow
- Admin: JWT + MFA required

Rate Limiting

Endpoint Type	Limit
Payment Initialize	100 req/min per merchant

Endpoint Type	Limit
Payment Verify	200 req/min per merchant
General API	1000 req/min per merchant
Checkout Page	60 req/min per IP

6. Team Structure

Team Composition (7 Members)

Role	Count	Responsibilities
Tech Lead	1	Architecture, code review, DevOps, security oversight
UI Designer	1	User experience, interface mockups, design system, prototyping
Backend Engineers	2	Laravel API, payment processing, integrations, settlements
Frontend Engineers	2	Next.js dashboard, checkout UI, responsive design
QA Engineer	1	Test planning, automation, quality assurance

Role Details

Tech Lead (Balogun Abdulsamad)

- 6+ years experience in payment systems and architecture
- Full-stack expertise (Laravel + Next.js)
- Infrastructure and DevOps management
- Security and PCI compliance oversight
- Code reviews and architectural decisions

UI Designer

- User experience (UX) research and design
- Interface mockups and prototypes (Figma)
- Design system and component library
- Checkout flow and dashboard layouts
- Collaboration with frontend engineers

Backend Engineer 1

- Payment processing and card tokenization
- Interswitch integration
- 3D Secure implementation

Backend Engineer 2

- Settlement engine and batch processing
- Webhook system

- Notifications and audit logging

Frontend Engineer 1

- Merchant dashboard
- Analytics views
- Settings pages

Frontend Engineer 2

- Checkout page and payment UI
- Public website
- altpay.js SDK

QA Engineer

- Test planning and execution
- Automated testing (Playwright, PHPUnit, Pest)
- API testing
- UAT coordination

---

7. Project Timeline

Phase Overview (5 Months Total)

Phase	Duration	Dates	Focus
MVP	8 weeks	Feb 3 - Mar 28, 2026	Core payment functionality
Phase 2	6 weeks	Mar 31 - May 9, 2026	Enhanced features
Phase 3	6 weeks	May 12 - Jun 20, 2026	Scale and advanced features

MVP Sprint Breakdown

Sprint 1: Foundation (Feb 3-14)

- Project setup, repositories, CI/CD pipelines
- AWS infrastructure provisioning
- Database schema and migrations
- Laravel project with Sanctum authentication
- Next.js project setup with design system

Sprint 2: Payment Core (Feb 17-28)

- Payment initialization API
- Interswitch integration
- Card tokenization service
- API key generation
- Checkout page (card form)

- Merchant dashboard foundation

Sprint 3: 3DS & Webhooks (Mar 3-14)

- 3D Secure integration
- Payment verification API
- Webhook delivery system
- Email/SMS notifications
- Transaction history views

Sprint 4: Settlement & Launch (Mar 17-28)

- Settlement calculation engine
- Bank transfer integration
- Settlement reports
- Final testing and bug fixes
- Production deployment
- MVP launch

Key Milestones

Date	Milestone
Feb 3, 2026	Development starts
Feb 14, 2026	Alpha - Registration working
Feb 28, 2026	Beta - Payments in test mode
Mar 14, 2026	Feature complete
Mar 28, 2026	MVP Launch
May 9, 2026	Phase 2 complete
Jun 20, 2026	Full Launch

---

8. Development Workflow

Git Workflow

- **main**: Production-ready code
- **develop**: Integration branch
- **feature/***: Feature branches
- **hotfix/***: Emergency fixes

CI/CD Pipeline

1. Code push triggers build
2. Run linting and static analysis
3. Execute unit and integration tests

4. Security vulnerability scan
5. Build Docker image
6. Deploy to appropriate environment

Deployment Strategy

- Blue-green deployment for zero downtime
- Rollback capability within 5 minutes
- Database migrations: forward-only, backward compatible

Code Review Process

All code requires review before merge:

- Tech Lead reviews architectural changes
- Peer review for feature code
- QA sign-off for release branches

Communication

Activity	Frequency	Participants
Daily Standup	Daily (15 min)	All team
Sprint Planning	Bi-weekly	All team
Code Review	Ongoing	Tech Lead + engineers
Sprint Retrospective	Bi-weekly	All team
Design Review	Weekly	Dev Team + UI Designer

9. Technical Risks

Risk Assessment

Risk	Likelihood	Impact	Mitigation
Security vulnerability	Low	Critical	PCI compliance, regular pen testing, security reviews
Interswitch downtime	Medium	High	Flutterwave as backup processor, 15-min switchover
Database performance	Low	High	Read replicas, query optimization, caching
Key developer leaves	Medium	Medium	Documentation, cross-training, knowledge sharing
Scope creep	High	Medium	Strict MVP scope, phased approach
Integration delays	Medium	Medium	Early integration testing, sandbox environments

Contingency Plans

Scenario	Response	Recovery Time
Primary processor down	Automatic failover to backup	15 minutes
Database failure	Promote RDS standby	5 minutes
Data center outage	Failover to backup region	30 minutes
Security incident	Isolate, investigate, notify	4 hours

10. Next Steps

Immediate Actions (Week 1)

Action	Owner	Deadline
Set up AWS infrastructure	Tech Lead	Feb 3
Create Git repositories	Tech Lead	Feb 3
Configure CI/CD pipelines	Tech Lead	Feb 5
Initialize Laravel project	Backend 1	Feb 5
Initialize Next.js project	Frontend 1	Feb 5
Create initial UI mockups	UI Designer	Feb 7
Set up development environments	All devs	Feb 7

Technical Decisions Required

Decision	Options	Recommendation	Deadline
Hosting approach	Forge vs EKS	Laravel Forge (MVP)	Feb 1
Primary processor	Interswitch vs Flutterwave	Interswitch	Feb 1
Email provider	SendGrid vs AWS SES	SendGrid	Feb 3
SMS provider	Termii vs Africa's Talking	Termii	Feb 3

Summary

Item	Details
Project	Payment gateway for card transactions
Stack	Next.js 15 + Laravel 12 + PostgreSQL
Infrastructure	AWS (Lagos region)

Item	Details
Team	7 members (1 Lead, 1 UI, 2 BE, 2 FE, 1 QA)
Timeline	2 months MVP, 5 months full launch
Launch Date	MVP: March 28, 2026

Document Version 1.0 | January 30, 2026