Somekamoney Technical Project Brief

Project: Somekamoney Web & Mobile Platform
Client: Tony "Owamanyige" Terrence Mutujju
Prepared by: Kanzu Finance Limited
Date: October 2025
Target: 1 million users in Year 1

Executive Summary

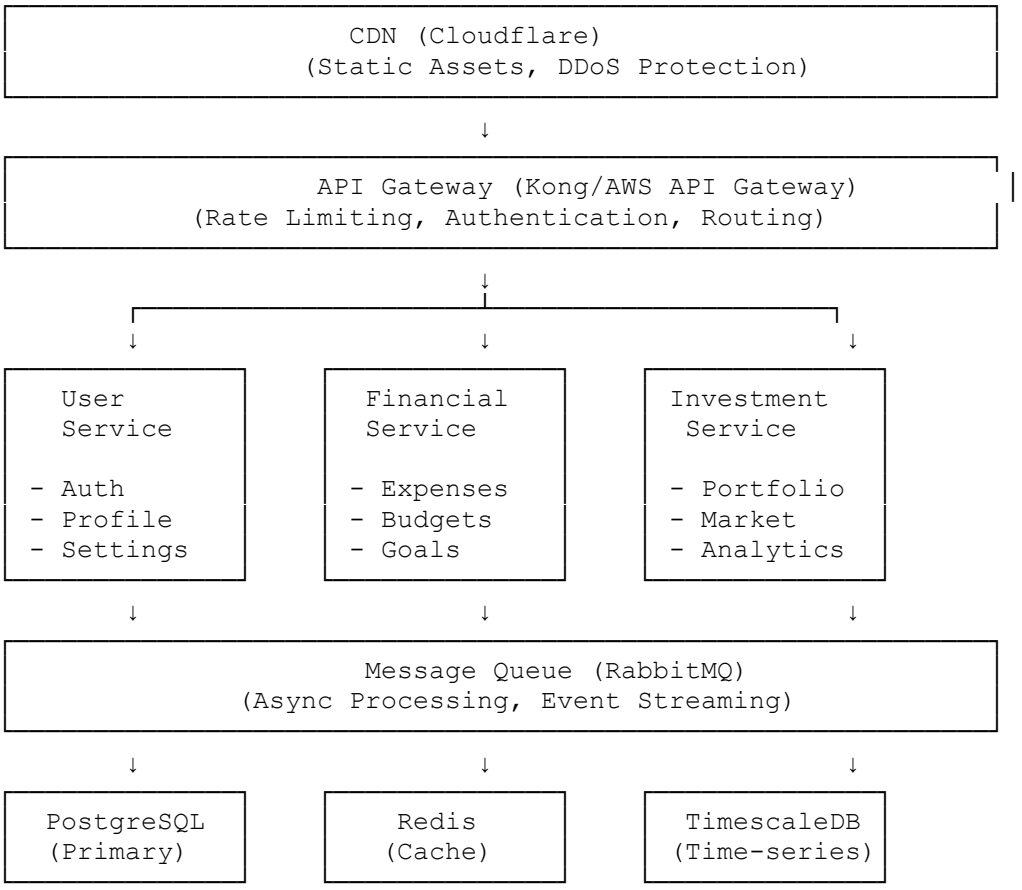
Somekamoney is a personal finance management platform designed to empower young Africans to track expenses, set financial goals, budget effectively, and invest wisely. This document outlines the technical architecture, technology choices, and scalability considerations for building a robust, secure, and high-performance platform capable of supporting rapid user growth.

---

1. System Architecture

High-Level Architecture

We propose a **microservices-based architecture** with the following key components:



Architecture Components

Frontend Layer:

- **Web Application:** Progressive Web App (PWA) for cross-platform compatibility
- **Mobile Applications:** Native iOS and Android apps sharing business logic

API Gateway:

- Single entry point for all client requests
- Handles authentication, rate limiting, request routing, and load balancing
- SSL termination and security policies enforcement

Microservices:

- **User Service:** Authentication, user profiles, preferences, notifications
- **Financial Service:** Expense tracking, budget management, financial goal setting
- **Investment Service:** Portfolio management, market data integration, investment recommendations
- **Analytics Service:** Financial insights, spending patterns, reporting
- **Notification Service:** Email, SMS, and push notifications

Data Layer:

- **Primary Database:** Transactional data and user information
- **Cache Layer:** Session management, frequently accessed data
- **Time-Series Database:** Financial metrics, expense trends, market data
- **Object Storage:** Document uploads, receipts, profile images

Infrastructure Layer:

- **Message Queue:** Asynchronous task processing and inter-service communication
- **Background Workers:** Scheduled jobs, data aggregation, report generation
- **Monitoring & Logging:** Application performance monitoring and centralized logging

---

2. Language & Framework Choices

Backend: Node.js with TypeScript + NestJS

Rationale:

1. **Performance:** Non-blocking I/O ideal for handling concurrent connections (critical for 1M users)
2. **Developer Ecosystem:** Largest package ecosystem (npm) with mature financial libraries
3. **Type Safety:** TypeScript reduces runtime errors and improves code maintainability
4. **Microservices Support:** NestJS provides excellent microservices architecture patterns out of the box
5. **Talent Pool:** Large developer community in Africa, easier to scale the engineering team
6. **Real-time Capabilities:** Native WebSocket support for live financial data updates
7. **Cost-Effective:** Efficient resource utilization reduces hosting costs at scale

Alternative Consideration: Python with Django/FastAPI (excellent for ML-based financial insights, but slightly lower concurrency performance)

Frontend Web: React with Next.js

Rationale:

1. **Performance:** Server-side rendering (SSR) and static site generation (SSG) for optimal load times
2. **SEO:** Critical for organic user acquisition through content marketing
3. **Developer Experience:** Large talent pool, excellent tooling, strong community support
4. **Progressive Web App:** Can function as mobile app in markets with limited smartphone access
5. **Code Reusability:** Shared components across web and mobile (with React Native)

Mobile: React Native with Expo

Rationale:

1. **Cross-Platform:** Single codebase for iOS and Android reduces development time by 40-50%
2. **Native Performance:** Near-native performance for financial calculations and UI
3. **Rapid Iteration:** Over-the-air updates for bug fixes without app store delays
4. **Shared Logic:** Reuse business logic and API integration code from web
5. **Offline-First:** Built-in support for offline data storage and sync
6. **Cost-Effective:** Significantly reduces development and maintenance costs

---

3. Database Choices

Primary Database: PostgreSQL 16

Rationale:

1. **ACID Compliance:** Critical for financial transactions and data integrity
2. **JSON Support:** Flexible schema for varying expense categories and user preferences
3. **Advanced Features:** Window functions, CTEs, and full-text search for analytics
4. **Scalability:** Proven performance with millions of concurrent users (Instagram, Spotify)
5. **Row-Level Security:** Built-in security features for multi-tenant data isolation
6. **Partitioning:** Table partitioning for efficient querying of historical financial data
7. **Open Source:** No licensing costs, strong community support

Schema Design Highlights:

- Partitioned tables for expenses (by month) to optimize query performance
- Materialized views for dashboard aggregations
- Read replicas for analytics and reporting queries

Cache Layer: Redis

Rationale:

1. **Performance:** Sub-millisecond latency for session data and frequently accessed information
2. **Session Management:** Distributed session storage for horizontal scaling
3. **Rate Limiting:** Built-in data structures for API rate limiting
4. **Real-Time Leaderboards:** Sorted sets for gamification features (financial challenges)
5. **Pub/Sub:** Real-time notifications and live updates

Time-Series Database: TimescaleDB (PostgreSQL Extension)

Rationale:

1. **Financial Data:** Optimized for time-series queries (expense trends, investment performance)
2. **Automatic Partitioning:** Efficient storage and querying of historical data
3. **Continuous Aggregates:** Pre-computed rollups for fast dashboard rendering
4. **PostgreSQL Compatibility:** Familiar query language, easy integration with primary database
5. **Compression:** Significant storage savings for historical financial data

Object Storage: AWS S3 / Cloudflare R2

Rationale:

1. **Scalability:** Unlimited storage for user receipts and documents
2. **Cost-Effective:** Pay only for what you use
3. **CDN Integration:** Fast global access to uploaded files
4. **Durability:** 99.999999999% durability guarantee

---

4. Hosting Choice

Recommended: AWS (Amazon Web Services) - Multi-Region Setup

Primary Region: eu-west-1 (Ireland) - Low latency to Africa
Secondary Region: af-south-1 (Cape Town) - Africa-specific deployment

Core Services:

- **EC2/ECS/EKS:** Application hosting with auto-scaling
- **RDS:** Managed PostgreSQL with automated backups and replication
- **ElastiCache:** Managed Redis for caching
- **S3:** Object storage for user uploads
- **CloudFront:** Global CDN for static assets and API acceleration
- **Route 53:** DNS with health checks and failover

- **SQS/SNS:** Message queuing and notifications
- **Lambda:** Serverless functions for background processing
- **WAF:** Web Application Firewall for security

Rationale:

1. **Africa Presence:** Cape Town region provides low latency for African users
2. **Proven Scale:** Supports platforms with hundreds of millions of users
3. **Managed Services:** Reduces operational overhead, faster time to market
4. **Security Compliance:** SOC 2, ISO 27001, PCI DSS certified
5. **Cost Management:** Reserved instances and spot instances for cost optimization
6. **Disaster Recovery:** Multi-region setup for business continuity
7. **Local Payment Integration:** Easy integration with African payment providers

Alternative: Google Cloud Platform (also has a Johannesburg region, excellent Kubernetes support)

Deployment Strategy:

- **Containerization:** Docker containers for consistent deployments
- **Orchestration:** Kubernetes (EKS) for container management and auto-scaling
- **CI/CD:** GitHub Actions → AWS CodePipeline → Automated deployments
- **Infrastructure as Code:** Terraform for reproducible infrastructure

---

5. Security Considerations

Authentication & Authorization

Implementation:

- **OAuth 2.0 + JWT:** Stateless authentication with refresh token rotation
- **Multi-Factor Authentication (MFA):** SMS and authenticator app support
- **Biometric Authentication:** Fingerprint/Face ID for mobile apps
- **Role-Based Access Control (RBAC):** Granular permissions for different user types
- **Session Management:** Redis-backed sessions with automatic expiry

Data Protection

Encryption:

- **At Rest:** AES-256 encryption for database and file storage
- **In Transit:** TLS 1.3 for all API communications
- **Field-Level Encryption:** Sensitive financial data (account numbers, investment details)
- **Key Management:** AWS KMS for encryption key rotation and management

Data Privacy:

- **GDPR Compliance:** Right to data portability and deletion
- **Data Residency:** Option to store data in African data centers
- **Anonymization:** PII anonymization for analytics and reporting
- **Audit Logging:** Comprehensive audit trail for all financial transactions

Application Security

Best Practices:

1. **Input Validation:** Server-side validation for all user inputs
2. **SQL Injection Prevention:** Parameterized queries and ORM usage
3. **XSS Protection:** Content Security Policy (CSP) and output encoding
4. **CSRF Protection:** Token-based CSRF prevention
5. **Rate Limiting:** API throttling to prevent abuse (100 requests/minute per user)
6. **DDoS Protection:** Cloudflare WAF and AWS Shield
7. **Dependency Scanning:** Automated vulnerability scanning (Snyk, Dependabot)
8. **Penetration Testing:** Quarterly security audits by external firms

Financial Security

Transaction Safety:

- **Idempotency Keys:** Prevent duplicate financial transactions
- **Two-Factor Authentication:** Required for investment transactions and withdrawals
- **Transaction Limits:** Daily/weekly limits for new accounts
- **Fraud Detection:** ML-based anomaly detection for suspicious activities
- **Audit Trail:** Immutable transaction logs for regulatory compliance

Compliance

- **PCI DSS:** If handling card data directly (recommend using payment processors)
- **KYC/AML:** Integration with identity verification services
- **Data Protection Acts:** Compliance with African data protection regulations
- **Financial Regulations:** Adherence to local financial services regulations

---

6. Scalability Considerations

Horizontal Scaling Strategy

Application Layer:

- **Stateless Services:** All microservices designed to be stateless for easy horizontal scaling
- **Auto-Scaling:** Kubernetes HPA (Horizontal Pod Autoscaler) based on CPU/memory metrics
- **Load Balancing:** Application Load Balancer distributing traffic across multiple instances
- **Blue-Green Deployments:** Zero-downtime deployments for continuous availability

Database Layer:

- **Read Replicas:** Multiple read replicas for query distribution (write to primary, read from replicas)
- **Connection Pooling:** PgBouncer for efficient database connection management
- **Query Optimization:** Database indexing strategy and query performance monitoring
- **Sharding Strategy:** User-based sharding for future growth beyond 10M users

Caching Strategy:

- **Multi-Level Caching:** Browser cache → CDN → Application cache → Database
- **Cache Invalidation:** Event-driven cache invalidation for data consistency
- **Redis Cluster:** Distributed Redis for high availability and data partitioning

Performance Optimization

Backend Optimization:

- **Asynchronous Processing:** Message queues for non-critical tasks (email sending, report generation)
- **Database Query Optimization:** Indexed queries, avoiding N+1 problems
- **API Response Compression:** Gzip compression for API responses
- **Pagination:** Cursor-based pagination for large datasets
- **Background Jobs:** Scheduled workers for data aggregation and analytics

Frontend Optimization:

- **Code Splitting:** Load only necessary JavaScript for each page
- **Lazy Loading:** Defer loading of below-the-fold content
- **Image Optimization:** WebP format, responsive images, lazy loading
- **Service Workers:** Offline functionality and instant page loads
- **Bundle Size:** Keep JavaScript bundles under 200KB (gzipped)

Data Management at Scale

Data Archival:

- **Hot Storage:** Last 12 months of data in primary database
- **Warm Storage:** 1-3 years in compressed partitions
- **Cold Storage:** 3+ years in archival storage (S3 Glacier)

Analytics Processing:

- **Data Pipeline:** Extract → Transform → Load (ETL) for analytics
- **Data Warehouse:** Separate analytics database (AWS Redshift or Snowflake)
- **Batch Processing:** Nightly aggregations for reporting dashboards

Monitoring & Observability

Metrics Collection:

- **APM:** Application Performance Monitoring (Datadog, New Relic, or AWS CloudWatch)
- **Custom Metrics:** Business metrics (user signups, transactions, revenue)
- **Real-User Monitoring:** Frontend performance tracking
- **Error Tracking:** Sentry for error reporting and debugging

Alerting:

- **Threshold Alerts:** CPU > 80%, error rate > 1%, response time > 500ms
- **Anomaly Detection:** ML-based alerts for unusual patterns
- **On-Call Rotation:** PagerDuty for incident management

Logging:

- **Centralized Logging:** ELK Stack (Elasticsearch, Logstash, Kibana) or AWS CloudWatch Logs
- **Structured Logging:** JSON-formatted logs for easy parsing
- **Log Retention:** 30 days hot, 1 year archived

---

7. Additional Considerations

Mobile-First Approach

Given the African market, we prioritize mobile experience:

- **Offline Mode:** Local data storage with background sync when online
- **Low Bandwidth Optimization:** Compressed API responses, progressive image loading
- **SMS Fallback:** Critical notifications via SMS for users without data
- **USSD Integration:** Basic functionality via USSD for feature phone users (Phase 2)

Payment Integration

Supported Methods:

- **Mobile Money:** M-Pesa, MTN Mobile Money, Airtel Money
- **Bank Transfers:** Integration with local banking APIs
- **Cards:** Visa/Mastercard via Flutterwave or Paystack
- **Crypto:** Bitcoin/stablecoin support for cross-border transactions (Phase 2)

Payment Service Providers:

- Primary: Flutterwave (Pan-African coverage)
- Secondary: Paystack (Nigeria, Ghana, South Africa)
- Tertiary: Direct bank integrations where available

Localization & Internationalization

- **Multi-Language Support:** English, French, Swahili, Hausa, Zulu (Phase 1)
- **Multi-Currency:** Support for major African currencies (KES, NGN, ZAR, GHS, UGX)
- **Regional Content:** Country-specific investment opportunities and financial products
- **Local Regulations:** Compliance framework for each operating country

Gamification & Engagement

- **Savings Challenges:** Community-based savings competitions
- **Financial Literacy:** Integrated educational content and quizzes
- **Achievements:** Badges for financial milestones
- **Leaderboards:** Anonymous rankings for motivation (with privacy controls)

Third-Party Integrations

Financial Data:

- **Bank Aggregation:** Okra, Mono (Open Banking APIs)
- **Market Data:** Alpha Vantage, Financial Modeling Prep
- **Credit Scoring:** Integration with credit bureaus

Analytics:

- **User Analytics:** Mixpanel or Amplitude
- **Business Intelligence:** Metabase or Looker for internal dashboards

Development Methodology

Agile Approach:

- **Sprint Duration:** 2-week sprints
- **MVP Timeline:** 4-6 months for core features
- **Beta Launch:** Month 5-6 with limited user group
- **Public Launch:** Month 7 with marketing campaign

- **Continuous Deployment:** Multiple releases per week post-launch

Team Structure (Recommended):

- 2 Backend Engineers (Node.js/TypeScript)
- 2 Frontend Engineers (React/React Native)
- 1 Mobile Engineer (Native optimization)
- 1 DevOps Engineer (Infrastructure, CI/CD)
- 1 QA Engineer (Automated testing)
- 1 UI/UX Designer
- 1 Product Manager
- 1 Technical Lead (You)

Cost Projections (First Year)

Infrastructure:

- Months 1-3 (0-10K users): $500-1,000/month
- Months 4-6 (10K-100K users): $2,000-5,000/month
- Months 7-12 (100K-1M users): $10,000-25,000/month

Optimization Strategies:

- Reserved instances for predictable workloads (40-60% cost reduction)
- Spot instances for batch processing (70-90% cost reduction)
- Auto-scaling to match demand patterns
- CDN caching to reduce origin server load

Risk Mitigation

Technical Risks:

- **Rapid Growth:** Over-provisioned infrastructure in early stages with auto-scaling triggers
- **Data Loss:** Automated daily backups with point-in-time recovery (7-day retention)
- **Security Breach:** WAF, DDoS protection, regular penetration testing, incident response plan
- **Service Outages:** Multi-AZ deployment, health checks, automatic failover (99.9% uptime SLA)

Business Risks:

- **Regulatory Changes:** Modular architecture allows quick adaptation to new regulations
- **Payment Provider Issues:** Multiple payment gateway integrations for redundancy
- **Scalability Bottlenecks:** Continuous load testing and performance monitoring

---

8. Success Metrics & KPIs

Technical KPIs:

- API Response Time: p95 < 500ms
- Uptime: 99.9% availability
- Error Rate: < 0.1% of requests
- Page Load Time: < 3 seconds on 3G connection
- Database Query Performance: p95 < 100ms

Business KPIs:

- User Acquisition: 1M users in 12 months
- Daily Active Users (DAU): Target 30% of total users
- User Retention: 60% after 30 days
- Transaction Volume: Track monthly transaction growth
- Customer Support Response Time: < 2 hours

---

9. Roadmap & Phases

Phase 1 (Months 1-4): MVP Development

- User authentication and profile management
- Expense tracking and categorization
- Budget creation and monitoring
- Basic financial goals
- Web and mobile apps (iOS/Android)

Phase 2 (Months 5-6): Beta Launch

- Investment portfolio tracking
- Bank account integration
- Financial insights and analytics
- Push notifications
- Limited beta user testing

Phase 3 (Months 7-8): Public Launch

- Full marketing campaign leveraging Tony's platform
- Referral program
- Enhanced security features
- Performance optimizations
- Customer support system

Phase 4 (Months 9-12): Scale & Enhance

- Investment recommendations
- Peer-to-peer lending (subject to regulatory approval)
- Advanced financial planning tools
- AI-powered financial advisor
- Multi-country expansion

---

10. Why Kanzu Finance Limited

Our approach demonstrates:

1. **Scalability-First Design:** Architecture built to handle 10M+ users from day one
2. **African Market Understanding:** Mobile-first, offline-capable, payment integration expertise
3. **Security & Compliance:** Financial-grade security with regulatory awareness
4. **Cost-Effectiveness:** Smart technology choices that optimize development and hosting costs
5. **Proven Technologies:** Battle-tested stack used by leading fintech companies globally
6. **Rapid Execution:** Agile methodology with MVP delivery in 4-6 months
7. **Future-Proof:** Modular architecture allows easy feature additions and market expansion

We're excited to partner with Tony to build Africa's leading personal finance platform and empower the next generation of financially savvy young Africans.

Next Steps:

1. Technical review and feedback session
2. Detailed project timeline and milestones
3. Resource allocation and team formation
4. Contract and commercials discussion
5. Project kickoff

Contact: Kanzu Finance Limited
Lead Engineer