**AI-Powered Crypto Risk Dashboard - Software Requirements Specification (SRS)**

**1. Project Overview**

**1.1 Purpose**

The AI-Powered Crypto Risk Dashboard is a web application that provides real-time portfolio risk analysis for cryptocurrency investments using advanced AI/ML techniques and on-chain data analysis.

**1.2 Problem Statement**

Current portfolio trackers focus on displaying numbers (prices, gains/losses) but fail to provide actionable risk insights. Most retail crypto investors lack sophisticated risk management tools, leading to poor diversification and exposure to unnecessary volatility.

**1.3 Solution Overview**

A comprehensive dashboard that combines real-time market data with AI-powered analysis to provide:

* Visual risk assessment through interactive heatmaps
* Personalized portfolio insights using LLM technology
* Automated alerts for risk threshold breaches
* Actionable recommendations for portfolio optimization

**2. System Architecture**

**2.1 High-Level Architecture**

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

│ Frontend │ │ Backend │ │ External APIs │

│ (Next.js) │◄──►│ (FastAPI) │◄──►│ (CoinGecko, │

│ │ │ │ │ Etherscan) │

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

│

┌──────▼──────┐

│ Database │

│(PostgreSQL/ │

│ TimescaleDB)│

└─────────────┘

│

┌──────▼──────┐

│ AI/ML │

│ (LangChain/ │

│ OpenAI) │

└─────────────┘

**2.2 Component Architecture**

Frontend (Next.js + TypeScript)

├── pages/

│ ├── dashboard/ # Main portfolio overview

│ ├── risk-analysis/ # Detailed risk metrics

│ ├── insights/ # AI-generated insights

│ └── alerts/ # Alert management

├── components/

│ ├── charts/ # Risk heatmaps, allocation charts

│ ├── portfolio/ # Portfolio management UI

│ ├── alerts/ # Alert configuration

│ └── common/ # Shared UI components

├── hooks/ # Custom React hooks

├── utils/ # Helper functions

└── types/ # TypeScript definitions

Backend (FastAPI + Python)

├── api/

│ ├── portfolio/ # Portfolio CRUD operations

│ ├── risk/ # Risk calculation endpoints

│ ├── insights/ # AI insights endpoints

│ └── alerts/ # Alert management endpoints

├── core/

│ ├── risk\_engine/ # Risk calculation logic

│ ├── ai\_engine/ # LangChain integration

│ └── data\_fetchers/ # External API integrations

├── models/ # Database models

├── schemas/ # Pydantic schemas

└── utils/ # Helper functions

**3. Functional Requirements**

**3.1 User Stories**

**Epic 1: Portfolio Management**

* **US-001**: As a user, I want to add cryptocurrencies to my portfolio so that I can track my investments
* **US-002**: As a user, I want to input my holdings (amount, purchase price) so the system can calculate my current position
* **US-003**: As a user, I want to connect my wallet addresses so the system can automatically track my holdings

**Epic 2: Risk Analysis**

* **US-004**: As a user, I want to see a visual heatmap of my portfolio risk so I can quickly identify high-risk positions
* **US-005**: As a user, I want to see volatility metrics for each holding so I can understand individual asset risk
* **US-006**: As a user, I want to see portfolio-level risk metrics (Sharpe ratio, VaR) so I can assess overall portfolio health
* **US-007**: As a user, I want to see concentration risk analysis so I can identify over-allocation issues

**Epic 3: AI Insights**

* **US-008**: As a user, I want weekly AI-generated portfolio insights so I can receive personalized recommendations
* **US-009**: As a user, I want AI explanations of risk metrics so I can understand what they mean for my portfolio
* **US-010**: As a user, I want AI-powered rebalancing suggestions so I can optimize my allocation

**Epic 4: Alerts & Monitoring**

* **US-011**: As a user, I want to set price alerts so I can be notified of significant market movements
* **US-012**: As a user, I want risk threshold alerts so I can be warned when my portfolio becomes too risky
* **US-013**: As a user, I want to receive alerts via email and Telegram so I can choose my preferred notification method

**3.2 Acceptance Criteria**

**Portfolio Heatmap (US-004)**

* Display portfolio holdings as colored tiles based on risk level (green=low, yellow=medium, red=high)
* Tile size represents allocation percentage
* Hover shows detailed risk metrics
* Click drills down to individual asset analysis
* Updates in real-time with price changes

**AI Weekly Insights (US-008)**

* Generate comprehensive weekly portfolio analysis using GPT-4
* Include specific recommendations (e.g., "Consider reducing Bitcoin allocation from 65% to 45%")
* Highlight concentration risks and diversification opportunities
* Compare portfolio performance to market benchmarks
* Provide sentiment analysis based on recent market conditions

**4. Database Schema Design**

**4.1 Core Tables**

-- Users and Authentication

CREATE TABLE users (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Portfolio Management

CREATE TABLE portfolios (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

name VARCHAR(100) NOT NULL,

description TEXT,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

CREATE TABLE crypto\_assets (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

symbol VARCHAR(20) UNIQUE NOT NULL,

name VARCHAR(100) NOT NULL,

coingecko\_id VARCHAR(100),

contract\_address VARCHAR(100),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

CREATE TABLE portfolio\_holdings (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

portfolio\_id UUID REFERENCES portfolios(id) ON DELETE CASCADE,

asset\_id UUID REFERENCES crypto\_assets(id),

quantity DECIMAL(36, 18) NOT NULL,

average\_cost DECIMAL(18, 8),

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Time Series Data (TimescaleDB Hypertables)

CREATE TABLE price\_history (

time TIMESTAMP WITH TIME ZONE NOT NULL,

asset\_id UUID REFERENCES crypto\_assets(id),

price\_usd DECIMAL(18, 8) NOT NULL,

volume\_24h DECIMAL(20, 2),

market\_cap DECIMAL(20, 2),

PRIMARY KEY (time, asset\_id)

);

SELECT create\_hypertable('price\_history', 'time');

CREATE TABLE risk\_metrics (

time TIMESTAMP WITH TIME ZONE NOT NULL,

portfolio\_id UUID REFERENCES portfolios(id),

volatility\_30d DECIMAL(8, 4),

volatility\_90d DECIMAL(8, 4),

sharpe\_ratio DECIMAL(8, 4),

max\_drawdown DECIMAL(8, 4),

var\_95 DECIMAL(18, 8),

concentration\_risk DECIMAL(8, 4),

PRIMARY KEY (time, portfolio\_id)

);

SELECT create\_hypertable('risk\_metrics', 'time');

-- AI Insights

CREATE TABLE ai\_insights (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

portfolio\_id UUID REFERENCES portfolios(id) ON DELETE CASCADE,

insight\_type VARCHAR(50) NOT NULL, -- 'weekly', 'alert', 'rebalancing'

content TEXT NOT NULL,

risk\_score INTEGER CHECK (risk\_score >= 1 AND risk\_score <= 10),

recommendations JSONB,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Alert System

CREATE TABLE alert\_rules (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

user\_id UUID REFERENCES users(id) ON DELETE CASCADE,

rule\_type VARCHAR(50) NOT NULL, -- 'price\_change', 'risk\_threshold', 'portfolio\_concentration'

conditions JSONB NOT NULL,

notification\_channels JSONB NOT NULL, -- ['email', 'telegram']

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

CREATE TABLE alert\_history (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

rule\_id UUID REFERENCES alert\_rules(id) ON DELETE CASCADE,

triggered\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

message TEXT NOT NULL,

notification\_sent BOOLEAN DEFAULT FALSE

);

**4.2 Indexes for Performance**

-- Optimize time-series queries

CREATE INDEX idx\_price\_history\_asset\_time ON price\_history (asset\_id, time DESC);

CREATE INDEX idx\_risk\_metrics\_portfolio\_time ON risk\_metrics (portfolio\_id, time DESC);

-- Optimize portfolio queries

CREATE INDEX idx\_portfolio\_holdings\_portfolio ON portfolio\_holdings (portfolio\_id);

CREATE INDEX idx\_ai\_insights\_portfolio\_created ON ai\_insights (portfolio\_id, created\_at DESC);

**5. API Endpoint Specifications**

**5.1 Portfolio Endpoints**

GET /api/v1/portfolios

- Get user's portfolios

- Returns: List of portfolios with basic info

POST /api/v1/portfolios

- Create new portfolio

- Body: {name, description}

- Returns: Created portfolio object

GET /api/v1/portfolios/{portfolio\_id}

- Get portfolio details with current holdings

- Returns: Portfolio with holdings, current values, allocations

POST /api/v1/portfolios/{portfolio\_id}/holdings

- Add/update holding in portfolio

- Body: {asset\_symbol, quantity, average\_cost}

- Returns: Updated holding object

DELETE /api/v1/portfolios/{portfolio\_id}/holdings/{holding\_id}

- Remove holding from portfolio

- Returns: Success confirmation

**5.2 Risk Analysis Endpoints**

GET /api/v1/portfolios/{portfolio\_id}/risk-metrics

- Get current risk metrics for portfolio

- Query params: ?period=30d|90d|1y

- Returns: {volatility, sharpe\_ratio, max\_drawdown, var\_95, concentration\_risk}

GET /api/v1/portfolios/{portfolio\_id}/risk-heatmap

- Get data for risk heatmap visualization

- Returns: Holdings with risk scores, colors, allocations

GET /api/v1/portfolios/{portfolio\_id}/correlation-matrix

- Get correlation matrix between portfolio holdings

- Returns: Matrix data for visualization

**5.3 AI Insights Endpoints**

GET /api/v1/portfolios/{portfolio\_id}/insights

- Get AI insights for portfolio

- Query params: ?type=weekly|alert|rebalancing&limit=10

- Returns: List of AI-generated insights

POST /api/v1/portfolios/{portfolio\_id}/insights/generate

- Trigger new AI insight generation

- Body: {insight\_type}

- Returns: Generated insight object

GET /api/v1/portfolios/{portfolio\_id}/recommendations

- Get current AI recommendations

- Returns: Structured recommendations with priorities

**5.4 Alert Endpoints**

GET /api/v1/alerts/rules

- Get user's alert rules

- Returns: List of configured alert rules

POST /api/v1/alerts/rules

- Create new alert rule

- Body: {rule\_type, conditions, notification\_channels}

- Returns: Created rule object

PUT /api/v1/alerts/rules/{rule\_id}

- Update alert rule

- Body: Updated rule configuration

- Returns: Updated rule object

GET /api/v1/alerts/history

- Get alert history

- Query params: ?limit=50&offset=0

- Returns: Paginated alert history

**5.5 Market Data Endpoints**

GET /api/v1/market/prices

- Get current prices for specified assets

- Query params: ?symbols=BTC,ETH,ADA

- Returns: Current prices with 24h change

GET /api/v1/market/assets/search

- Search for crypto assets

- Query params: ?q=bitcoin

- Returns: List of matching assets

GET /api/v1/market/assets/{asset\_id}/history

- Get price history for asset

- Query params: ?period=7d|30d|90d|1y

- Returns: Time series price data

**6. Frontend Component Hierarchy**

**6.1 Page Components**

src/

├── app/

│ ├── dashboard/

│ │ └── page.tsx # Portfolio overview dashboard

│ ├── risk-analysis/

│ │ └── page.tsx # Detailed risk metrics

│ ├── insights/

│ │ └── page.tsx # AI insights display

│ ├── alerts/

│ │ └── page.tsx # Alert management

│ └── layout.tsx # Root layout with navigation

├── components/

│ ├── dashboard/

│ │ ├── PortfolioOverview.tsx

│ │ ├── AssetAllocation.tsx

│ │ └── QuickStats.tsx

│ ├── charts/

│ │ ├── RiskHeatmap.tsx

│ │ ├── AllocationChart.tsx

│ │ ├── VolatilityChart.tsx

│ │ └── CorrelationMatrix.tsx

│ ├── portfolio/

│ │ ├── HoldingsList.tsx

│ │ ├── AddHolding.tsx

│ │ └── PortfolioSettings.tsx

│ ├── insights/

│ │ ├── WeeklyInsights.tsx

│ │ ├── RecommendationCard.tsx

│ │ └── InsightHistory.tsx

│ ├── alerts/

│ │ ├── AlertRuleForm.tsx

│ │ ├── AlertHistory.tsx

│ │ └── NotificationSettings.tsx

│ └── common/

│ ├── Header.tsx

│ ├── Sidebar.tsx

│ ├── LoadingSpinner.tsx

│ └── ErrorBoundary.tsx

└── hooks/

├── usePortfolio.ts

├── useRiskMetrics.ts

├── useRealTimePrice.ts

└── useAlerts.ts

**6.2 State Management Structure**

// Global State (React Query + Context)

interface AppState {

user: User | null;

selectedPortfolio: Portfolio | null;

realTimePrices: Record<string, number>;

alerts: Alert[];

}

// Portfolio State

interface Portfolio {

id: string;

name: string;

holdings: Holding[];

totalValue: number;

dayChange: number;

riskScore: number;

}

// Risk Metrics State

interface RiskMetrics {

volatility30d: number;

sharpeRatio: number;

maxDrawdown: number;

concentrationRisk: number;

var95: number;

}

**7. Development Timeline & Milestones**

**7.1 Phase 1: Foundation (Days 1-2)**

**Milestone 1.1: Project Setup**

* ✅ Repository structure created
* ✅ Development environment configured
* ✅ Database schema implemented
* ✅ Basic FastAPI server running
* ✅ Next.js frontend initialized

**Milestone 1.2: Data Integration**

* ✅ CoinGecko API integration working
* ✅ Price data fetching and storage
* ✅ Basic CRUD operations for portfolios
* ✅ Database seeded with sample data

**7.2 Phase 2: Core Features (Days 3-4)**

**Milestone 2.1: Portfolio Management**

* ✅ Portfolio creation and editing
* ✅ Holdings management (add/edit/remove)
* ✅ Real-time portfolio valuation
* ✅ Basic portfolio dashboard

**Milestone 2.2: Risk Engine**

* ✅ Risk calculation algorithms implemented
* ✅ Volatility and Sharpe ratio calculations
* ✅ Risk metrics API endpoints
* ✅ Risk heatmap visualization

**7.3 Phase 3: AI Integration (Days 4-5)**

**Milestone 3.1: LangChain Setup**

* ✅ LangChain integration configured
* ✅ GPT-4 prompt engineering completed
* ✅ AI insight generation working
* ✅ Structured output parsing

**Milestone 3.2: Frontend Polish**

* ✅ Interactive charts implemented
* ✅ Real-time data updates
* ✅ Responsive design completed
* ✅ Error handling and loading states

**7.4 Phase 4: Advanced Features (Days 5-6)**

**Milestone 4.1: Alert System**

* ✅ Alert rule configuration
* ✅ Email notification system
* ✅ Telegram bot integration
* ✅ Alert history tracking

**Milestone 4.2: Performance & Polish**

* ✅ Performance optimization
* ✅ Caching implementation
* ✅ UI/UX improvements
* ✅ Mobile responsiveness

**7.5 Phase 5: Deployment (Days 6-7)**

**Milestone 5.1: Production Setup**

* ✅ Frontend deployed to Vercel
* ✅ Backend deployed to Railway/Render
* ✅ Database hosted (Supabase/PlanetScale)
* ✅ Domain and SSL configured

**Milestone 5.2: Documentation & Demo**

* ✅ Complete documentation written
* ✅ Demo video recorded
* ✅ Portfolio presentation prepared
* ✅ Performance monitoring setup

**8. Technical Specifications**

**8.1 Performance Requirements**

* **Page Load Time**: < 2 seconds for initial load
* **Real-time Updates**: < 500ms latency for price updates
* **API Response Time**: < 200ms for portfolio operations
* **Database Queries**: < 100ms for risk calculations
* **Concurrent Users**: Support 100+ concurrent users

**8.2 Security Requirements**

* JWT-based authentication
* Input validation and sanitization
* Rate limiting on API endpoints
* Secure password hashing (bcrypt)
* CORS configuration
* Environment variable management

**8.3 Scalability Considerations**

* Horizontal scaling for FastAPI backend
* Database connection pooling
* Redis caching for frequent queries
* CDN for static assets
* Background job processing for heavy calculations

**9. Success Metrics**

**9.1 Technical Metrics**

* ✅ 100% uptime during demo period
* ✅ < 2s average page load time
* ✅ All API endpoints responding < 200ms
* ✅ Real-time updates working smoothly
* ✅ Mobile-responsive design

**9.2 Feature Completion**

* ✅ Portfolio management fully functional
* ✅ Risk heatmap displaying accurate data
* ✅ AI insights generating meaningful recommendations
* ✅ Alert system sending notifications
* ✅ Professional documentation and demo

**9.3 Portfolio Readiness**

* ✅ Live deployed application
* ✅ Professional GitHub repository
* ✅ Comprehensive documentation
* ✅ Demo video showcasing features
* ✅ Technical blog post written

**10. Risk Assessment & Mitigation**

**10.1 Technical Risks**

**Risk**: API rate limiting from external services **Mitigation**: Implement caching, multiple API sources, graceful degradation

**Risk**: Real-time data synchronization issues **Mitigation**: WebSocket fallback, local state management, error boundaries

**Risk**: AI API costs exceeding budget **Mitigation**: Response caching, request batching, usage monitoring

**10.2 Timeline Risks**

**Risk**: Complex risk calculations taking longer than expected **Mitigation**: Start with basic calculations, iterate and improve

**Risk**: AI integration complexity **Mitigation**: Begin with simple prompts, expand gradually

**Risk**: Deployment issues **Mitigation**: Deploy early and often, have backup hosting options

This comprehensive SRS provides the foundation for building a production-ready crypto risk dashboard. Each component is designed to work together seamlessly while maintaining modularity for easier development and testing.