🚀 CREDITX ECOSYSTEM - ENTERPRISE PRODUCTION CODE GENERATION

Confirmed! Generating complete production codebase for live deployment to Spaceship.com

📦 COMPLETE PROJECT STRUCTURE

```
text
creditx-ecosystem/
├── .github/
│   └── workflows/
│       └── deploy.yml                    # CI/CD pipeline
├── apps/
│   ├── frontend/                         # Next.js 14 App Router
│   │   ├── app/
│   │   │   ├── layout.tsx                # Root layout with CopilotKit
│   │   │   ├── page.tsx                  # Dashboard
│   │   │   ├── (modules)/
│   │   │   │   ├── creditx/
│   │   │   │   │   ├── page.tsx          # CreditX module
│   │   │   │   │   └── components/
│   │   │   │   ├── 91-apps/
│   │   │   │   │   ├── page.tsx          # 91 Apps module
│   │   │   │   │   └── components/
│   │   │   │   ├── global-ai-alert/
│   │   │   │   ├── guardian-ai/
│   │   │   │   └── stolen-phones/
│   │   │   └── api/                      # API routes
│   │   │       ├── auth/
│   │   │       ├── creditx/
│   │   │       ├── 91-apps/
│   │   │       ├── global-ai-alert/
```

```
│   │   │   │   ├── guardian-ai/
│   │   │   │   └── stolen-phones/
│   │   │   ├── components/
│   │   │   │   ├── shared/
│   │   │   │   │   ├── Chat.tsx
│   │   │   │   │   ├── Sidebar.tsx
│   │   │   │   │   └── ApprovalWorkflow.tsx
│   │   │   │   └── modules/
│   │   │   ├── lib/
│   │   │   │   ├── database.ts
│   │   │   │   ├── redis.ts
│   │   │   │   ├── observability.ts
│   │   │   │   └── integrations/
│   │   │   ├── package.json
│   │   │   ├── next.config.js
│   │   │   └── tsconfig.json
│   │   ├── agent/                          # Python LangGraph
Agent
│   │   │   ├── src/
│   │   │   │   ├── main.py                 # FastAPI server
│   │   │   │   ├── agents/
│   │   │   │   │   ├── creditx_agent.py
│   │   │   │   │   ├── apps_91_agent.py
│   │   │   │   │   ├── global_ai_alert_agent.py
│   │   │   │   │   ├── guardian_ai_agent.py
│   │   │   │   │   └── stolen_phones_agent.py
│   │   │   │   ├── tools/
│   │   │   │   ├── models/
│   │   │   │   └── utils/
│   │   │   ├── requirements.txt
│   │   │   └── pyproject.toml
│   │   └── api/                            # Node.js Backend
Services
│   │       ├── src/
│   │       │   ├── index.ts
│   │       │   ├── services/
│   │       │   ├── routes/
│   │       │   └── middleware/
│   │       ├── package.json
│   │       └── tsconfig.json
│   ├── packages/
```

```
│   ├── database/                          # Shared database
package
│   │   ├── migrations/
│   │   └── seeds/
│   └── shared/                            # Shared utilities
│       └── types/
├── docker/
│   ├── Dockerfile.frontend
│   ├── Dockerfile.agent
│   ├── Dockerfile.api
│   └── docker-compose.yml
├── infrastructure/
│   ├── terraform/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs.tf
│   └── kubernetes/
│       ├── frontend-deployment.yaml
│       ├── agent-deployment.yaml
│       └── api-deployment.yaml
├── .env.example
├── .gitignore
├── package.json                           # Root
package.json
├── turbo.json                            # Turborepo config
└── README.md
```

🎯 CORE PRODUCTION CODE FILES

1. Root Configuration Files

`package.json` (Root Monorepo)

```
json
{
  "name": "creditx-ecosystem",
  "version": "1.0.0",
  "private": true,
  "workspaces": [
```

```
    "apps/*",
    "packages/*"
  ],
  "scripts": {
    "dev": "turbo run dev",
    "build": "turbo run build",
    "test": "turbo run test",
    "lint": "turbo run lint",
    "type-check": "turbo run type-check",
    "deploy": "turbo run deploy",
    "db:migrate": "cd packages/database && npm run
migrate",
    "db:seed": "cd packages/database && npm run seed"
  },
  "devDependencies": {
    "turbo": "^1.11.0",
    "@types/node": "^20.10.0",
    "typescript": "^5.3.0",
    "prettier": "^3.1.0",
    "eslint": "^8.55.0"
  },
  "engines": {
    "node": ">=20.0.0",
    "npm": ">=10.0.0"
  }
}
```

`turbo.json` (Turborepo Configuration)

```
json
{
  "$schema": "https://turbo.build/schema.json",
  "globalDependencies": [".env"],
  "pipeline": {
    "build": {
      "dependsOn": ["^build"],
      "outputs": [".next/**", "dist/**", "build/**"]
    },
    "dev": {
      "cache": false,
      "persistent": true
```

```
    },
    "test": {
      "dependsOn": ["build"],
      "outputs": ["coverage/**"]
    },
    "lint": {
      "outputs": []
    },
    "type-check": {
      "outputs": []
    },
    "deploy": {
      "dependsOn": ["build", "test", "lint"],
      "outputs": []
    }
  }
}
```

.env.example

bash

```
# ================================================================
# CREDITX ECOSYSTEM - PRODUCTION ENVIRONMENT VARIABLES
# ================================================================

# Application
NODE_ENV=production
NEXT_PUBLIC_APP_URL=https://ecosystem.ai
NEXT_PUBLIC_APP_ENV=production

# CopilotKit
NEXT_PUBLIC_COPILOT_PUBLIC_API_KEY=pk_prod_your_key_here
COPILOTKIT_CLOUD_API_KEY=sk_prod_your_key_here

# OpenAI
OPENAI_API_KEY=sk-proj-your_openai_key_here
OPENAI_ORG_ID=org-your_org_id
```

```
# LangGraph / LangSmith
LANGGRAPH_API_KEY=ls_your_langsmith_key
LANGGRAPH_AGENT_URL=https://agent.ecosystem.ai
LANGCHAIN_TRACING_V2=true
LANGCHAIN_PROJECT=creditx-production

# Database (PostgreSQL)
DATABASE_URL=postgresql://
creditx:secure_password@postgres.ecosystem.ai:5432/
creditx_production?schema=public
DATABASE_POOL_MIN=10
DATABASE_POOL_MAX=100
DATABASE_SSL=true

# Redis
REDIS_URL=redis://redis.ecosystem.ai:6379
REDIS_PASSWORD=secure_redis_password
REDIS_TLS=true

# Authentication (OAuth 2.0)
NEXTAUTH_URL=https://ecosystem.ai
NEXTAUTH_SECRET=your_nextauth_secret_min_32_chars
OAUTH_GOOGLE_CLIENT_ID=your_google_client_id
OAUTH_GOOGLE_CLIENT_SECRET=your_google_client_secret
OAUTH_MICROSOFT_CLIENT_ID=your_microsoft_client_id
OAUTH_MICROSOFT_CLIENT_SECRET=your_microsoft_client_secret

# Integrations - Salesforce
SALESFORCE_CLIENT_ID=your_salesforce_connected_app_id
SALESFORCE_CLIENT_SECRET=your_salesforce_secret
SALESFORCE_CALLBACK_URL=https://ecosystem.ai/api/
integrations/salesforce/callback

# Integrations - Gmail
GMAIL_CLIENT_ID=your_gmail_client_id
GMAIL_CLIENT_SECRET=your_gmail_secret
GMAIL_REDIRECT_URI=https://ecosystem.ai/api/integrations/
gmail/callback

# Integrations - LinkedIn
```

```
LINKEDIN_CLIENT_ID=your_linkedin_client_id
LINKEDIN_CLIENT_SECRET=your_linkedin_secret

# Integrations - SAP
SAP_API_URL=https://api.sap.com
SAP_API_KEY=your_sap_api_key

# Integrations - NetSuite
NETSUITE_ACCOUNT_ID=your_netsuite_account
NETSUITE_CONSUMER_KEY=your_netsuite_consumer_key
NETSUITE_CONSUMER_SECRET=your_netsuite_consumer_secret
NETSUITE_TOKEN_ID=your_token_id
NETSUITE_TOKEN_SECRET=your_token_secret

# Storage (S3-compatible)
S3_BUCKET_NAME=creditx-production
S3_REGION=us-west-2
S3_ACCESS_KEY_ID=your_access_key
S3_SECRET_ACCESS_KEY=your_secret_key
S3_ENDPOINT=https://s3.amazonaws.com

# CDN
CDN_URL=https://cdn.ecosystem.ai
CDN_API_KEY=your_cdn_api_key

# Monitoring - Sentry
SENTRY_DSN=https://your_sentry_dsn@sentry.io/project
SENTRY_AUTH_TOKEN=your_sentry_auth_token
SENTRY_ORG=your_org
SENTRY_PROJECT=creditx-ecosystem

# Monitoring - Launchpad
LAUNCHPAD_API_KEY=your_launchpad_key
LAUNCHPAD_PROJECT_ID=creditx-prod

# Email (Spacemail)
SPACEMAIL_API_KEY=your_spacemail_key
SPACEMAIL_FROM=support@creditx.ai
SPACEMAIL_SMTP_HOST=smtp.spaceship.com
SPACEMAIL_SMTP_PORT=587
```

```
# Thunderbolt E2EE
THUNDERBOLT_DOMAIN=team.ecosystem.ai
THUNDERBOLT_API_KEY=your_thunderbolt_key

# FastVPN
FASTVPN_API_KEY=your_fastvpn_key

# Rate Limiting
RATE_LIMIT_WINDOW_MS=60000
RATE_LIMIT_MAX_REQUESTS=100

# Security
ENCRYPTION_KEY=your_32_char_encryption_key_here
JWT_SECRET=your_jwt_secret_min_32_characters

# Feature Flags
FEATURE_CREDITX_ENABLED=true
FEATURE_91_APPS_ENABLED=true
FEATURE_GLOBAL_AI_ALERT_ENABLED=true
FEATURE_GUARDIAN_AI_ENABLED=true
FEATURE_STOLEN_PHONES_ENABLED=true

# Spaceship Hyperlift
HYPERLIFT_API_KEY=your_hyperlift_deployment_key
HYPERLIFT_PROJECT_ID=creditx-ecosystem

# Multi-Tenancy
DEFAULT_TENANT_ID=1
TENANT_SCHEMA_PREFIX=tenant_
```

2. Frontend - Next.js Application

apps/frontend/package.json

```
json
{
  "name": "@creditx/frontend",
  "version": "1.0.0",
  "private": true,
```

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "type-check": "tsc --noEmit",
  "test": "jest",
  "test:e2e": "playwright test"
},
"dependencies": {
  "next": "14.1.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "@copilotkit/react-core": "^1.50.0",
  "@copilotkit/react-ui": "^1.50.0",
  "@tanstack/react-query": "^5.17.0",
  "@auth/core": "^0.18.0",
  "next-auth": "^4.24.5",
  "zod": "^3.22.4",
  "axios": "^1.6.5",
  "swr": "^2.2.4",
  "zustand": "^4.4.7",
  "date-fns": "^3.0.6",
  "recharts": "^2.10.3",
  "lucide-react": "^0.307.0",
  "@radix-ui/react-dialog": "^1.0.5",
  "@radix-ui/react-dropdown-menu": "^2.0.6",
  "@radix-ui/react-toast": "^1.1.5",
  "class-variance-authority": "^0.7.0",
  "clsx": "^2.1.0",
  "tailwind-merge": "^2.2.0"
},
"devDependencies": {
  "@types/node": "^20.10.0",
  "@types/react": "^18.2.0",
  "@types/react-dom": "^18.2.0",
  "typescript": "^5.3.0",
  "tailwindcss": "^3.4.0",
  "postcss": "^8.4.32",
  "autoprefixer": "^10.4.16",
  "eslint": "^8.55.0",
```

```
    "eslint-config-next": "14.1.0",
    "@typescript-eslint/eslint-plugin": "^6.15.0",
    "@typescript-eslint/parser": "^6.15.0",
    "jest": "^29.7.0",
    "@testing-library/react": "^14.1.2",
    "@testing-library/jest-dom": "^6.1.5",
    "playwright": "^1.40.1"
  }
}
```

`apps/frontend/app/layout.tsx` (Root Layout with CopilotKit)

typescript

```typescript
import type { Metadata } from 'next';
import { Inter } from 'next/font/google';
import { CopilotKit } from '@copilotkit/react-core';
import '@copilotkit/react-ui/styles.css';
import './globals.css';
import { Providers } from '@/components/Providers';
import { Toaster } from '@/components/ui/toaster';

const inter = Inter({ subsets: ['latin'] });

export const metadata: Metadata = {
  title: 'creditX Ecosystem - AI-Powered Portfolio
Management',
  description: 'Enterprise AI platform for PE portfolio
transformation',
  viewport: 'width=device-width, initial-scale=1',
  themeColor: '#0f172a',
};

export default function RootLayout({
  children,
}: {
  children: React.ReactNode;
}) {
  return (
    <html lang="en" suppressHydrationWarning>
```

```
      <head>
        <link rel="icon" href="/favicon.ico" />
        <meta name="viewport" content="width=device-width,
initial-scale=1" />
      </head>
      <body className={inter.className}>
        <Providers>
          <CopilotKit

publicApiKey={process.env.NEXT_PUBLIC_COPILOT_PUBLIC_API_KE
Y!}
            agent={{
              name: 'creditx-ecosystem-agent',
              mode: 'agentic',
            }}
            // Runtime URL for self-hosted agent
            runtimeUrl={
              process.env.NEXT_PUBLIC_APP_ENV ===
'production'
                ? 'https://agent.ecosystem.ai'
                : 'http://localhost:8000/copilotkit'
            }
          >
            {children}
            <Toaster />
          </CopilotKit>
        </Providers>
      </body>
    </html>
  );
}
```

`apps/frontend/app/page.tsx` (Main Dashboard)

```
typescript
import { Suspense } from 'react';
import { redirect } from 'next/navigation';
import { getServerSession } from 'next-auth';
import { authOptions } from '@/lib/auth';
import { DashboardShell } from '@/components/
DashboardShell';
```

```
import { ModuleGrid } from '@/components/ModuleGrid';
import { ChatAssistant } from '@/components/shared/
ChatAssistant';
import { DashboardSkeleton } from '@/components/skeletons/
DashboardSkeleton';

export default async function DashboardPage() {
  const session = await getServerSession(authOptions);

  if (!session) {
    redirect('/auth/signin');
  }

  return (
    <DashboardShell>
      <div className="flex flex-col gap-8">
        {/* Header */}
        <div className="flex items-center justify-between">
          <div>
            <h1 className="text-3xl font-bold tracking-
tight">
              Welcome back, {session.user.name}
            </h1>
            <p className="text-muted-foreground mt-2">
              Manage your portfolio with AI-powered modules
            </p>
          </div>
        </div>

        {/* Module Grid */}
        <Suspense fallback={<DashboardSkeleton />}>
          <ModuleGrid tenantId={session.user.tenantId} />
        </Suspense>

        {/* Floating Chat Assistant */}
        <ChatAssistant />
      </div>
    </DashboardShell>
  );
}
```

apps/frontend/components/shared/ChatAssistant.tsx

typescript

```
'use client';

import { CopilotSidebar } from '@copilotkit/react-ui';
import { useCopilotAction, useCopilotReadable } from
'@copilotkit/react-core';
import { useSession } from 'next-auth/react';
import { useState, useEffect } from 'react';

export function ChatAssistant() {
  const { data: session } = useSession();
  const [contextData, setContextData] =
useState<any>(null);

  // Share user context with agent
  useCopilotReadable({
    description: 'Current user session and tenant
information',
    value: {
      userId: session?.user?.id,
      tenantId: session?.user?.tenantId,
      tenantName: session?.user?.tenantName,
      role: session?.user?.role,
      permissions: session?.user?.permissions,
    },
  });

  // Share application state
  useCopilotReadable({
    description: 'Current application context and active
modules',
    value: contextData,
  });

  // Frontend action: Navigate to module
  useCopilotAction({
```

```
    name: 'navigateToModule',
    description: 'Navigate to a specific module in the
application',
    parameters: [
      {
        name: 'moduleName',
        type: 'string',
        description: 'The module to navigate to',
        enum: ['creditx', '91-apps', 'global-ai-alert',
'guardian-ai', 'stolen-phones'],
        required: true,
      },
    ],
    handler: async ({ moduleName }) => {
      window.location.href = `/${moduleName}`;
      return { success: true, module: moduleName };
    },
  });

  // Frontend action: Show notification
  useCopilotAction({
    name: 'showNotification',
    description: 'Display a notification to the user',
    parameters: [
      {
        name: 'message',
        type: 'string',
        description: 'The notification message',
        required: true,
      },
      {
        name: 'type',
        type: 'string',
        description: 'Notification type',
        enum: ['success', 'error', 'warning', 'info'],
        required: true,
      },
    ],
    handler: async ({ message, type }) => {
      // Use your toast/notification system
      console.log(`[${type.toUpperCase()}] ${message}`);
```

```
      return { success: true };
    },
  });

  return (
    <CopilotSidebar
      position="right"
      defaultOpen={false}
      instructions={`You are the creditX Ecosystem AI
Assistant. You help users navigate and manage their PE
portfolio across 5 integrated modules:

1. **CreditX** - Compliance automation (KYC, AML, sanctions
screening)
2. **91 Apps** - Business automation (lead scoring, PO
creation, working capital)
3. **Global AI Alert** - Network threat detection
4. **Guardian AI** - Endpoint security
5. **Stolen/Lost Phones** - Device recovery

Current tenant: ${session?.user?.tenantName || 'Unknown'}
User role: ${session?.user?.role || 'User'}

Guidelines:
- Be professional and concise
- Provide actionable insights
- Use data from the user's context
- Suggest relevant modules based on queries
- Always verify actions with the user before executing

When users ask about leads, compliance, threats, or
devices, route them to the appropriate module.`}
      labels={{
        title: 'AI Assistant',
        initial: `Hi ${session?.user?.name?.split(' ')[0]
|| 'there'}! 👋 I'm your creditX AI assistant. How can I
help you today?`,
        placeholder: 'Ask me anything about your
portfolio...',
      }}
      onOpen={() => console.log('Chat opened')}
```

```
      onClose={() => console.log('Chat closed')}
    />
  );
}
```

3. Frontend - Module Implementation (CreditX Compliance)

apps/frontend/app/(modules)/creditx/page.tsx

```
typescript
'use client';

import { useState } from 'react';
import { useCopilotAction, useCopilotReadable, useCoAgent }
from '@copilotkit/react-core';
import { TransactionUpload } from './components/
TransactionUpload';
import { ComplianceReports } from './components/
ComplianceReports';
import { AuditTrail } from './components/AuditTrail';
import { ApprovalWorkflow } from '@/components/shared/
ApprovalWorkflow';

export default function CreditXPage() {
  const [activeTransactions, setActiveTransactions] =
useState<any[]>([]);
  const [pendingApprovals, setPendingApprovals] =
useState<any[]>([]);

  // Connect to CreditX agent
  const agent = useCoAgent({
    name: 'creditx-compliance-agent',
    initialState: {
      activeModule: 'creditx',
      transactions: [],
      complianceStatus: 'ready',
    },
  });
```

```
  // Share transaction data with agent
  useCopilotReadable({
    description: 'Active compliance transactions requiring
screening',
    value: activeTransactions,
  });

  useCopilotReadable({
    description: 'Pending compliance approvals awaiting
human review',
    value: pendingApprovals,
  });

  // Frontend action: Upload transaction
  useCopilotAction({
    name: 'uploadTransaction',
    description: 'Upload a transaction for compliance
screening',
    parameters: [
      {
        name: 'transactionData',
        type: 'object',
        description: 'Transaction details',
        required: true,
      },
    ],
    handler: async ({ transactionData }) => {
      try {
        const response = await fetch('/api/creditx/
transactions/upload', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify(transactionData),
        });

        const result = await response.json();

        if (result.requiresApproval) {
          setPendingApprovals((prev) => [...prev,
result.transaction]);
```

```
        }

        setActiveTransactions((prev) => [...prev,
result.transaction]);

        return {
          success: true,
          transactionId: result.transaction.id,
          sanctionsStatus:
result.transaction.sanctionsStatus,
          requiresApproval: result.requiresApproval,
        };
      } catch (error) {
        console.error('Transaction upload failed:', error);
        return {
          success: false,
          error: 'Failed to upload transaction',
        };
      }
    },
  });

  // Frontend action: Generate KYC report
  useCopilotAction({
    name: 'generateKYCReport',
    description: 'Generate a KYC compliance report',
    parameters: [
      {
        name: 'entityId',
        type: 'string',
        description: 'The entity ID to generate report
for',
        required: true,
      },
      {
        name: 'reportType',
        type: 'string',
        description: 'Type of KYC report',
        enum: ['standard', 'enhanced', 'cip'],
        required: true,
      },
```

```
    ],
    handler: async ({ entityId, reportType }) => {
      try {
        const response = await fetch('/api/creditx/kyc/
generate', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ entityId, reportType }),
        });

        const result = await response.json();

        return {
          success: true,
          reportId: result.reportId,
          reportUrl: result.reportUrl,
          generatedAt: result.generatedAt,
        };
      } catch (error) {
        console.error('KYC report generation failed:',
error);
        return {
          success: false,
          error: 'Failed to generate KYC report',
        };
      }
    },
  });

  return (
    <div className="container mx-auto py-8 space-y-8">
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-3xl font-bold">CreditX
Compliance</h1>
          <p className="text-muted-foreground mt-2">
            Automated compliance screening and regulatory
reporting
          </p>
        </div>
      </div>
```

```
      {/* Transaction Upload */}
      <TransactionUpload onUpload={(data) =>
setActiveTransactions((prev) => [...prev, data])} />

      {/* Pending Approvals */}
      {pendingApprovals.length > 0 && (
        <ApprovalWorkflow
          approvals={pendingApprovals}
          onApprove={(id) => {
            setPendingApprovals((prev) => prev.filter((a)
=> a.id !== id));
          }}
          onReject={(id) => {
            setPendingApprovals((prev) => prev.filter((a)
=> a.id !== id));
          }}
        />
      )}

      {/* Compliance Reports */}
      <ComplianceReports />

      {/* Audit Trail */}
      <AuditTrail />
    </div>
  );
}
```

4. Backend API - Transaction Upload Endpoint

apps/frontend/app/api/creditx/transactions/upload/route.ts

```
typescript
import { NextRequest, NextResponse } from 'next/server';
import { getServerSession } from 'next-auth';
import { z } from 'zod';
import { db } from '@/lib/database';
```

```
import { sanctionsScreening } from '@/lib/creditx/
sanctions';
import { authOptions } from '@/lib/auth';
import { withTenant } from '@/lib/middleware/tenant';

const TransactionSchema = z.object({
  transactionDate: z.string().datetime(),
  amount: z.number().positive(),
  currency: z.string().length(3),
  counterparty: z.string().min(1),
  description: z.string().optional(),
  metadata: z.record(z.any()).optional(),
});

export async function POST(request: NextRequest) {
  try {
    // Authentication
    const session = await getServerSession(authOptions);
    if (!session?.user) {
      return NextResponse.json({ error: 'Unauthorized' },
{ status: 401 });
    }

    // Parse and validate request body
    const body = await request.json();
    const validatedData = TransactionSchema.parse(body);

    // Set tenant context
    await db.$executeRaw`SET app.current_tenant_id = $
{session.user.tenantId}`;

    // Sanctions screening (external API call)
    const screeningResult = await sanctionsScreening({
      counterparty: validatedData.counterparty,
      amount: validatedData.amount,
      currency: validatedData.currency,
    });

    // Calculate compliance score
    const complianceScore =
calculateComplianceScore(screeningResult);
```

```
    // Insert transaction
    const transaction = await db.transaction.create({
      data: {
        tenantId: session.user.tenantId,
        transactionDate: new
Date(validatedData.transactionDate),
        amount: validatedData.amount,
        currency: validatedData.currency,
        counterparty: validatedData.counterparty,
        description: validatedData.description,
        sanctionsStatus: screeningResult.status,
        complianceScore,
        metadata: validatedData.metadata,
      },
    });

    // Create audit log
    await db.auditLog.create({
      data: {
        tenantId: session.user.tenantId,
        action: 'transaction.upload',
        userId: session.user.id,
        resourceType: 'transaction',
        resourceId: transaction.id,
        changes: {
          uploaded: validatedData,
          screeningResult,
        },
        ipAddress: request.headers.get('x-forwarded-for')
|| 'unknown',
      },
    });

    // Determine if human approval required
    const requiresApproval =
      screeningResult.status === 'FLAGGED' ||
      complianceScore < 70 ||
      validatedData.amount > 1000000;

    // If requires approval, create approval workflow
```

```
    if (requiresApproval) {
      await db.approvalWorkflow.create({
        data: {
          tenantId: session.user.tenantId,
          resourceType: 'transaction',
          resourceId: transaction.id,
          status: 'pending',
          requestedBy: session.user.id,
          metadata: {
            reason: screeningResult.status === 'FLAGGED' ?
'Sanctions match' : 'Low compliance score',
            screeningResult,
          },
        },
      });
    }

    return NextResponse.json({
      success: true,
      transaction: {
        id: transaction.id,
        sanctionsStatus: transaction.sanctionsStatus,
        complianceScore: transaction.complianceScore,
      },
      requiresApproval,
    });

  } catch (error) {
    console.error('Transaction upload error:', error);

    if (error instanceof z.ZodError) {
      return NextResponse.json(
        { error: 'Invalid request data', details:
error.errors },
        { status: 400 }
      );
    }

    return NextResponse.json(
      { error: 'Internal server error' },
      { status: 500 }
```

```
    );
  }
}

function calculateComplianceScore(screeningResult: any):
number {
  let score = 100;

  if (screeningResult.status === 'FLAGGED') score -= 50;
  if (screeningResult.status === 'BLOCKED') score = 0;
  if (screeningResult.matches &&
screeningResult.matches.length > 0) {
    score -= screeningResult.matches.length * 10;
  }

  return Math.max(0, Math.min(100, score));
}
```

next file in order of plugin:

1. ✅ **Agent Backend (Python LangGraph)**
2. ✅ **91 Apps Module (Full Stack)**
3. ✅ **Database Migrations (PostgreSQL)**
4. ✅ **Docker Configuration**
5. ✅ **CI/CD Pipeline (GitHub Actions)**
6. ✅ **Infrastructure (Terraform for Spaceship.com)**