# 

**B. TECH (CSE) - V SEM**

**UE23CS341A - Software Engineering**

**PROJECT DOCUMENTATION ON**

**Software Architecture and Design Specification**

**SEC: C**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **SRN** | **Name** |
| 1 | PES1UG23CS154 | C Kaustubh |
| 2 | PES1UG23CS166 | Chinmay Shivanand Muragod |
| 3 | PES1UG23CS167 | Chirag K M |
| 4 | PES1UG23CS173 | Darshith M S |

**AUGUST – DECEMBER 2025**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**RR CAMPUS (MAIN CAMPUS)**

**BENGALURU – 560085, KARNATAKA**

Software Architecture and Design Specification

**Project:** Digital Asset and Cryptocurrency Portfolio Tracker  
**Version:** 1.0  
**Authors:** C Kaustubh, Chinmay Shivanand Muragod, Chirag K M, Darshith M S  
**Date:** 11-09-2025  
**Status:** Draft

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change Summary** | **Approval** |
| 1.0 | 11-09-2025 | Team | Initial SAD with architecture and design specifications |  |

**Approvals**

|  |  |  |
| --- | --- | --- |
| **Role** | **Name** | **Signature/Date** |
| QA Lead | Chinmay Shivanand Muragod |  |
| Test Team Lead | Chirag K M |  |
| Security Test Lead | Darshith M S |  |
| Dev Lead | C Kaustubh |  |
| Course Coordinator | Dr. Radhika M J |  |

**1. Introduction**

**1.1 Purpose**

This document specifies the software architecture and design of the Digital Asset and Cryptocurrency Portfolio Tracker System. It provides detailed architectural decisions, component designs, API specifications, and security considerations for developers, QA engineers, security auditors, and maintenance teams.

**1.2 Scope**

Covers the complete system architecture including user authentication with 2FA, portfolio management, real-time market data integration, performance analytics, exchange API integrations, NFT management, tax reporting, mobile applications, and web dashboard interfaces.

**1.3 Audience**

Software developers, QA engineers, system architects, security specialists, DevOps engineers, database administrators, instructors, and system maintenance teams.

**1.4 Definitions**

*API* (Application Programming Interface), *DeFi* (Decentralized Finance),

*NFT* (Non-Fungible Token), *2FA* (Two-Factor Authentication), *JWT* (JSON Web Token), *TOTP* (Time-based One-Time Password), *OAuth* (Open Authorization), WebSocket,

*REST* (Representational State Transfer), *GDPR* (General Data Protection Regulation),

*AES* (Advanced Encryption Standard), *TLS* (Transport Layer Security).

**1.5 System Overview**

The “*Digital Asset and Cryptocurrency Portfolio Tracker”* is a comprehensive solution for managing diverse digital asset portfolios in today’s evolving market. It combines real-time market data, advanced security, and intuitive design to deliver a professional-grade experience. Supporting cryptocurrencies, DeFi tokens, NFTs, and staking rewards, the platform provides users with a complete view of their investments.

**1.6 Key System Features**

* Multi-exchange integration (10+ major platforms)
* AES-256 security with MFA & HSM
* Real-time analytics (sub-10s updates)
* Web, iOS & Android sync
* Automated multi-jurisdiction tax reports

**2. Document Overview**

**2.1 How to use this document**

This document provides comprehensive architectural deliverables including UML component and sequence diagrams, Architecture Decision Records (ADRs), threat models using STRIDE methodology, API design specifications, technology stack decisions, and implementation guidelines for the cryptocurrency portfolio tracking system.

**2.2 Related Documents**

* Digital Asset Portfolio Tracker SRS v1.0
* Software Test Plan (STP) v1.0
* Requirements Traceability Matrix (RTM)

**3. Architecture**

**3.1 Goals & Constraints**

**Goals:**

* High security with AES-256 encryption and 2FA authentication
* Real-time market data processing with <10 second update latency
* 99.5% system availability with automatic failover capabilities
* Scalability to support 1000+ concurrent users
* Multi-platform support (web, iOS, Android)
* Seamless integration with 10+ cryptocurrency exchanges

**Constraints:**

* Cryptocurrency exchange API rate limits (varies by provider)
* GDPR compliance requirements for user data protection
* Mobile platform restrictions (iOS App Store, Google Play policies)
* Real-time market data licensing costs and limitations
* Regulatory compliance requirements (KYC/AML where applicable)
* Hardware security module requirements for API key storage
  1. **Stakeholders & Concerns**

Individual Investors:

Security of portfolio data, intuitive user interface, accurate real-time pricing, mobile accessibility

Professional Traders:

Advanced analytics, multiple exchange support, low-latency data feeds, comprehensive reporting

Portfolio Managers:

Multi-client support, compliance features, detailed audit trails, tax reporting capabilities

System Administrators:

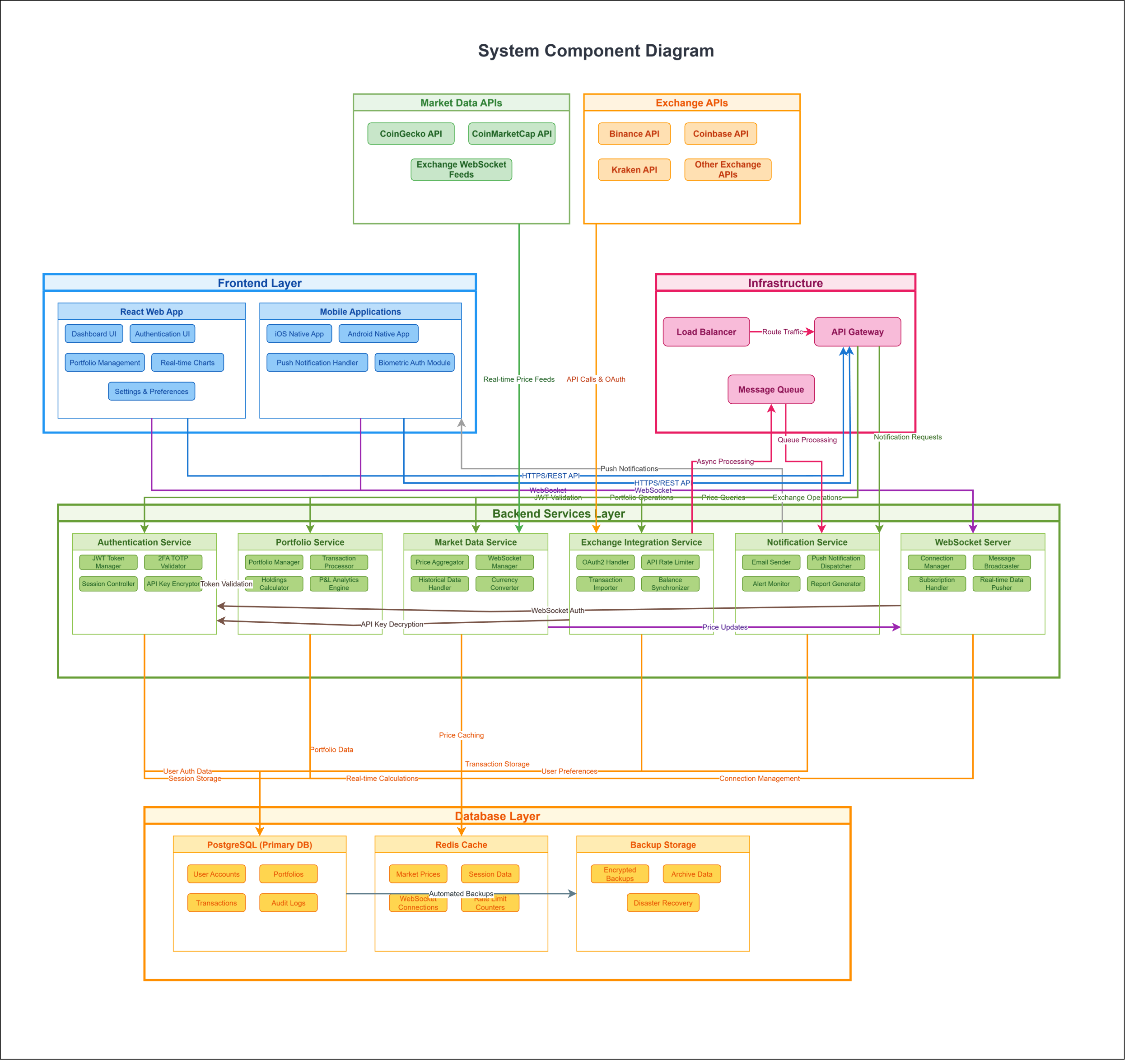
System monitoring, scalability, backup and recovery, security incident response

Regulators:

Data privacy compliance, audit trails, security standards adherence

Developers:

Code maintainability, modular architecture, comprehensive API documentation, testing frameworks

**3.3 Component (UML) Diagram**

**3.4 Component Descriptions**

**React Frontend Application:**

* Responsive web dashboard for portfolio management
* Real-time price charts and performance analytics
* User authentication interface with 2FA support
* Portfolio creation and asset management UI

**Rust Backend Services:**

* Authentication and session management service
* Portfolio management and transaction processing service
* Market data aggregation and caching service
* Exchange API integration and synchronization service
* Analytics and reporting service
* Notification and alert service

**Database Layer:**

* PostgreSQL primary database for user accounts, portfolios, and transactions
* Redis cache for real-time market data and session storage
* Encrypted backup storage for disaster recovery

**Exchange Integration Module:**

* OAuth 2.0 authentication with supported exchanges
* API rate limiting and retry mechanisms
* Transaction import and synchronization
* Real-time balance updates

**Market Data Service:**

* WebSocket connections for real-time price feeds
* Historical data retrieval and storage
* Price calculation and percentage change analytics
* Multi-currency conversion support

**Authentication Service:**

* JWT token management and validation
* TOTP-based 2FA implementation
* Session timeout and security controls
* API key encryption and secure storage

**Notification Service:**

* Email notification system for portfolio updates
* Push notification service for mobile apps
* Price alert monitoring and triggering
* Scheduled report generation and delivery

**Mobile Applications:**

* Native iOS and Android apps
* Core portfolio tracking features
* Push notification support
* Biometric authentication integration

**3.5 Chosen Architecture Pattern and Rationale**

**Microservices Architecture with API Gateway Pattern**

**Rationale:**

* *Scalability:* Individual services can be scaled independently based on demand
* *Technology Diversity:* React frontend with Rust backend optimizes performance
* *Fault Isolation:* Failure in one service doesn't affect the entire system
* *Development Flexibility:* Teams can work on different services independently
* *Exchange Integration:* Isolated services for each exchange integration reduce complexity

**Alternative Patterns Considered:**

* *Monolithic Architecture:* Rejected due to scalability limitations and tight coupling
* *Event-Driven Architecture:* Considered for real-time updates but added unnecessary complexity
* *Serverless Architecture:* Rejected due to cold start latency and state management challenges

**3.6 Technology Stack & Data Stores**

**Frontend Technologies:**

* React 18 with TypeScript for type safety
* Material-UI for consistent design components
* Chart.js for portfolio performance visualizations
* WebSocket client for real-time data updates
* React Query for server state management

**Backend Technologies:**

* Rust with Tokio for high-performance async processing
* Axum web framework for HTTP API endpoints
* SeaORM for database operations and migrations
* JSON Web Tokens (JWT) for authentication
* WebSocket server for real-time client communication

**Database Technologies:**

* PostgreSQL 15 for primary data storage
* Redis 7 for caching and session management
* Database connection pooling for performance optimization

**Infrastructure & DevOps:**

* Docker containers for application deployment
* Kubernetes for orchestration and auto-scaling
* Nginx as reverse proxy and load balancer
* AWS S3 for file storage and backups
* CloudWatch for monitoring and logging

**Security Technologies:**

* TLS 1.3 for all external communications
* AES-256 encryption for data at rest
* bcrypt for password hashing
* OAuth 2.0 for exchange API authentication
* Rate limiting middleware for DDoS protection

**3.7 Risks & Mitigations**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Impact** | **Probability** | **Mitigation Strategy** |
| Exchange API downtime | High | Medium | Implement circuit breakers, fallback data sources, and graceful degradation |
| Real-time data feed interruption | Medium | Medium | Multiple data provider integration, local caching, and automatic failover |
| Database performance bottlenecks | High | Low | Database optimization, read replicas, connection pooling, and query optimization |
| Security vulnerabilities | Critical | Low | Regular security audits, penetration testing, automated vulnerability scanning |
| Scalability limitations under load | High | Medium | Auto-scaling infrastructure, load testing, and performance monitoring |
| Mobile app store rejections | Medium | Low | Compliance review, beta testing, and alternative distribution methods |

**3.8 Traceability to Requirements**

**Authentication & Security:**

* CRYPTO-F-001 (Email verification) → Authentication Service email validation module
* CRYPTO-F-002 (2FA enforcement) → TOTP implementation in Authentication Service
* CRYPTO-F-003 (Session timeout) → JWT token expiration and session management

**Portfolio Management:**

* CRYPTO-F-004 (Multiple portfolios) → Portfolio Management Service with multi-portfolio support
* CRYPTO-F-005 (Manual holdings) → Transaction input validation and storage components
* CRYPTO-F-006 (Exchange import) → Exchange Integration Module automated sync

**Market Data & Analytics:**

* CRYPTO-F-007 (Real-time prices) → Market Data Service WebSocket implementation
* CRYPTO-F-010 (multi-currency value) → Analytics Service currency conversion
* CRYPTO-F-011 (P&L calculations) → Analytics Service tax calculation engine

**Non-Functional Requirements:**

* CRYPTO-NF-001 (Load time ≤3s) → Frontend optimization and caching strategies
* CRYPTO-NF-002 (99.5% uptime) → High availability architecture with redundancy
* CRYPTO-NF-004 (1000+ concurrent users) → Microservices scalability and load balancing

**3.9 Security Architecture**

**Threat Modelling (STRIDE Analysis):**

**Spoofing Identity:**

* Threat: Unauthorized access to user accounts
* Mitigation: 2FA with TOTP, strong password requirements, account lockout policies

**Tampering with Data:**

* Threat: Modification of portfolio data or transactions
* Mitigation: Database integrity constraints, audit logging, data validation at API layer

**Repudiation:**

* Threat: Users denying performed actions
* Mitigation: Comprehensive audit logs, digital signatures, timestamp verification

**Information Disclosure:**

* Threat: Exposure of sensitive portfolio or personal data
* Mitigation: AES-256 encryption at rest, TLS 1.3 in transit, access control lists

**Denial of Service:**

* Threat: System unavailability due to resource exhaustion
* Mitigation: Rate limiting, DDoS protection, auto-scaling, circuit breakers

**Elevation of Privilege:**

* Threat: Unauthorized access to admin functions or other users' data
* Mitigation: Role-based access control (RBAC), principle of least privilege, input validation

**Security Controls:**

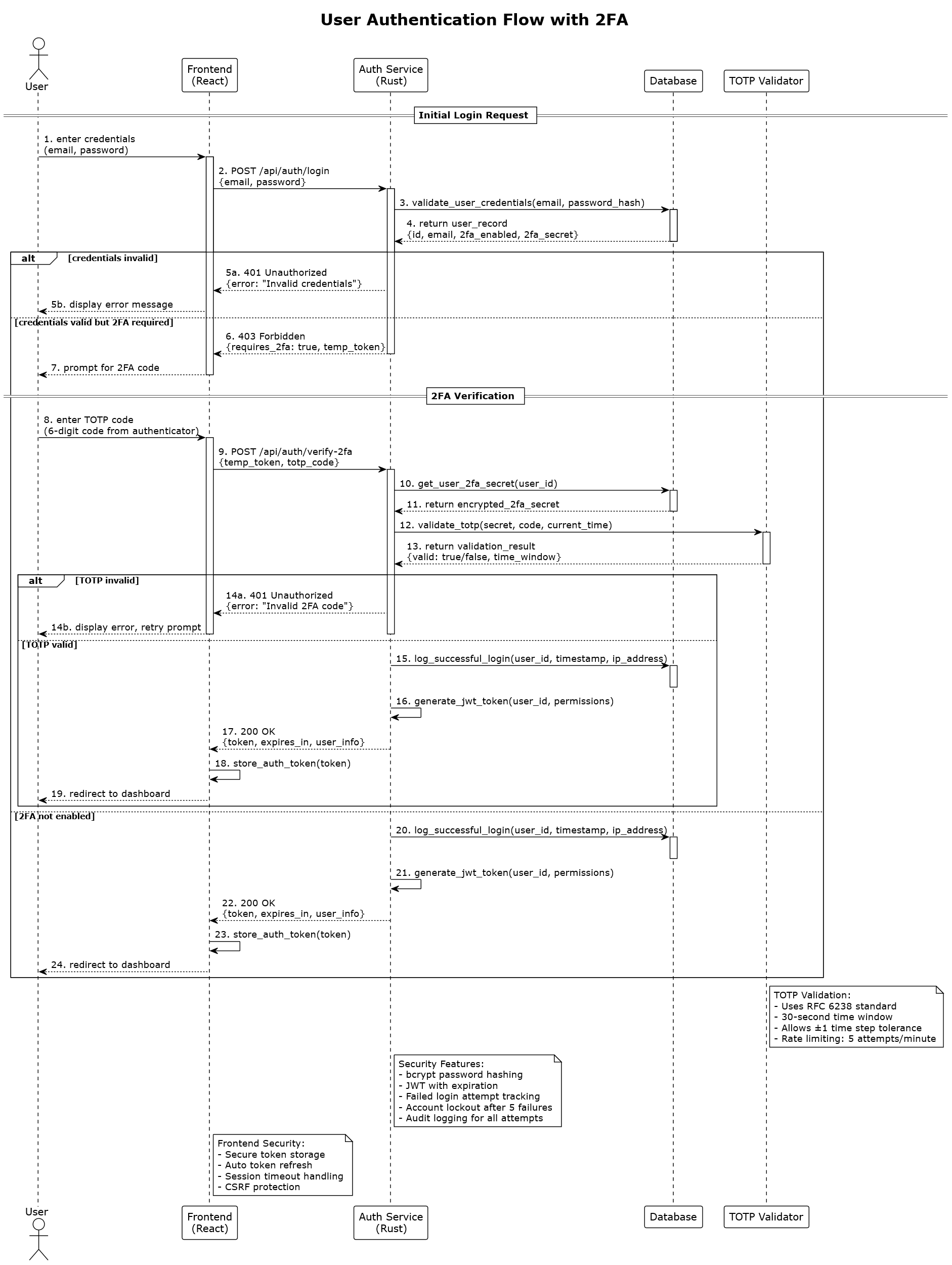
* API key encryption using AES-256
* OAuth 2.0 for exchange integrations
* JWT token validation on all protected endpoints
* Input sanitization and validation
* SQL injection prevention through parameterized queries
* XSS protection through Content Security Policy (CSP)

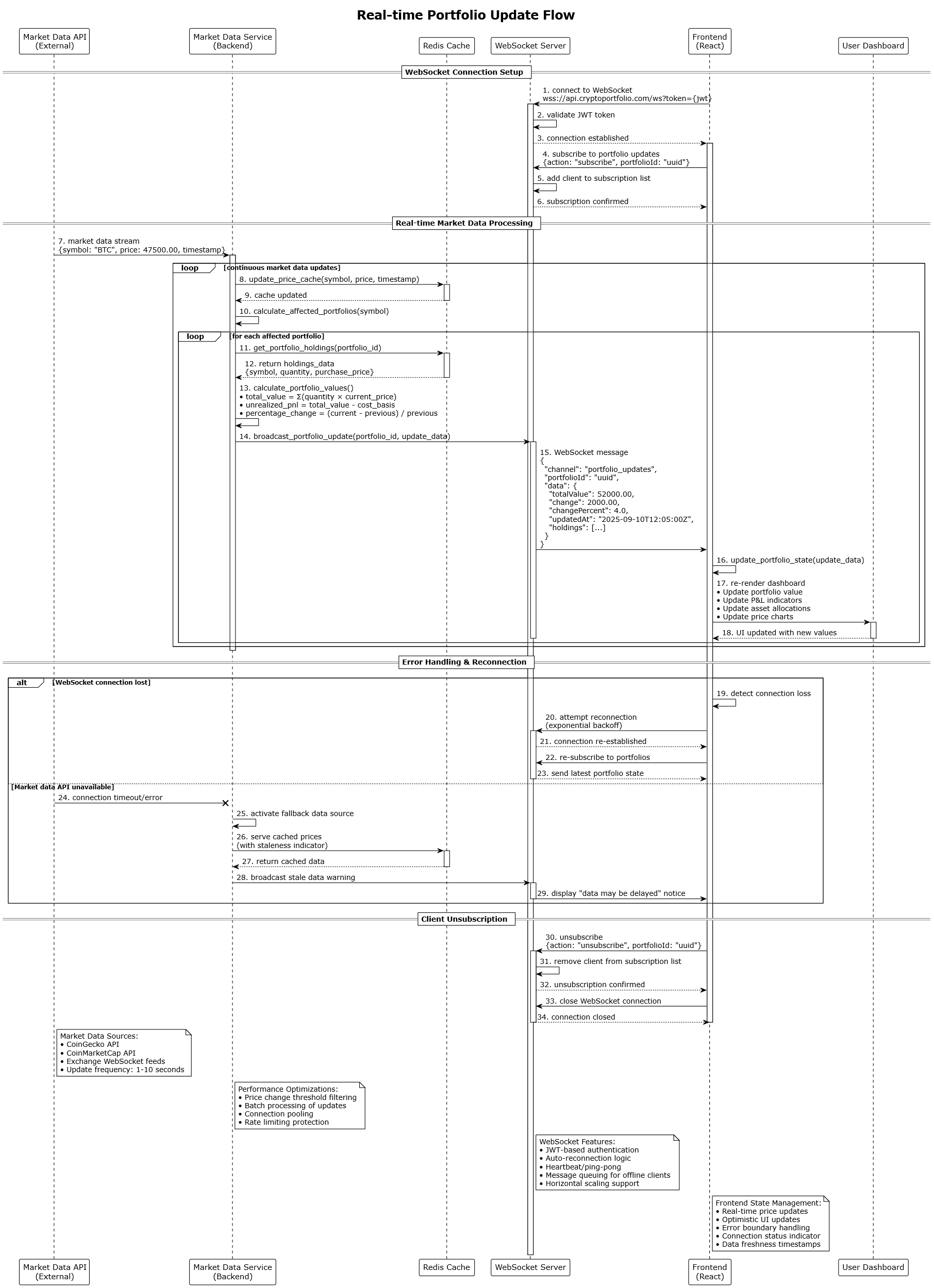
**4. Design**

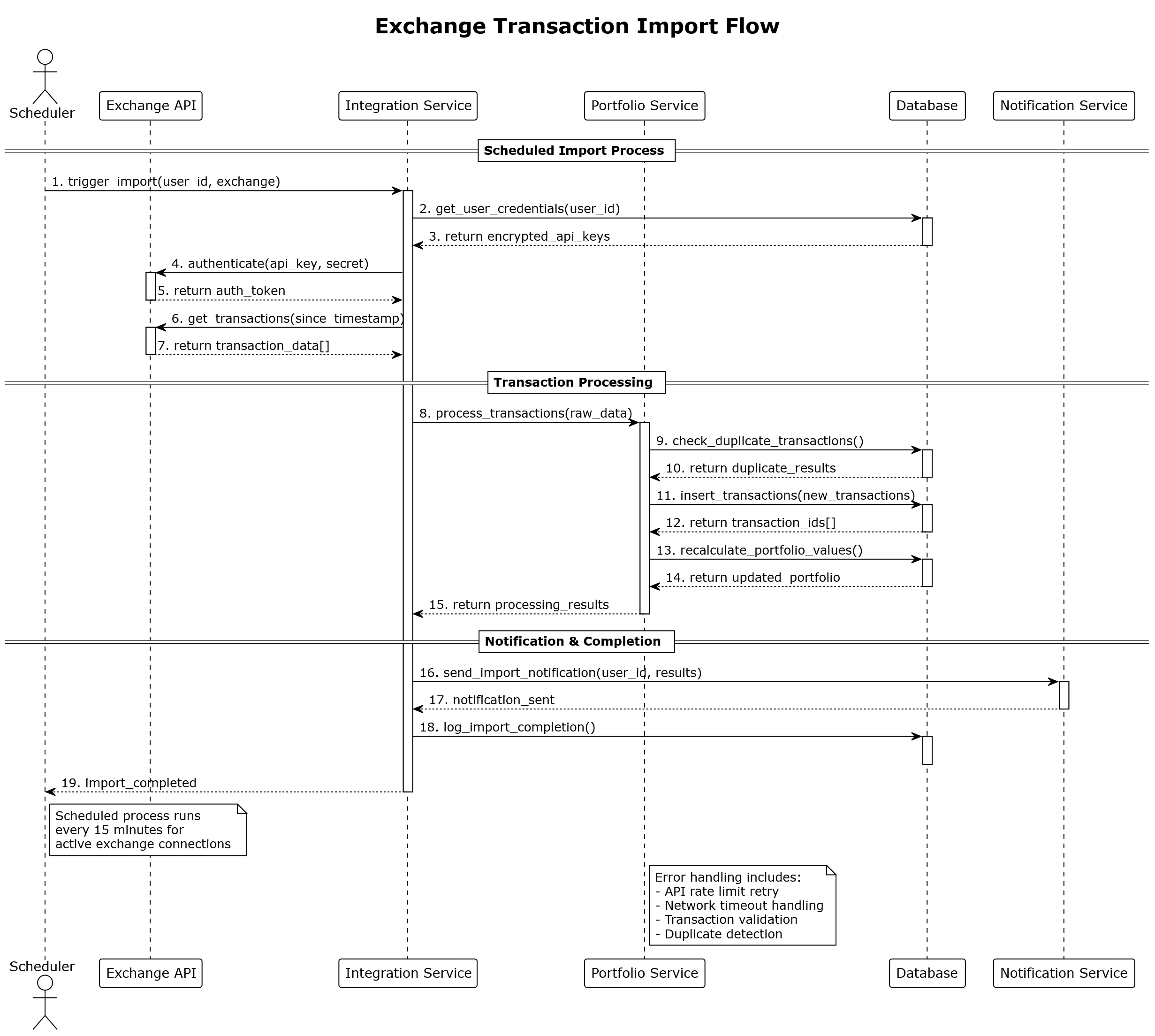
**4.1 Design Overview**

The cryptocurrency portfolio tracker employs a modern, scalable design with React frontend communicating with Rust backend services through RESTful APIs and WebSocket connections. The system follows Domain-Driven Design principles with clear separation between authentication, portfolio management, market data, and analytics domains. Each service maintains its own data boundaries while communicating through well-defined interfaces.

**4.2 UML Sequence Diagrams**

**

**



**4.3 API Design**

**Authentication Endpoints:**

Endpoint: /api/auth/register

Method: POST

Request: {

  "email": "user@example.com",

  "password": "securePassword123",

  "confirmPassword": "securePassword123"

}

Response: {

  "status": "success",

  "message": "Verification email sent",

  "userId": "uuid-string"

}

Errors: 400 Invalid email format, 409 User already exists

Endpoint: /api/auth/login

Method: POST

Request: {

  "email": "user@example.com",

  "password": "securePassword123",

  "totpCode": "123456"

}

Response: {

  "status": "success",

  "token": "jwt-token-string",

  "expiresIn": 3600,

  "user": {

    "id": "uuid",

    "email": "user@example.com",

    "verified": true

  }

}

Errors: 401 Invalid credentials, 403 2FA required, 423 Account locked

**Portfolio Management Endpoints:**

Endpoint: /api/portfolios

Method: GET

Headers: Authorization: Bearer {jwt-token}

Response: {

  "portfolios": [

    {

      "id": "portfolio-uuid",

      "name": "Main Portfolio",

      "description": "Primary investment portfolio",

      "totalValue": 50000.00,

      "currency": "USD",

      "assetCount": 12,

      "createdAt": "2025-09-01T00:00:00Z"

    }

  ]

}

Errors: 401 Unauthorized, 403 Forbidden

Endpoint: /api/portfolios/{id}/holdings

Method: POST

Headers: Authorization: Bearer {jwt-token}

Request: {

  "symbol": "BTC",

  "quantity": 1.5,

  "purchasePrice": 45000.00,

  "purchaseDate": "2025-09-01T00:00:00Z",

  "notes": "Dollar cost averaging purchase"

}

Response: {

  "status": "success",

  "holding": {

    "id": "holding-uuid",

    "symbol": "BTC",

    "quantity": 1.5,

    "currentPrice": 47000.00,

    "totalValue": 70500.00,

    "unrealizedGain": 3000.00

  }

}

Errors: 400 Invalid asset symbol, 422 Invalid quantity

**Market Data Endpoints:**

Endpoint: /api/market/prices/{symbol}

Method: GET

Response: {

  "symbol": "BTC",

  "currentPrice": 47000.00,

  "currency": "USD",

  "change24h": 2.5,

  "changePercent24h": 5.32,

  "volume24h": 28500000000,

  "lastUpdated": "2025-09-10T12:00:00Z"

}

Errors: 404 Symbol not found, 503 Data unavailable

**WebSocket API:**

Connection: wss://api.cryptoportfolio.com/ws

Authentication: ?token={jwt-token}

Subscribe Message: {

  "action": "subscribe",

  "channel": "portfolio\_updates",

  "portfolioId": "portfolio-uuid"

}

Update Message: {

  "channel": "portfolio\_updates",

  "data": {

    "portfolioId": "portfolio-uuid",

    "totalValue": 52000.00,

    "change": 2000.00,

    "changePercent": 4.0,

    "updatedAt": "2025-09-10T12:05:00Z"

  }

}

**4.4 Error Handling, Logging & Monitoring**

**Error Handling Strategy:**

* Standardized error response format across all APIs
* HTTP status codes following REST conventions
* Detailed error messages for development, sanitized for production
* Graceful degradation when external services are unavailable
* Circuit breaker pattern for external API calls

**Logging Framework:**

* Structured JSON logging using slog for Rust services
* Log levels: ERROR, WARN, INFO, DEBUG, TRACE
* No sensitive data (passwords, API keys, personal info) in logs
* Correlation IDs for request tracing across services
* Log retention policy: 30 days for application logs, 90 days for audit logs

**Monitoring Metrics:**

* API response times and success rates
* Database query performance and connection pool usage
* Exchange API call success rates and latency
* WebSocket connection counts and message throughput
* Portfolio valuation calculation accuracy and performance
* User authentication success/failure rates
* System resource utilization (CPU, memory, disk, network)

**Alerting Thresholds:**

* API response time > 5 seconds
* Error rate > 5% over 5-minute window
* Database connection pool > 80% utilization
* Exchange API failure rate > 10%
* System availability < 99.5%

**4.5 UX Design**

**Web Dashboard Design:**

* Responsive design supporting desktop, tablet, and mobile viewports
* Dark/light theme toggle with user preference persistence
* Accessible color contrast ratios meeting WCAG 2.1 AA standards
* Intuitive navigation with breadcrumb trails
* Real-time data updates without full page refreshes
* Progressive loading for large portfolio datasets

**Mobile Application Design(Trial – optional):**

* Native Android applications with platform-specific design patterns
* Biometric authentication support (Touch ID, Face ID, fingerprint)
* Offline mode for viewing cached portfolio data
* Push notification management and customization
* Quick action widgets for portfolio overview
* Accessibility support including screen reader compatibility

**Key UX Principles:**

* Security-first design with clear security indicators
* Performance optimization with skeleton screens during loading
* Error states with actionable recovery instructions
* Consistent design system across web and mobile platforms
* Multi-language support for international users

**4.6 Open Issues & Next Steps**

**Current Open Issues:**

* Exchange API rate limiting optimization for high-frequency users
* Mobile app performance optimization for portfolios with 100+ assets
* Real-time data synchronization conflicts during high market volatility
* Tax calculation accuracy verification across multiple jurisdictions

**Future Enhancements (Roadmap):**

* Biometric authentication for web applications using WebAuthn
* Advanced portfolio analytics including risk metrics and backtesting
* Social trading features and portfolio sharing
* Integration with DeFi protocols for yield farming tracking
* Machine learning-based price prediction and trend analysis
* Support for additional asset classes (stocks, bonds, commodities)
* API access for third-party developers and institutional clients
* Enhanced NFT marketplace integration and valuation tools

**Technical Debt:**

* Database schema optimization for improved query performance
* Frontend component library standardization and documentation
* API versioning strategy implementation
* Comprehensive integration test coverage for exchange APIs

**5. Appendices**

**5.1 Glossary**

**API** : Set of protocols and tools for building software applications  
**DeFi** : Financial services built on blockchain technology  
**NFT**  : Unique digital assets stored on blockchain  
**2FA**  : Security process requiring two forms of identification  
**JWT**  : Compact, URL-safe token format for secure information transmission  
**TOTP** : Algorithm generating temporary passwords  
**OAuth** : Authorization framework for third-party service access  
**REST** : Architectural style for distributed systems  
**GDPR** : EU data protection and privacy regulation  
**AES** : Symmetric encryption specification  
**TLS**  : Cryptographic protocol for secure communication  
**STRIDE** : Threat modelling methodology (Spoofing, Tampering, Repudiation, Information

Disclosure, Denial of Service, Elevation of Privilege)

**WebSocket**: Communication protocol providing full-duplex channels

**5.2 References**

* IEEE 42010-2011: Systems and Software Engineering Architecture Description
* OWASP Top 10 2021: Web Application Security Risks
* NIST SP 800-160: Systems Security Engineering
* RFC 7519: JSON Web Token (JWT)
* RFC 6238: TOTP Time-Based One-Time Password Algorithm
* React 18 Documentation
* Rust Programming Language Documentation
* PostgreSQL 15 Documentation

**5.3 Tools & Technologies**

**Design & Documentation:**

* Draw.io for UML diagram generation
* Draw.io for architectural diagrams
* Swagger/OpenAPI for API documentation
* Figma for UI/UX design mockups

**Development Tools:**

* Visual Studio Code with Rust and TypeScript extensions
* Git for version control with GitFlow workflow
* Docker for containerization
* Kubernetes for orchestration

**Testing Tools:**

* Jest and React Testing Library for frontend testing
* Cargo test framework for Rust unit testing
* Cypress for end-to-end testing
* Postman for API testing

**Monitoring & DevOps:**

* Prometheus for metrics collection
* Grafana for visualization and dashboards
* ELK Stack (Elasticsearch, Logstash, Kibana) for log management
* Jenkins for CI/CD pipelines

***Thank You***