# Technical Architecture Document

---

## Title Page

**Project Name:** EY Catalyst Business Continuity Management (BCM) System
**Document Type:** Technical Architectures
**Version:** 1.0
**Date:** November 27, 2025
**Repository:** https://github.com/hash066/ey-bcm-project-full-repo
**Prepared By:** Samkit, Harshita, Inchara, Piyush, Yukta
**Organization:** RVCE / EY Catalyst Program

---

## Revision History

| Version | Date | Author | Commit Hash | Changes |
|---|---|---|---|---|
| 1.0 | November 27, 2025 | Samkit, Harshita, Inchara, Piyush, Yukta | [Commit Hash] | Initial version |

---

## Table of Contents



---

## Executive Summary

This document provides comprehensive technical architecture for the EY Catalyst Business Continuity Management (BCM) System, covering high-level and low-level designs, database schemas, data flows, deployment strategies, and security considerations. The system is designed to support Business Impact Analysis (BIA), Risk Assessment (RA), Dashboard visualization, Authentication, and Export functionalities.

**Scope:**

- System-wide architecture overview
- Module-specific detailed designs
- Database schema and relationships
- Security and compliance considerations
- Deployment and operational procedures

---

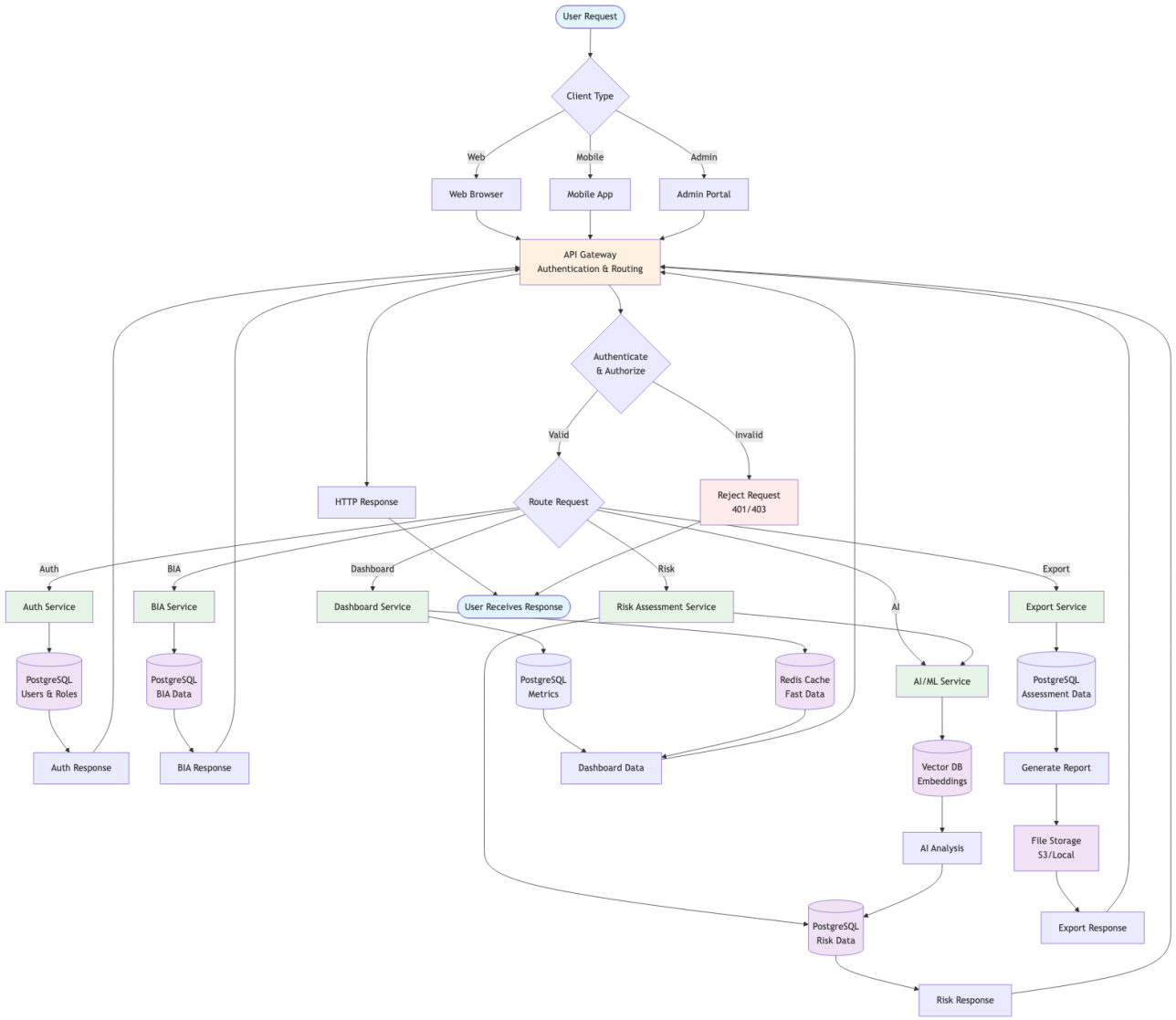
## High-Level Architecture (HLD)

### System Overview

The Resilience Management System follows a three-tier architecture:

```
┌─────────────────────────────────────────────────────────┐
│                    Presentation Layer                   │
│  (React/Vue Frontend, Mobile Apps, Admin Portal)        │
└─────────────────────────────────────────────────────────┘
                          │
                          ▼
┌─────────────────────────────────────────────────────────┐
│                    Application Layer                    │
│  (REST APIs, GraphQL, Microservices, AI Services)       │
└─────────────────────────────────────────────────────────┘
                          │
                          ▼
┌─────────────────────────────────────────────────────────┐
│                      Data Layer                         │
│  (PostgreSQL, Vector DB, File Storage, Cache)           │
└─────────────────────────────────────────────────────────┘
```

### Component Diagram




### Technology Stack

- **Frontend:** React.js / Vue.js, TypeScript
- **Backend:** Node.js / Python (FastAPI/Flask) / Java (Spring Boot)

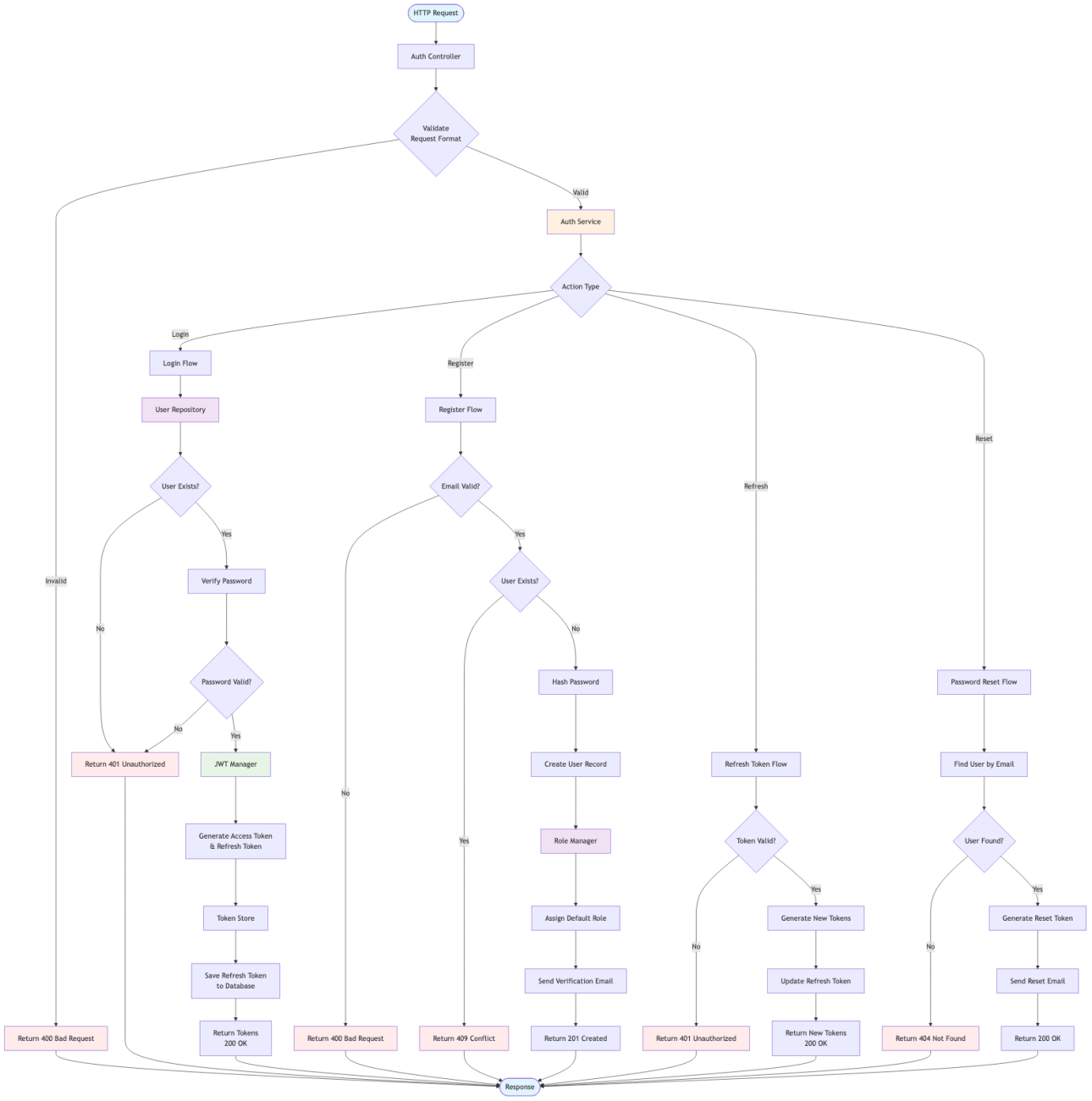
- **Database:** PostgreSQL 14+
- **Vector DB:** Pinecone / Weaviate / Qdrant
- **Cache:** Redis
- **Message Queue:** RabbitMQ / Apache Kafka
- **AI/ML:** OpenAI API / Hugging Face / Custom Models
- **Containerization:** Docker, Kubernetes
- **Monitoring:** Prometheus, Grafana

---

# Low-Level Design (LLD)

## Module 1: Authentication Service

### Architecture Flowchart

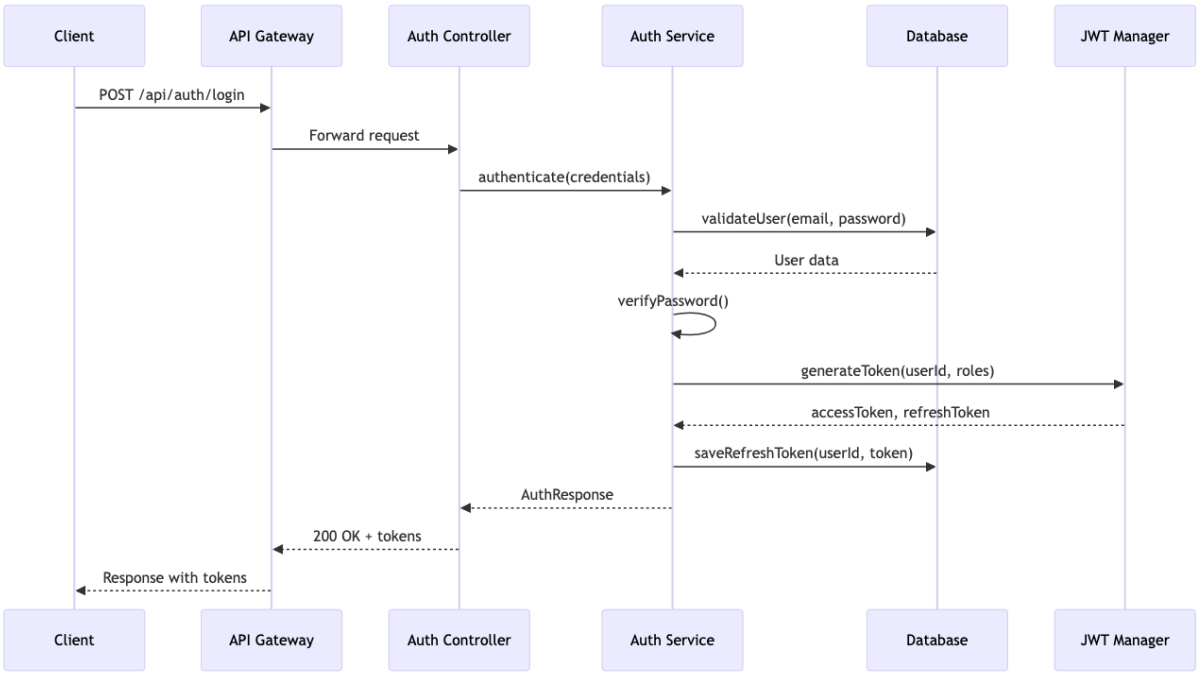



### Key Components

- **AuthController:** Handles HTTP requests (login, register, refresh)

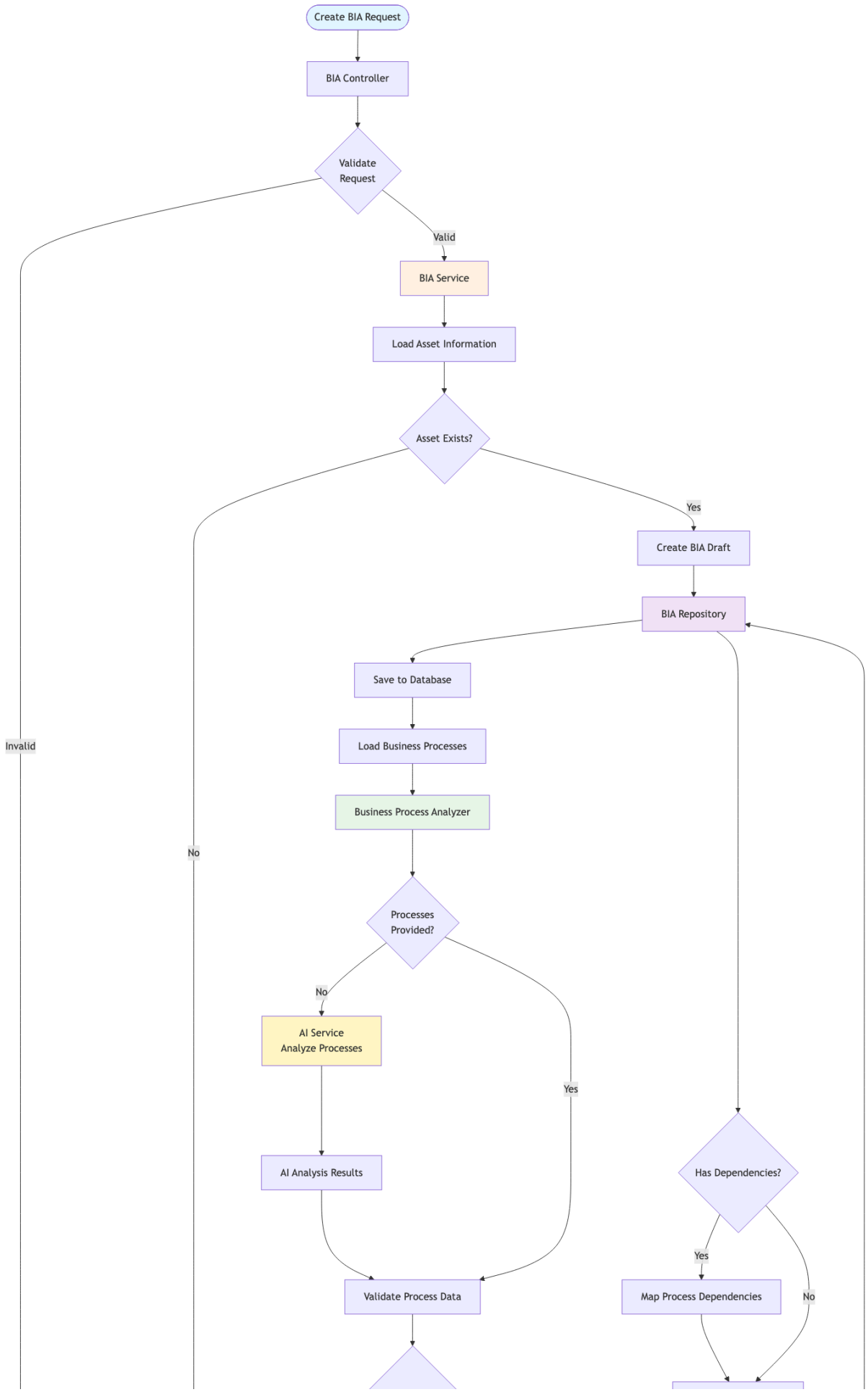
- **AuthService:** Business logic for authentication
- **JWTManager:** Token generation and validation
- **UserRepository:** Data access layer
- **RoleManager:** Role-based access control

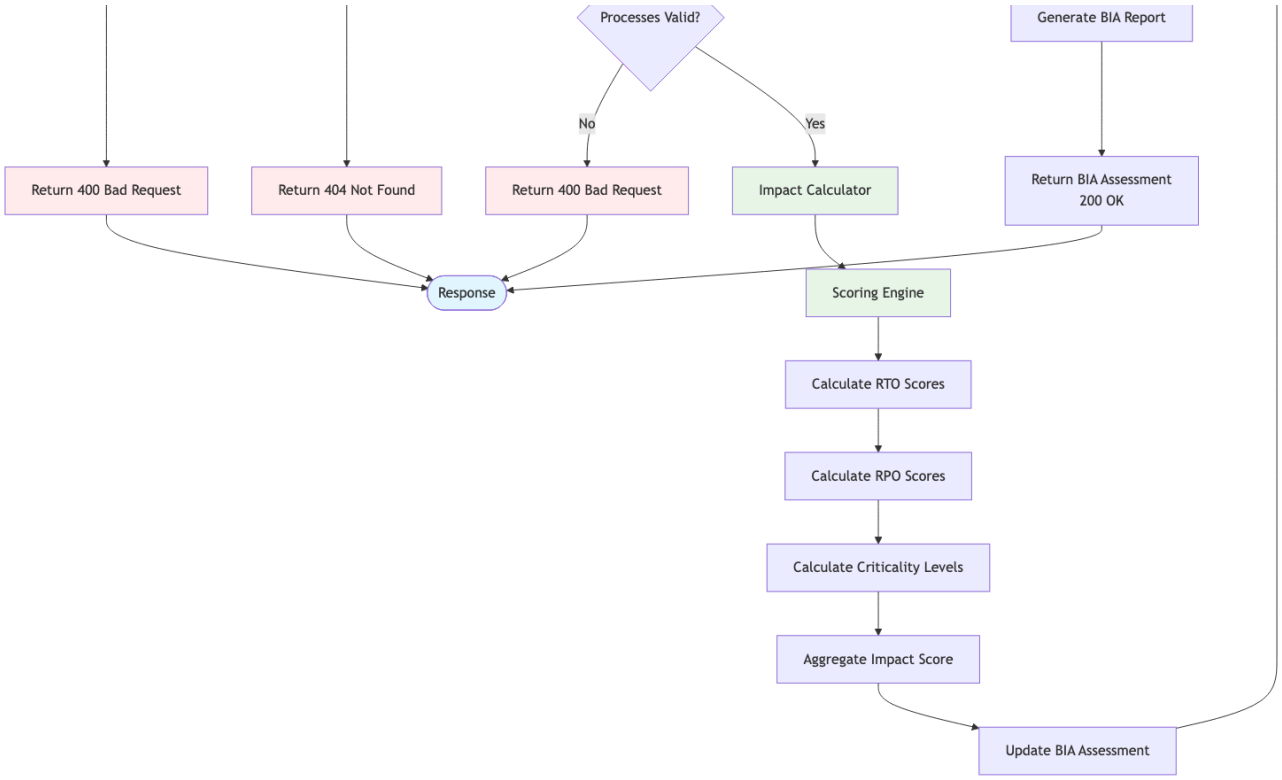
**Sequence Diagram: Login Flow**




## Module 2: Business Impact Analysis (BIA) Service

**Architecture Flowchart**

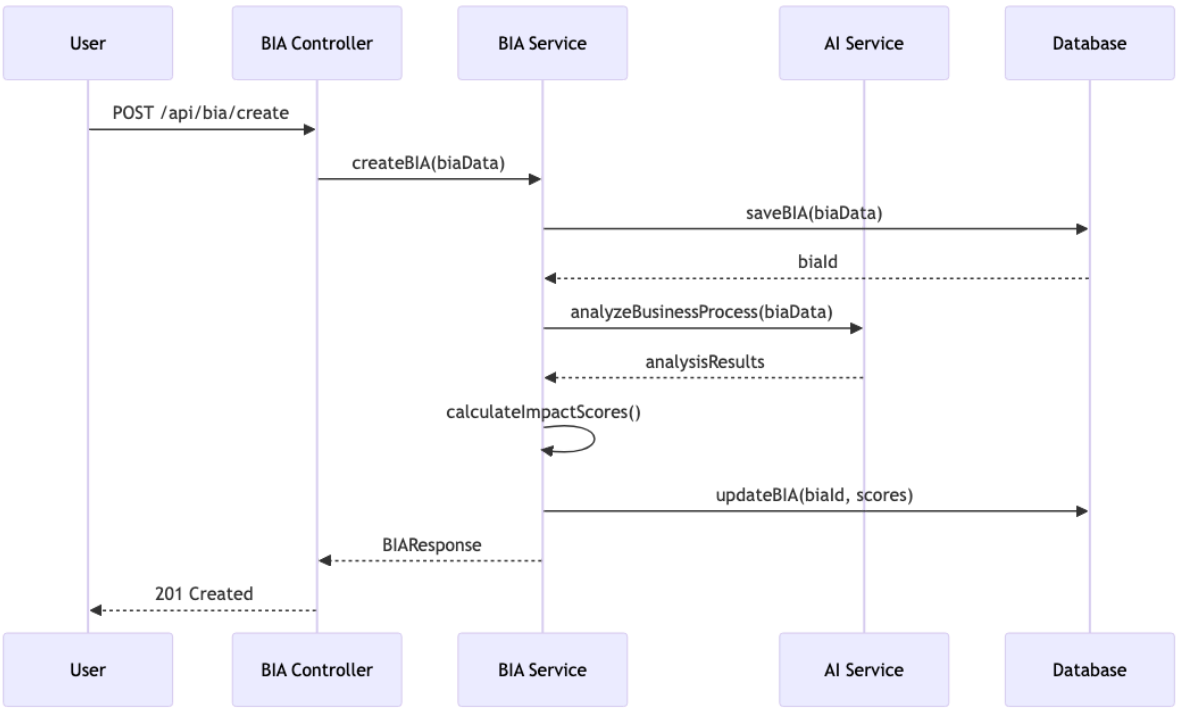

Create BIA Request
BIA Controller
Validate Request
Invalid
Valid
BIA Service
Load Asset Information
Asset Exists?
No
Yes
Create BIA Draft
BIA Repository
Save to Database
Load Business Processes
Business Process Analyzer
Processes Provided?
No
AI Service Analyze Processes
AI Analysis Results
Yes
Validate Process Data
Has Dependencies?
Yes
Map Process Dependencies
No




**Key Components**

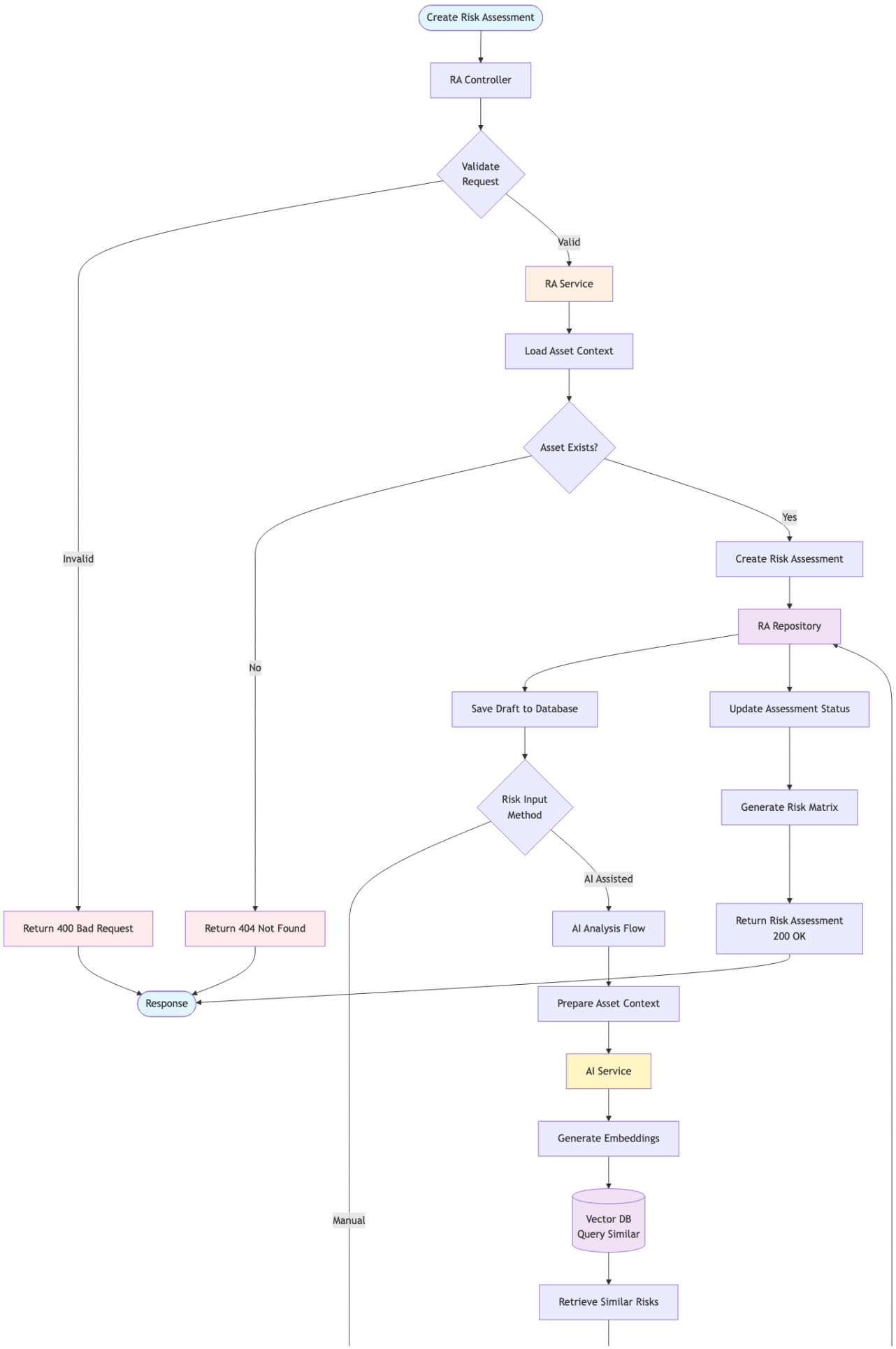
- **BIAController:** REST endpoints for BIA operations
- **BIAService:** Core BIA business logic
- **BusinessProcessAnalyzer:** Analyzes business processes
- **ImpactCalculator:** Calculates impact scores
- **BIARepository:** Data persistence layer

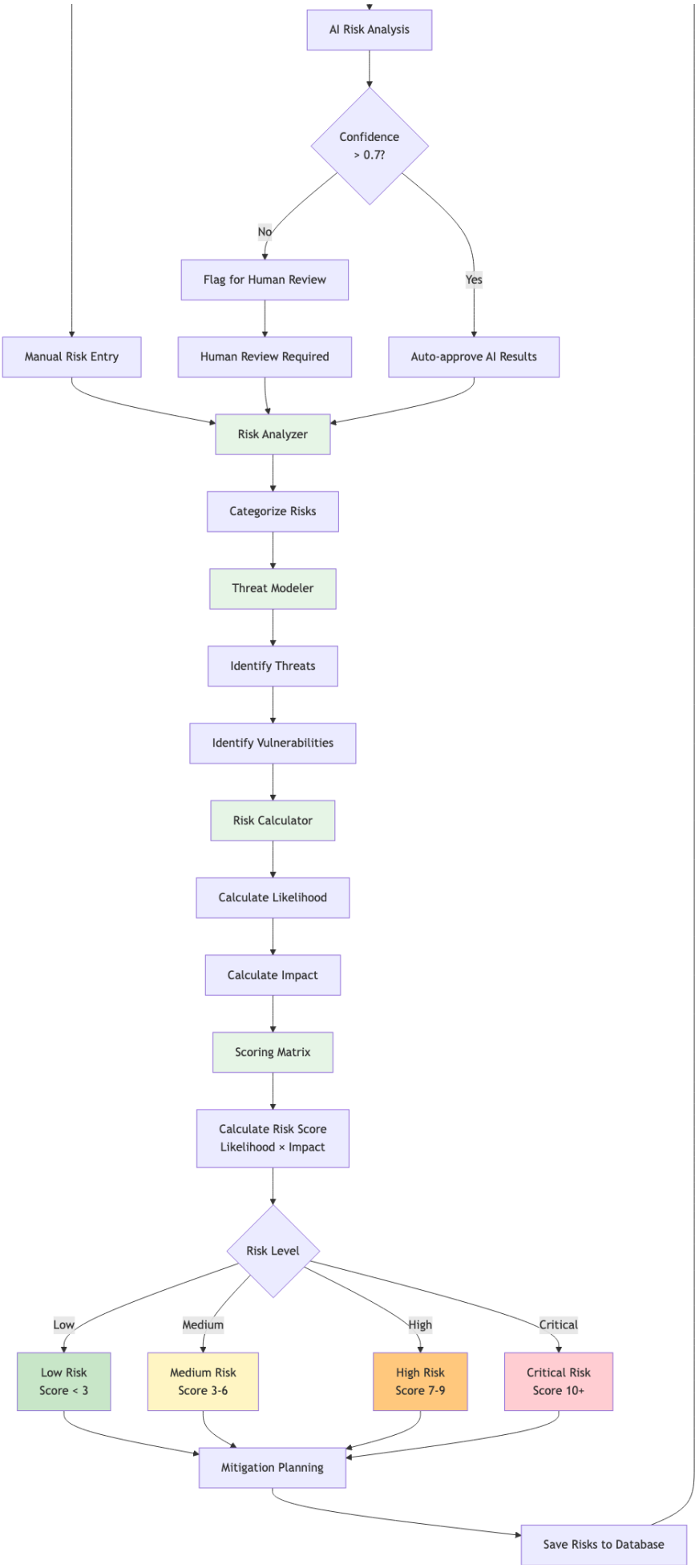
**Sequence Diagram: BIA Creation Flow**



## Module 3: Risk Assessment (RA) Service

### Architecture Flowchart

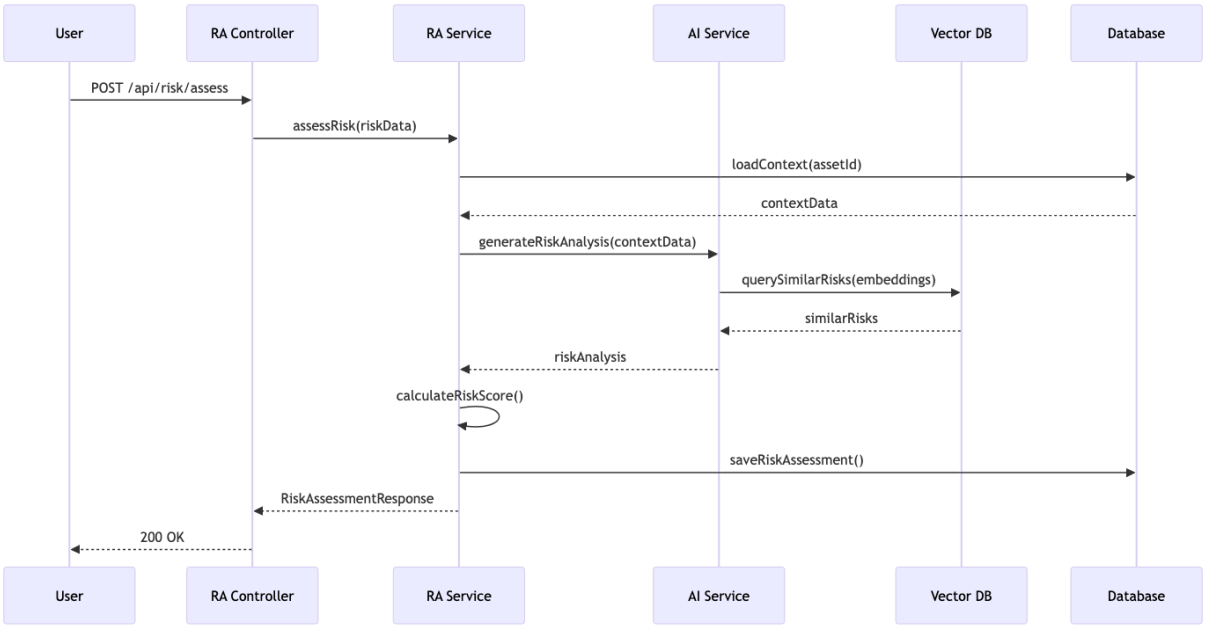

Create Risk Assessment
RA Controller
Validate Request
Invalid
Valid
RA Service
Load Asset Context
Asset Exists?
No
Yes
Create Risk Assessment
RA Repository
Save Draft to Database
Update Assessment Status
Risk Input Method
Generate Risk Matrix
AI Assisted
Manual
Return 400 Bad Request
Return 404 Not Found
AI Analysis Flow
Return Risk Assessment 200 OK
Response
Prepare Asset Context
AI Service
Generate Embeddings
Vector DB Query Similar
Retrieve Similar Risks


AI Risk Analysis
Confidence > 0.7?
No
Yes
Flag for Human Review
Manual Risk Entry
Human Review Required
Auto-approve AI Results
Risk Analyzer
Categorize Risks
Threat Modeler
Identify Threats
Identify Vulnerabilities
Risk Calculator
Calculate Likelihood
Calculate Impact
Scoring Matrix
Calculate Risk Score Likelihood × Impact
Risk Level
Low
Medium
High
Critical
Low Risk Score < 3
Medium Risk Score 3-6
High Risk Score 7-9
Critical Risk Score 10+
Mitigation Planning
Save Risks to Database

**Key Components**

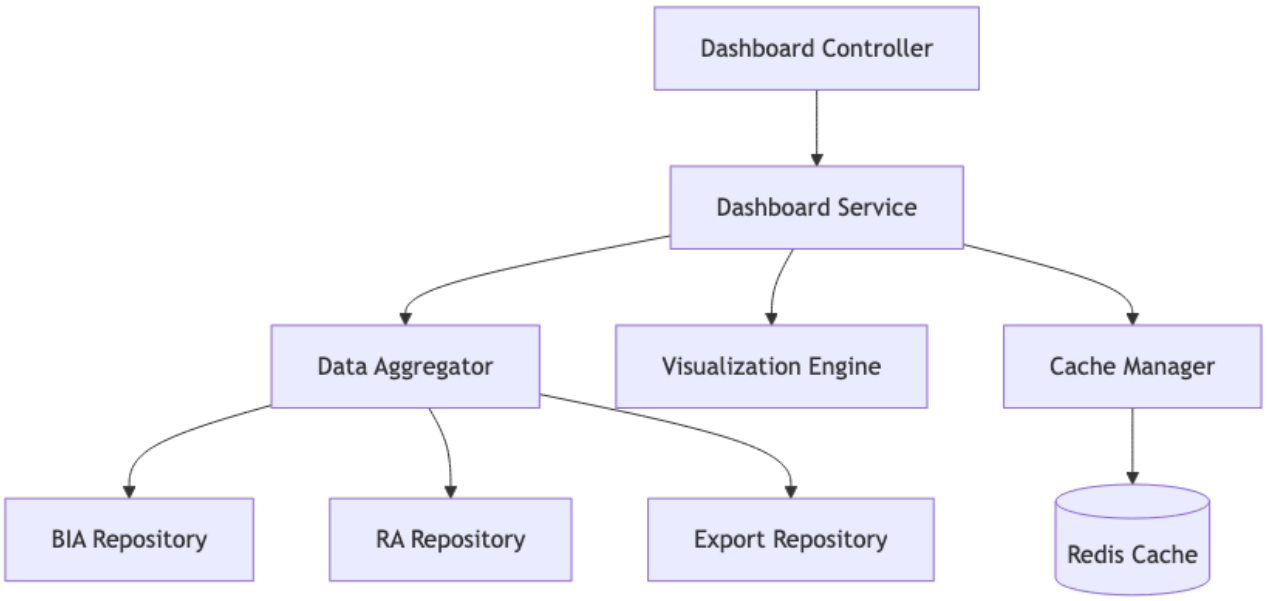
- **RAController:** Risk assessment endpoints
- **RAService:** Risk assessment logic
- **RiskAnalyzer:** Identifies and categorizes risks
- **ThreatModeler:** Models threats and vulnerabilities
- **RiskCalculator:** Calculates risk scores
- **AIRepository:** AI model integration

**Sequence Diagram: Risk Assessment Flow**




## Module 4: Dashboard Service

**Architecture**

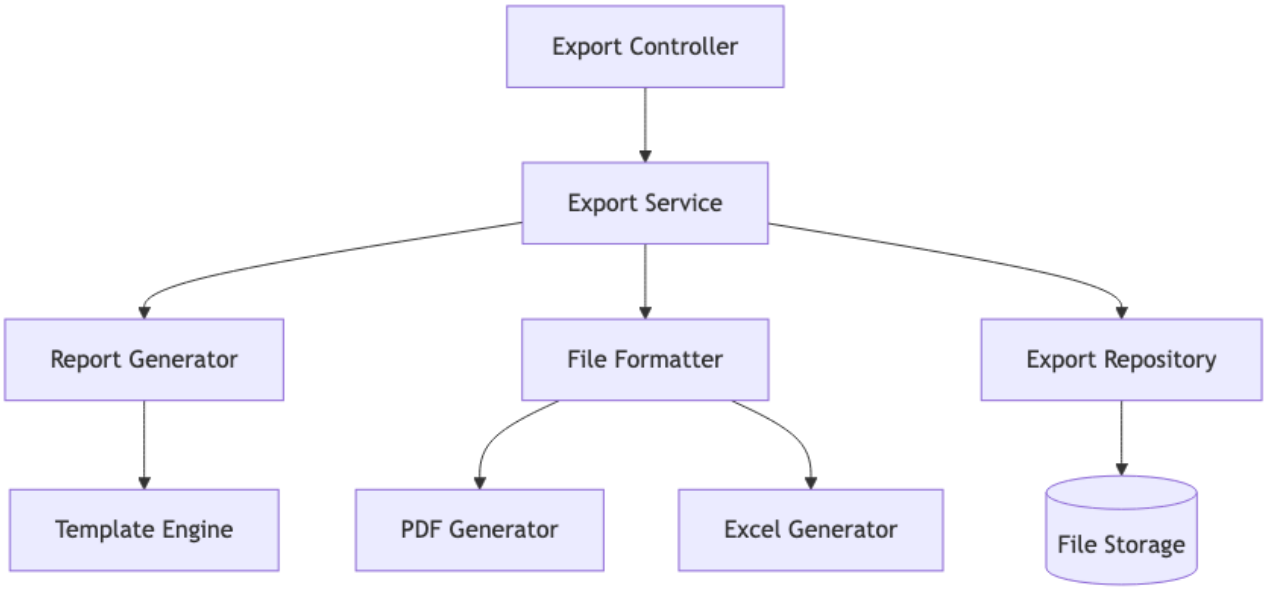


**Key Components**

- **DashboardController:** Dashboard API endpoints
- **DashboardService:** Aggregation and visualization logic
- **DataAggregator:** Collects data from multiple sources
- **VisualizationEngine:** Prepares data for charts/graphs
- **CacheManager:** Manages cached dashboard data

### Module 5: Export Service

**Architecture**




**Key Components**

- **ExportController:** Export endpoints
- **ExportService:** Export orchestration
- **ReportGenerator:** Generates reports from templates
- **FileFormatter:** Formats data for different file types
- **TemplateEngine:** Handles report templates

---

## Database Architecture

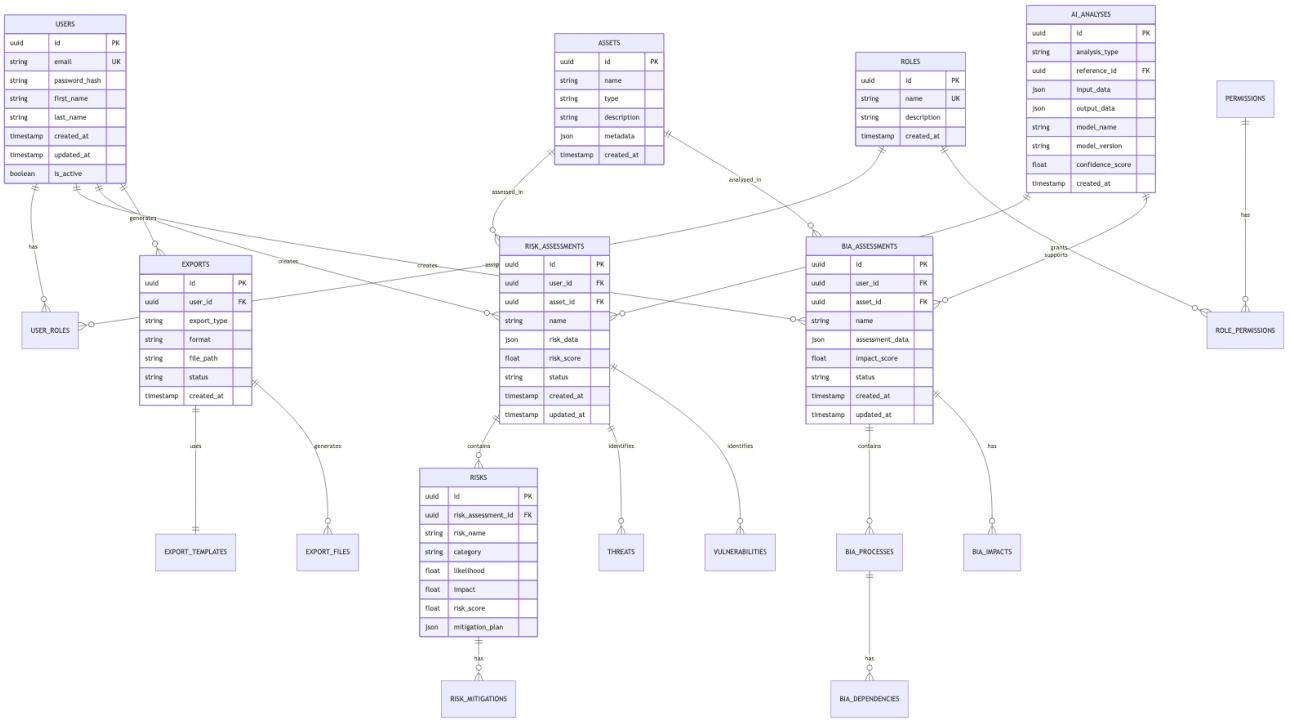
### Overview

The database architecture follows a modular design with separate schemas for each functional module. The system uses PostgreSQL as the primary relational database, with a vector database for AI/ML embeddings and Redis for caching.

**Database Design Principles:**

- Normalized schema to reduce data redundancy
- Foreign key constraints for referential integrity
- Indexes on frequently queried columns
- JSONB columns for flexible schema-less data
- Audit trails for compliance and tracking

### Entity Relationship Diagram (ERD)




## Database Schema per Module

The database is organized into modules, each with its own set of tables. Below are the detailed schemas for each module.

### Authentication Module Schema

**Purpose:** Manages user accounts, authentication, authorization, and session management.

**Key Tables:**

- `users` : Core user account information
- `roles` : System roles (Admin, Resilience Manager, Risk Analyst, etc.)
- `user_roles` : Many-to-many relationship between users and roles
- `permissions` : Granular permissions for resources and actions
- `role_permissions` : Many-to-many relationship between roles and permissions
- `refresh_tokens` : JWT refresh token storage for session management

**Relationships:**

- One user can have multiple roles
- One role can be assigned to multiple users
- One role can have multiple permissions
- One permission can be granted to multiple roles

```
-- Users Table
CREATE TABLE users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);

-- Roles Table
CREATE TABLE roles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(50) UNIQUE NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- User Roles Junction Table
CREATE TABLE user_roles (
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    role_id UUID REFERENCES roles(id) ON DELETE CASCADE,
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, role_id)
);

-- Permissions Table
CREATE TABLE permissions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(100) UNIQUE NOT NULL,
    resource VARCHAR(100) NOT NULL,
    action VARCHAR(50) NOT NULL
);

-- Role Permissions Junction Table
CREATE TABLE role_permissions (
    role_id UUID REFERENCES roles(id) ON DELETE CASCADE,
    permission_id UUID REFERENCES permissions(id) ON DELETE CASCADE,
    PRIMARY KEY (role_id, permission_id)
);

-- Refresh Tokens Table
CREATE TABLE refresh_tokens (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(500) NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_refresh_tokens_user_id ON refresh_tokens(user_id);
CREATE INDEX idx_refresh_tokens_token ON refresh_tokens(token);
```

**BIA Module Schema**

**Purpose:** Stores Business Impact Analysis assessments, processes, impacts, and dependencies.

**Key Tables:**

- `bia_assessments` : Main BIA assessment records
- `bia_processes` : Business processes within assessments
- `bia_impacts` : Impact analysis results
- `assets` : Assets being analyzed (referenced by assessments)

**Relationships:**

- One user can create multiple BIA assessments
- One asset can have multiple BIA assessments
- One BIA assessment contains multiple processes
- One BIA assessment has multiple impact records
- Processes can have dependencies (stored as JSONB)

```sql
-- Assets Table
CREATE TABLE assets (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    type VARCHAR(50) NOT NULL,
    description TEXT,
    metadata JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- BIA Assessments Table
CREATE TABLE bia_assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    asset_id UUID REFERENCES assets(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    assessment_data JSONB NOT NULL,
    impact_score DECIMAL(5,2),
    status VARCHAR(50) DEFAULT 'draft',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- BIA Processes Table
CREATE TABLE bia_processes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    bia_assessment_id UUID REFERENCES bia_assessments(id) ON DELETE CASCADE,
    process_name VARCHAR(255) NOT NULL,
    description TEXT,
    criticality_level VARCHAR(50),
    rto_hours INTEGER,
    rpo_hours INTEGER,
    dependencies JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- BIA Impacts Table
CREATE TABLE bia_impacts (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    bia_assessment_id UUID REFERENCES bia_assessments(id) ON DELETE CASCADE,
    impact_type VARCHAR(50) NOT NULL,
    impact_level VARCHAR(50),
    impact_score DECIMAL(5,2),
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE INDEX idx_bia_assessments_user_id ON bia_assessments(user_id);
CREATE INDEX idx_bia_assessments_asset_id ON bia_assessments(asset_id);
CREATE INDEX idx_bia_processes_assessment_id ON bia_processes(bia_assessment_id);
```

**Risk Assessment Module Schema**

**Purpose:** Manages risk assessments, identified risks, threats, vulnerabilities, and mitigation plans.

**Key Tables:**

- `risk_assessments` : Main risk assessment records
- `risks` : Individual risks identified in assessments
- `threats` : Threat entities
- `vulnerabilities` : Vulnerability records with CVSS scores
- `risk_mitigations` : Mitigation strategies for risks

**Relationships:**

- One user can create multiple risk assessments
- One asset can have multiple risk assessments
- One risk assessment contains multiple risks
- One risk assessment identifies multiple threats
- One risk assessment identifies multiple vulnerabilities
- One risk can have multiple mitigation strategies

```
-- Risk Assessments Table
CREATE TABLE risk_assessments (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    asset_id UUID REFERENCES assets(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    risk_data JSONB NOT NULL,
    risk_score DECIMAL(5,2),
    status VARCHAR(50) DEFAULT 'draft',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Risks Table
CREATE TABLE risks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    risk_assessment_id UUID REFERENCES risk_assessments(id) ON DELETE CASCADE,
    risk_name VARCHAR(255) NOT NULL,
    category VARCHAR(100),
    description TEXT,
    likelihood DECIMAL(3,2),
    impact DECIMAL(3,2),
    risk_score DECIMAL(5,2),
    mitigation_plan JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Threats Table
CREATE TABLE threats (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    risk_assessment_id UUID REFERENCES risk_assessments(id) ON DELETE CASCADE,
```

```
    threat_name VARCHAR(255) NOT NULL,
    threat_type VARCHAR(100),
    description TEXT,
    severity VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Vulnerabilities Table
CREATE TABLE vulnerabilities (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    risk_assessment_id UUID REFERENCES risk_assessments(id) ON DELETE CASCADE,
    vulnerability_name VARCHAR(255) NOT NULL,
    cvss_score DECIMAL(3,1),
    description TEXT,
    remediation_steps JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Risk Mitigations Table
CREATE TABLE risk_mitigations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    risk_id UUID REFERENCES risks(id) ON DELETE CASCADE,
    mitigation_strategy TEXT NOT NULL,
    implementation_status VARCHAR(50),
    effectiveness_score DECIMAL(3,2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_risk_assessments_user_id ON risk_assessments(user_id);
CREATE INDEX idx_risk_assessments_asset_id ON risk_assessments(asset_id);
CREATE INDEX idx_risks_assessment_id ON risks(risk_assessment_id);
CREATE INDEX idx_risks_category ON risks(category);
```

**Dashboard Module Schema**

**Purpose:** Stores dashboard configurations and cached data for performance.

**Key Tables:**

- `dashboard_configs` : User-specific dashboard widget configurations
- `dashboard_cache` : Backup cache storage (primary cache is Redis)

**Relationships:**

- One user can have multiple dashboard configurations
- Cache entries are key-value pairs with expiration

```
-- Dashboard Configurations Table
CREATE TABLE dashboard_configs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    dashboard_name VARCHAR(255) NOT NULL,
    widget_configs JSONB NOT NULL,
    layout_config JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);

-- Dashboard Cache Table (for Redis backup)
CREATE TABLE dashboard_cache (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    cache_key VARCHAR(255) UNIQUE NOT NULL,
    cache_data JSONB NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_dashboard_configs_user_id ON dashboard_configs(user_id);
CREATE INDEX idx_dashboard_cache_key ON dashboard_cache(cache_key);
CREATE INDEX idx_dashboard_cache_expires ON dashboard_cache(expires_at);
```

**Export Module Schema**

**Purpose:** Manages export jobs, templates, and generated files.

**Key Tables:**

- `export_templates` : Reusable export templates (PDF, Excel, CSV)
- `exports` : Export job records
- `export_files` : Generated export files metadata

**Relationships:**

- One export uses one template
- One export can generate multiple files
- One user can create multiple exports

```
-- Export Templates Table
CREATE TABLE export_templates (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    template_name VARCHAR(255) NOT NULL,
    template_type VARCHAR(50) NOT NULL,
    template_content TEXT NOT NULL,
    format VARCHAR(20) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Exports Table
CREATE TABLE exports (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE CASCADE,
    export_type VARCHAR(50) NOT NULL,
    format VARCHAR(20) NOT NULL,
    template_id UUID REFERENCES export_templates(id),
    file_path VARCHAR(500),
    file_size BIGINT,
    status VARCHAR(50) DEFAULT 'pending',
    error_message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    completed_at TIMESTAMP
);
```

```sql
-- Export Files Table
CREATE TABLE export_files (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    export_id UUID REFERENCES exports(id) ON DELETE CASCADE,
    file_name VARCHAR(255) NOT NULL,
    file_path VARCHAR(500) NOT NULL,
    file_type VARCHAR(50),
    file_size BIGINT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_exports_user_id ON exports(user_id);
CREATE INDEX idx_exports_status ON exports(status);
CREATE INDEX idx_export_files_export_id ON export_files(export_id);
```

**AI Module Schema**

**Purpose:** Tracks AI/ML analyses, embeddings metadata, and audit logs for AI-generated content.

**Key Tables:**

- `ai_analyses` : Records of AI analysis operations
- `ai_embeddings_metadata` : Metadata for vector DB embeddings
- `ai_audit_log` : Audit trail for AI operations and human overrides

**Relationships:**

- One analysis references one assessment (BIA or RA)
- One analysis can have multiple audit log entries
- Embeddings metadata links to content in vector DB
- Low confidence analyses (<0.7) are flagged for human review

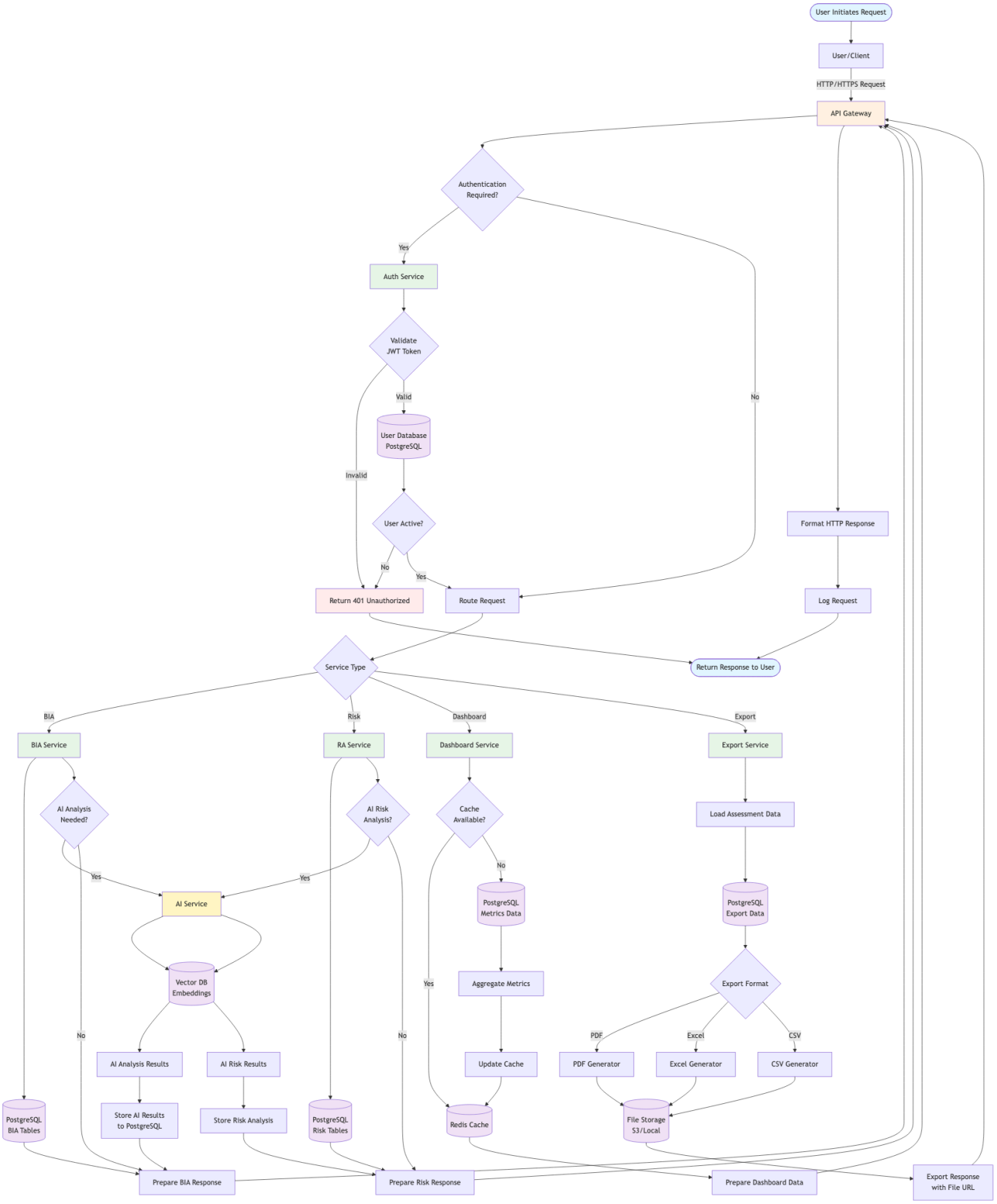
```sql
-- AI Analyses Table
CREATE TABLE ai_analyses (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    analysis_type VARCHAR(50) NOT NULL,
    reference_id UUID NOT NULL,
    reference_type VARCHAR(50) NOT NULL,
    input_data JSONB NOT NULL,
    output_data JSONB NOT NULL,
    model_name VARCHAR(255) NOT NULL,
    model_version VARCHAR(50) NOT NULL,
    confidence_score DECIMAL(3,2),
    requires_review BOOLEAN DEFAULT false,
    reviewed_by UUID REFERENCES users(id),
    reviewed_at TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- AI Embeddings Table (metadata for vector DB)
CREATE TABLE ai_embeddings_metadata (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    embedding_id VARCHAR(255) NOT NULL,
    vector_index_name VARCHAR(100) NOT NULL,
    content_type VARCHAR(50) NOT NULL,
    content_id UUID NOT NULL,
    embedding_dimensions INTEGER NOT NULL,
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- AI Audit Log Table
CREATE TABLE ai_audit_log (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id),
    analysis_id UUID REFERENCES ai_analyses(id),
    action VARCHAR(50) NOT NULL,
    prior_output JSONB,
    override_output JSONB,
    reason TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_ai_analyses_reference ON ai_analyses(reference_type, reference_id);
CREATE INDEX idx_ai_analyses_type ON ai_analyses(analysis_type);
CREATE INDEX idx_ai_analyses_confidence ON ai_analyses(confidence_score);
CREATE INDEX idx_ai_audit_log_user_id ON ai_audit_log(user_id);
CREATE INDEX idx_ai_audit_log_analysis_id ON ai_audit_log(analysis_id);
```

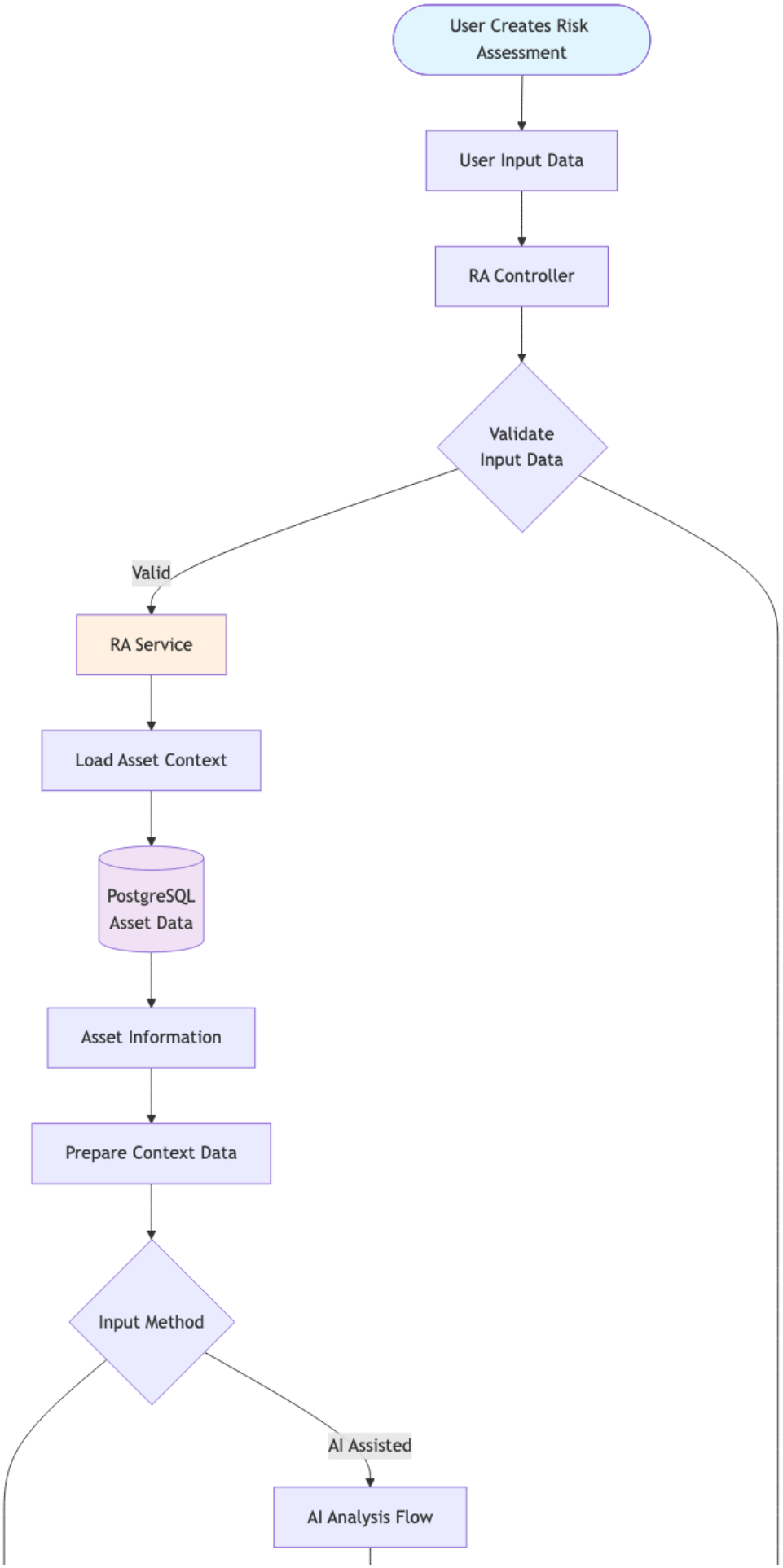
---

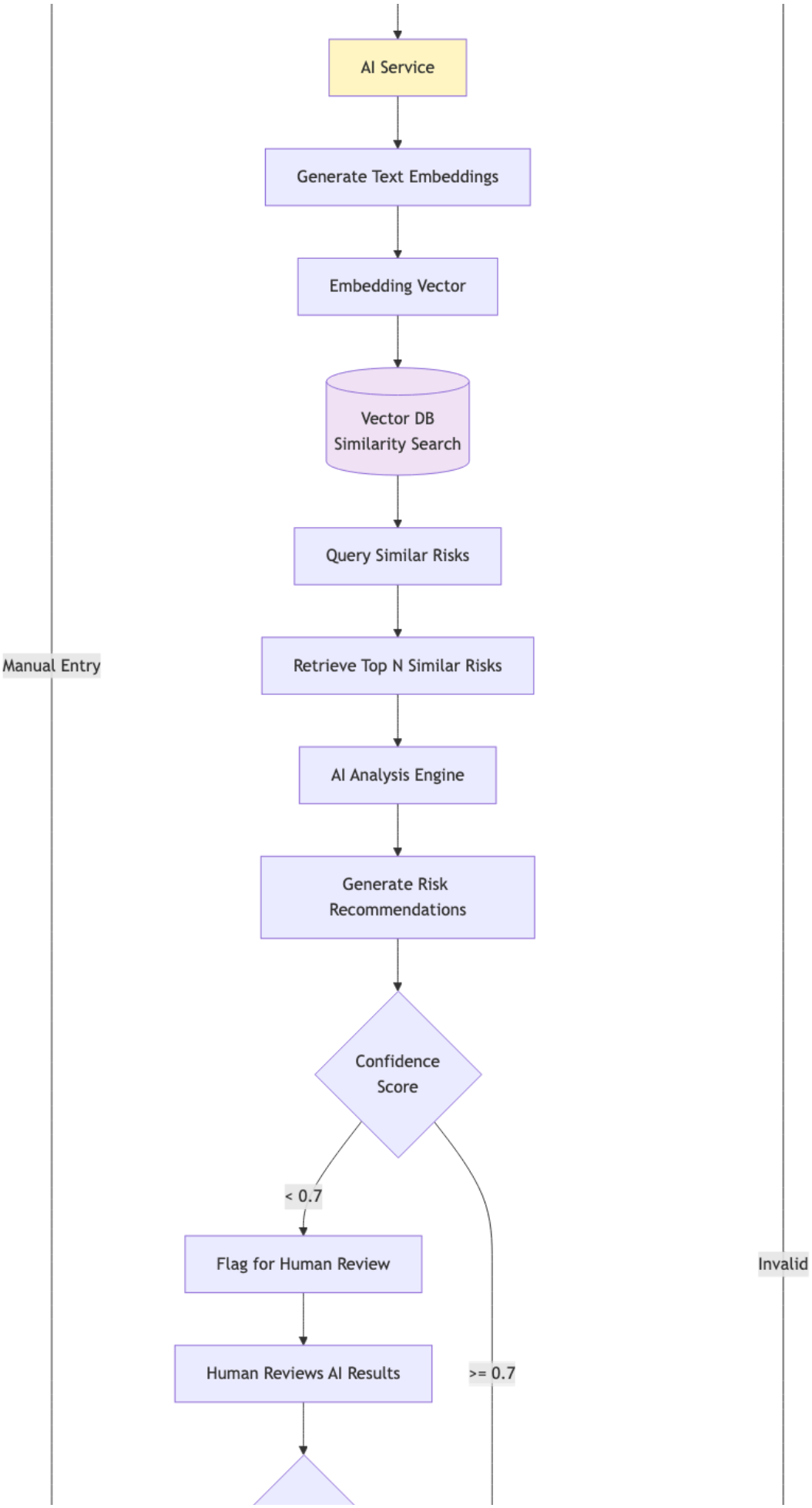
## Data Flow Diagrams

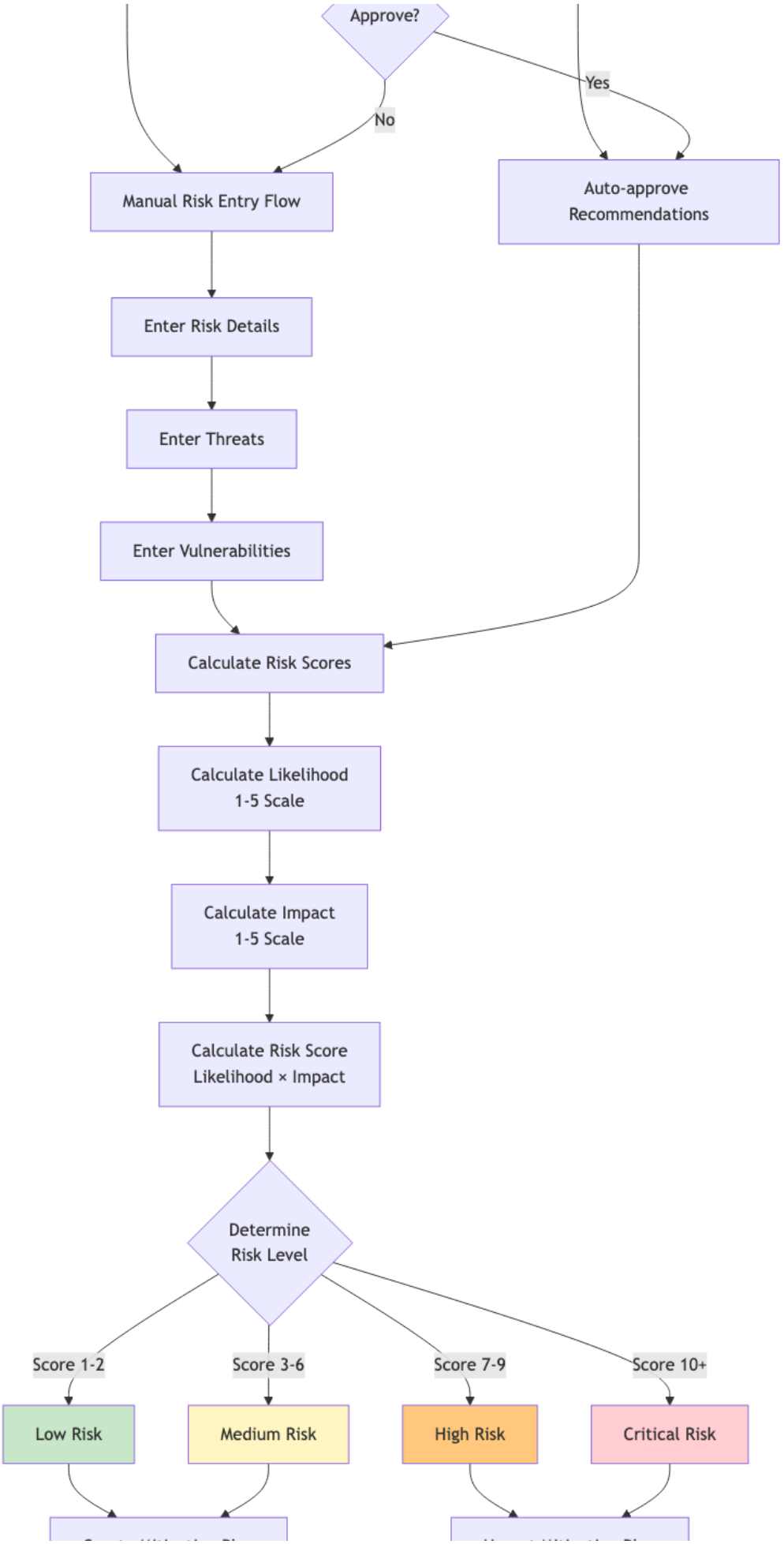
### System-Level Data Flow

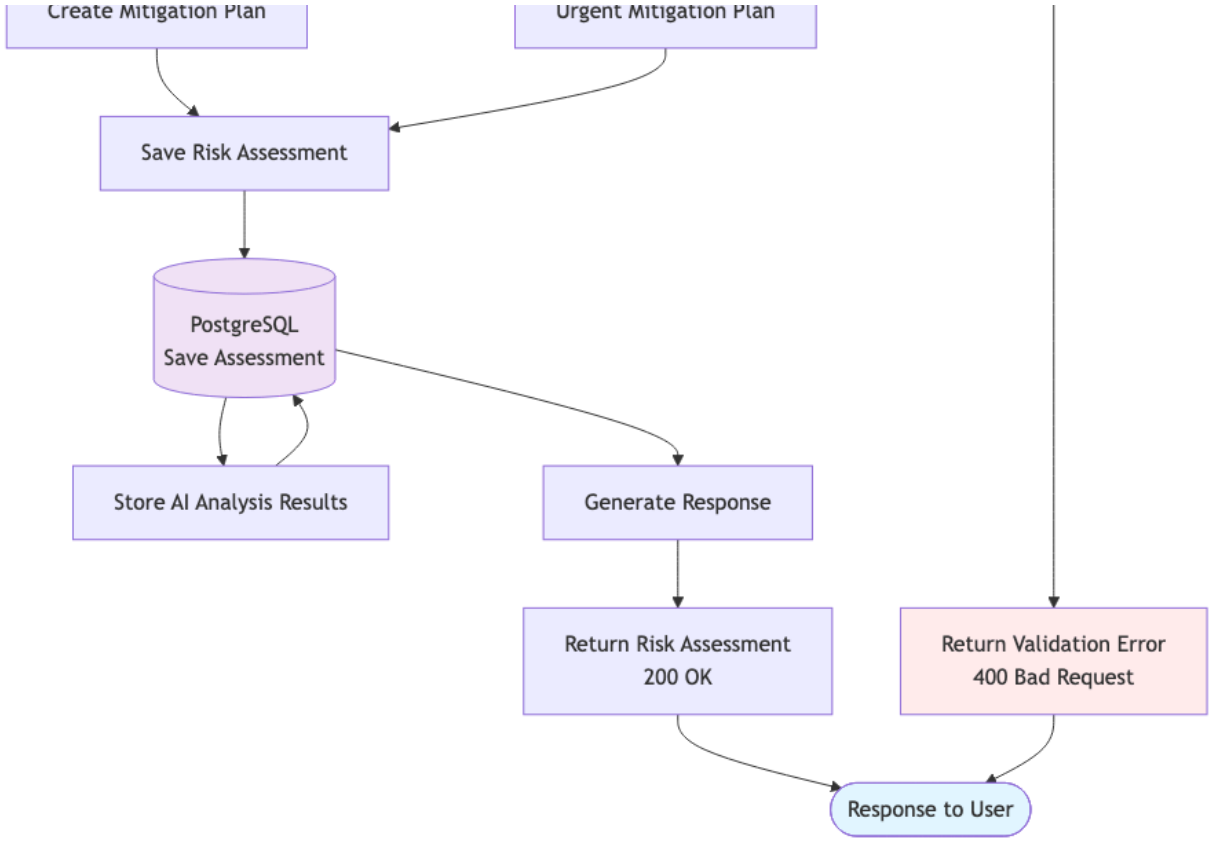



## Risk Assessment Data Flow

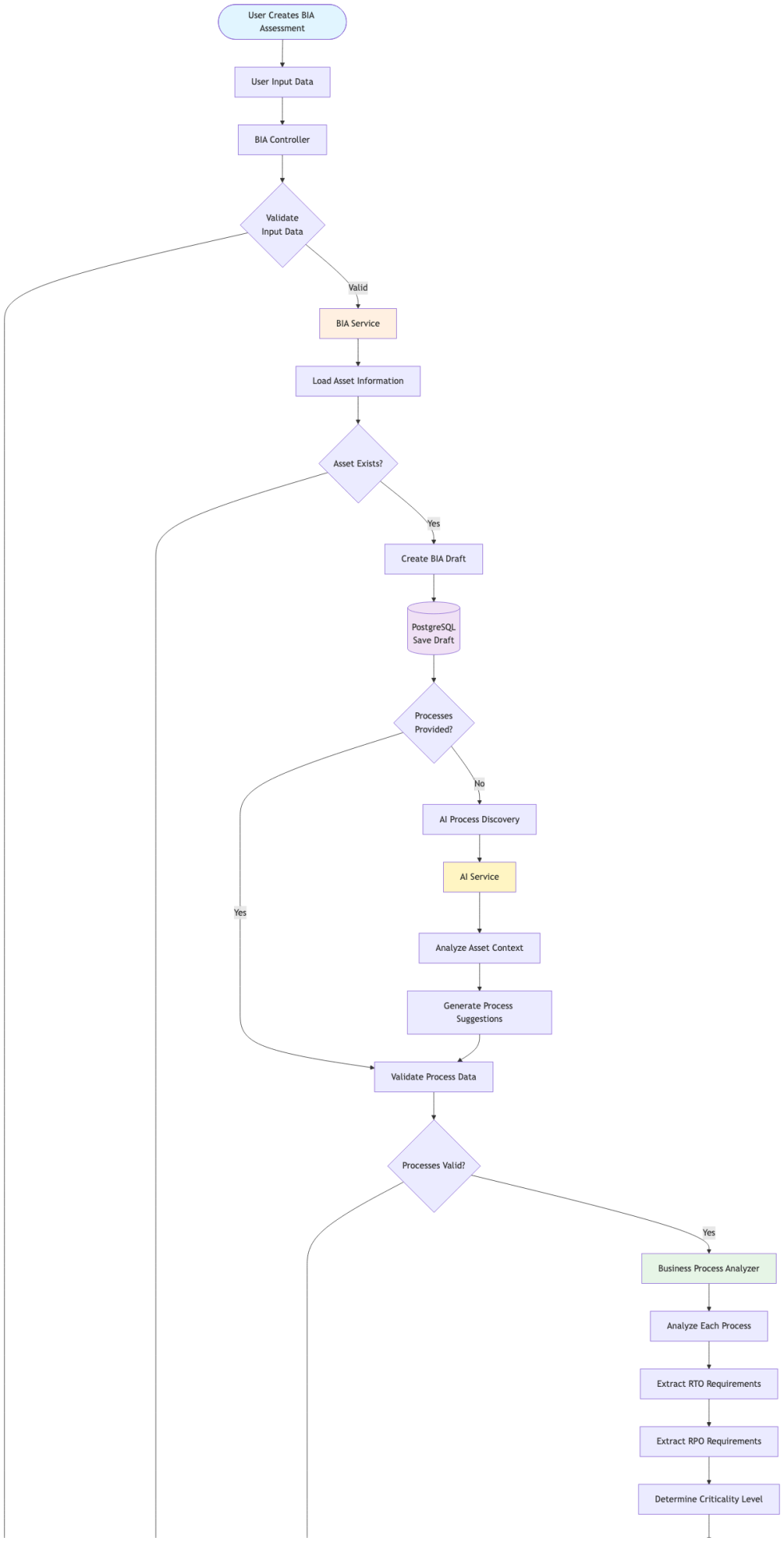

User Creates Risk Assessment
User Input Data
RA Controller
Validate Input Data
Valid
RA Service
Load Asset Context
PostgreSQL Asset Data
Asset Information
Prepare Context Data
Input Method
AI Assisted
AI Analysis Flow

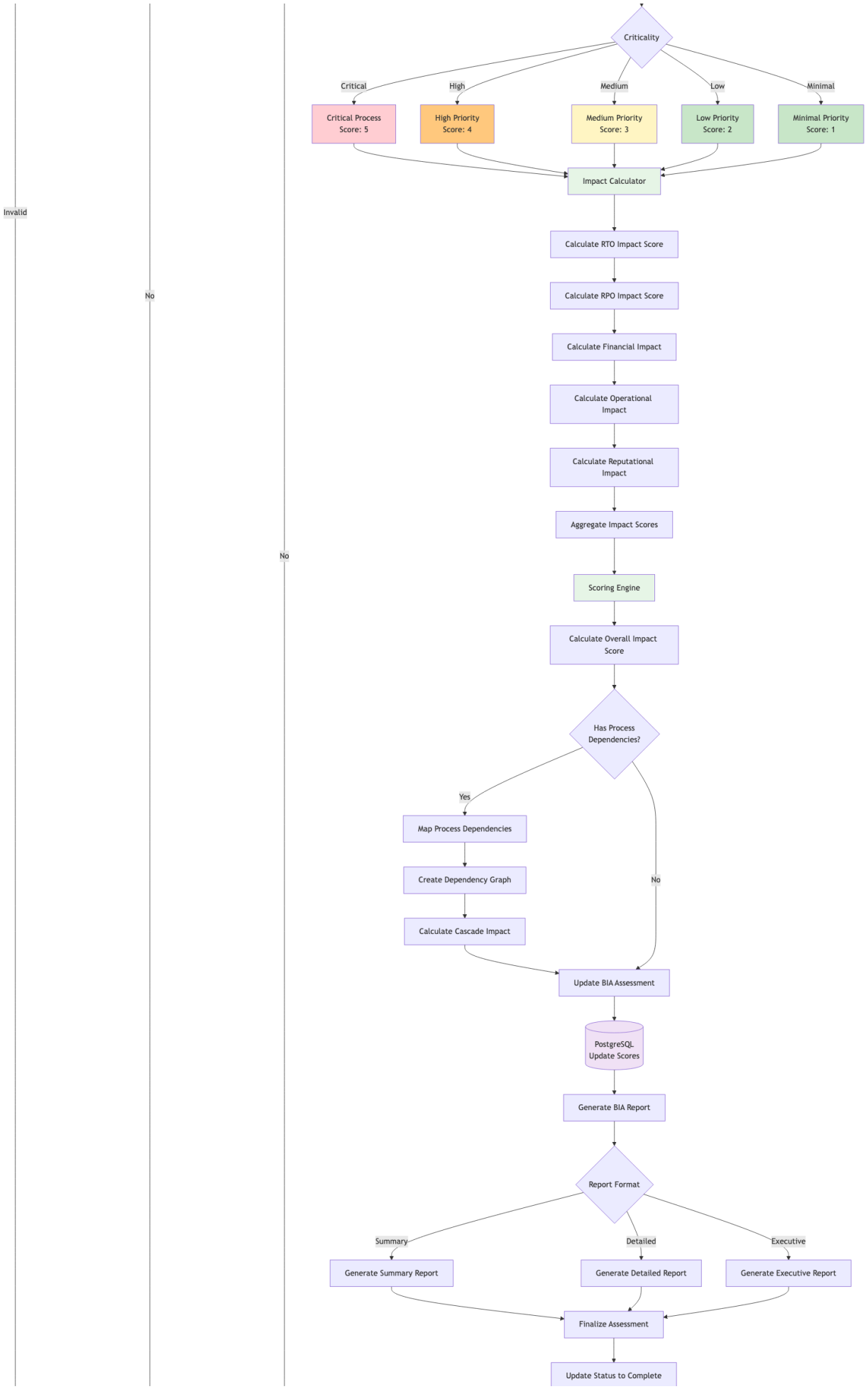

AI Service
Generate Text Embeddings
Embedding Vector
Vector DB
Similarity Search
Query Similar Risks
Manual Entry
Retrieve Top N Similar Risks
AI Analysis Engine
Generate Risk
Recommendations
Confidence
Score
< 0.7
Flag for Human Review
Invalid
Human Reviews AI Results
>= 0.7

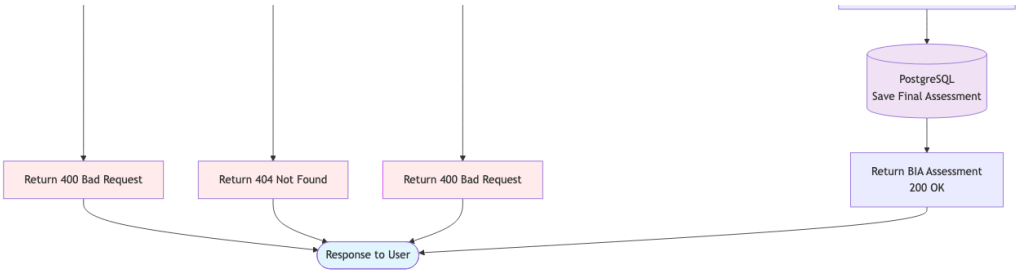

Approve?
No
Yes
Manual Risk Entry Flow
Auto-approve Recommendations
Enter Risk Details
Enter Threats
Enter Vulnerabilities
Calculate Risk Scores
Calculate Likelihood 1-5 Scale
Calculate Impact 1-5 Scale
Calculate Risk Score Likelihood × Impact
Determine Risk Level
Score 1-2
Score 3-6
Score 7-9
Score 10+
Low Risk
Medium Risk
High Risk
Critical Risk



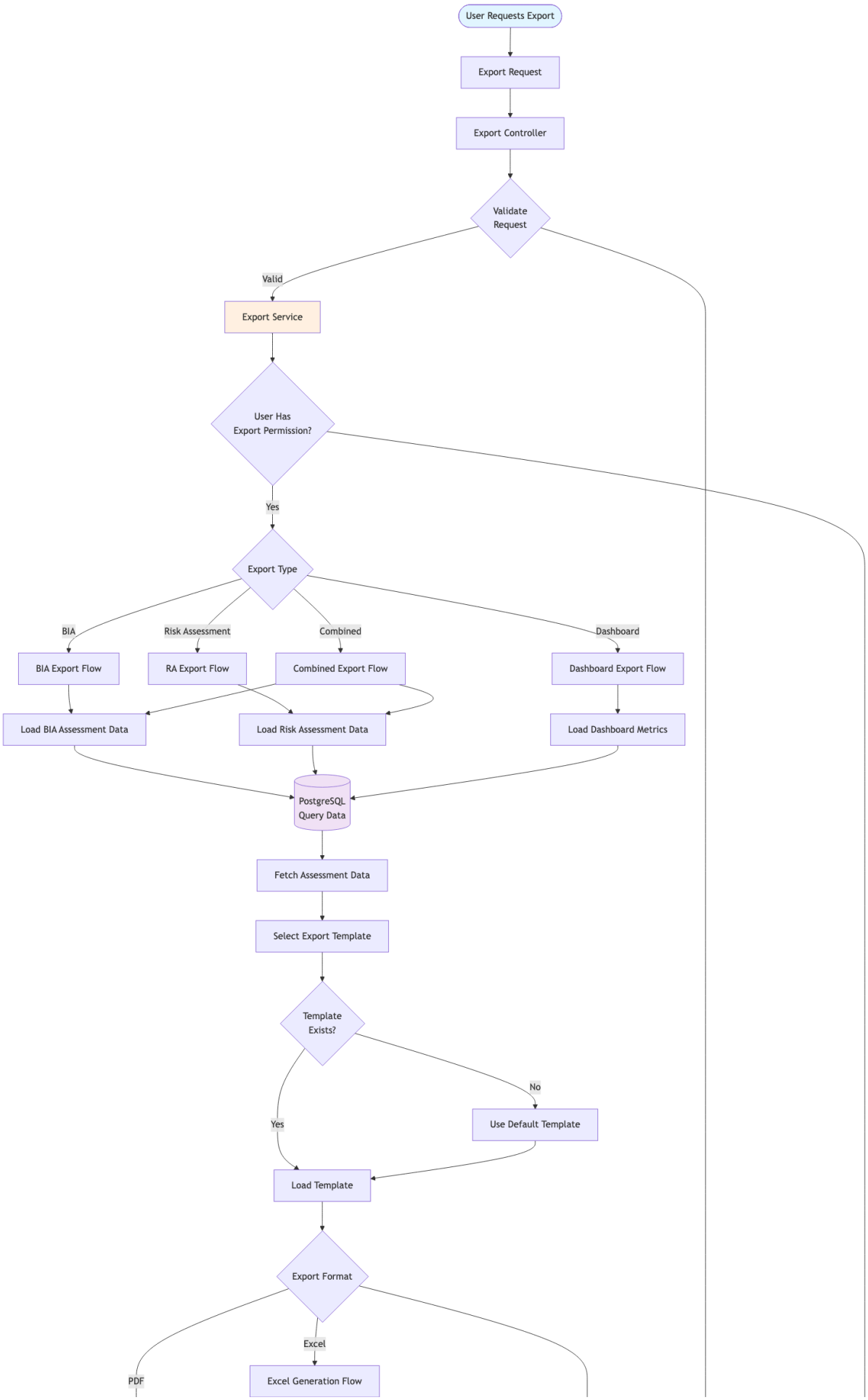

**BIA Assessment Data Flow**

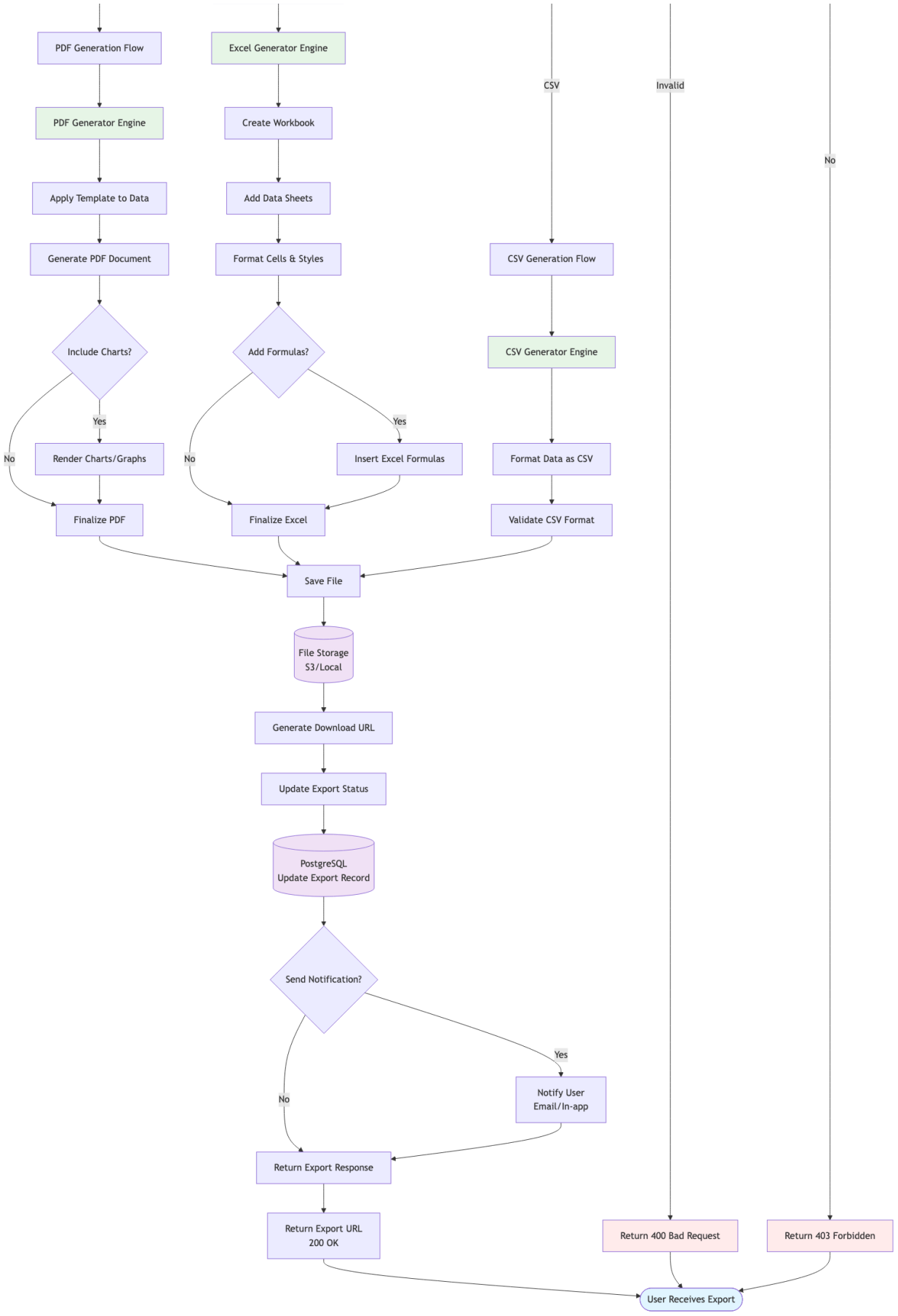

User Creates BIA Assessment
User Input Data
BIA Controller
Validate Input Data
Valid
BIA Service
Load Asset Information
Asset Exists?
Yes
Create BIA Draft
PostgreSQL Save Draft
Processes Provided?
No
AI Process Discovery
AI Service
Yes
Analyze Asset Context
Generate Process Suggestions
Validate Process Data
Processes Valid?
Yes
Business Process Analyzer
Analyze Each Process
Extract RTO Requirements
Extract RPO Requirements
Determine Criticality Level


Invalid
No
No
Criticality
Critical
High
Medium
Low
Minimal
Critical Process Score: 5
High Priority Score: 4
Medium Priority Score: 3
Low Priority Score: 2
Minimal Priority Score: 1
Impact Calculator
Calculate RTO Impact Score
Calculate RPO Impact Score
Calculate Financial Impact
Calculate Operational Impact
Calculate Reputational Impact
Aggregate Impact Scores
Scoring Engine
Calculate Overall Impact Score
Has Process Dependencies?
Yes
No
Map Process Dependencies
Create Dependency Graph
Calculate Cascade Impact
Update BIA Assessment
PostgreSQL Update Scores
Generate BIA Report
Report Format
Summary
Detailed
Executive
Generate Summary Report
Generate Detailed Report
Generate Executive Report
Finalize Assessment
Update Status to Complete




## Export Data Flow

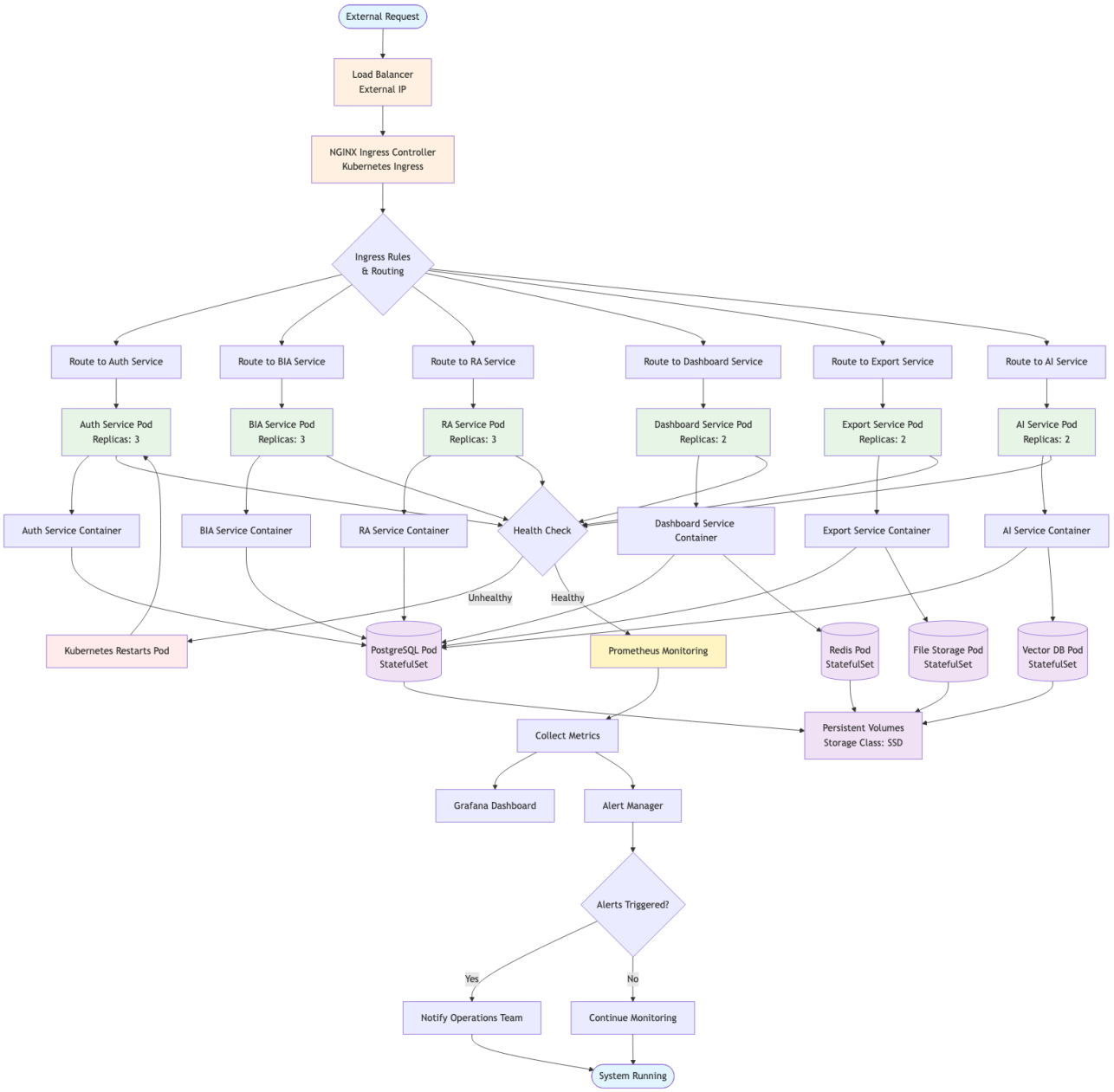

User Requests Export
Export Request
Export Controller
Validate Request
Valid
Export Service
User Has Export Permission?
Yes
Export Type
BIA
Risk Assessment
Combined
Dashboard
BIA Export Flow
RA Export Flow
Combined Export Flow
Dashboard Export Flow
Load BIA Assessment Data
Load Risk Assessment Data
Load Dashboard Metrics
PostgreSQL Query Data
Fetch Assessment Data
Select Export Template
Template Exists?
No
Yes
Use Default Template
Load Template
Export Format
Excel
PDF
Excel Generation Flow



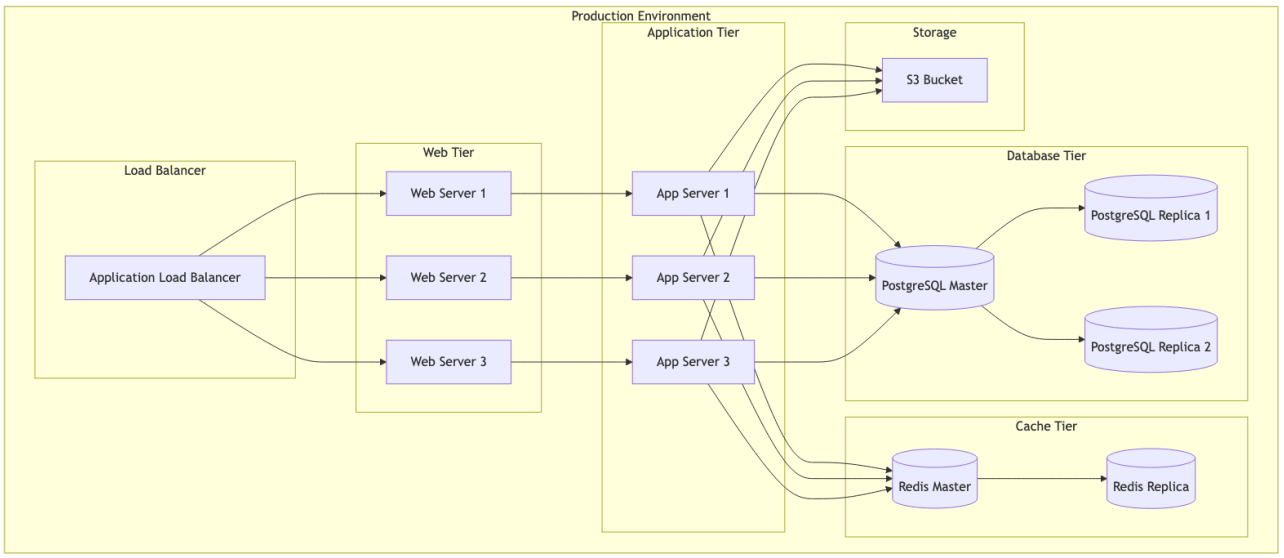

---

## Deployment Architecture

**Container Architecture**




## Deployment Diagram



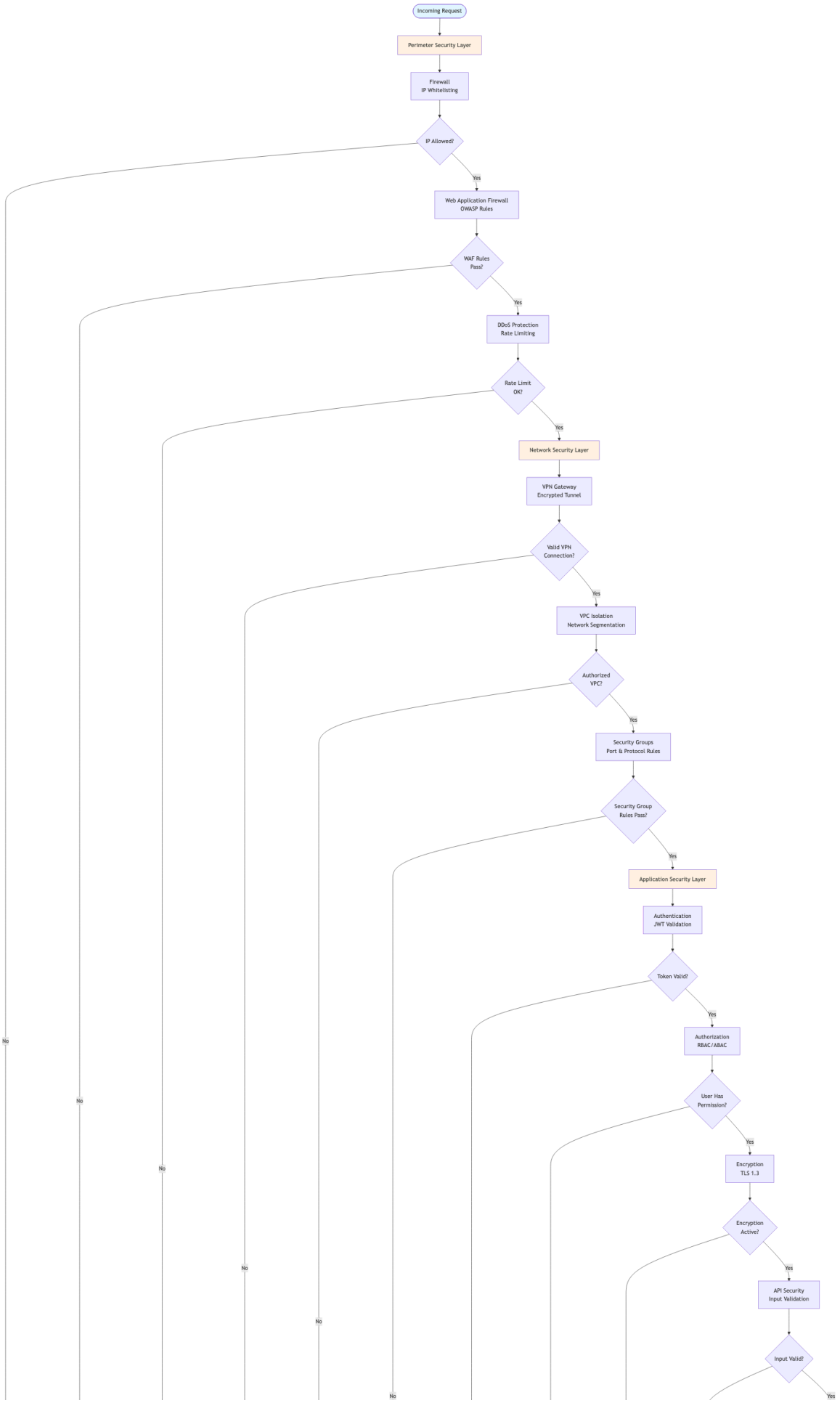

### Docker Compose Architecture

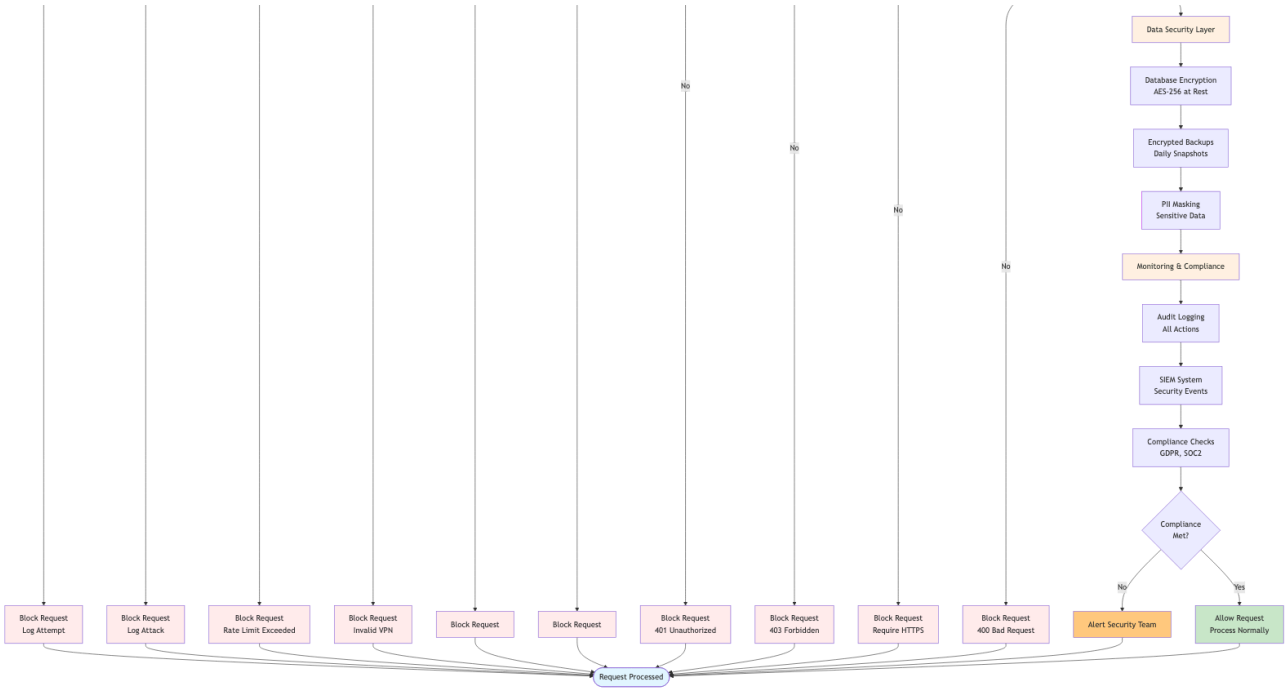
```
# docker-compose.yml structure (conceptual)
services:
  - api-gateway
  - auth-service
  - bia-service
  - ra-service
  - dashboard-service
  - export-service
  - ai-service
  - postgresql
  - redis
  - vector-db
  - nginx
```

---

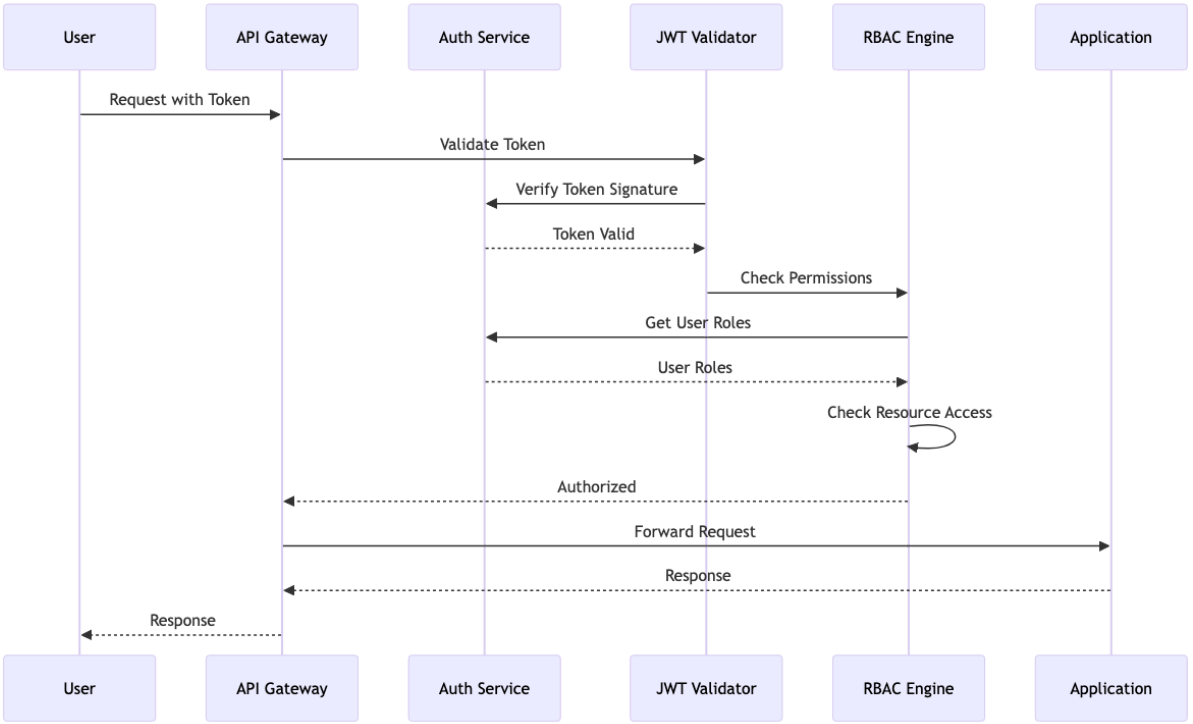
## Security Architecture

### Security Layers

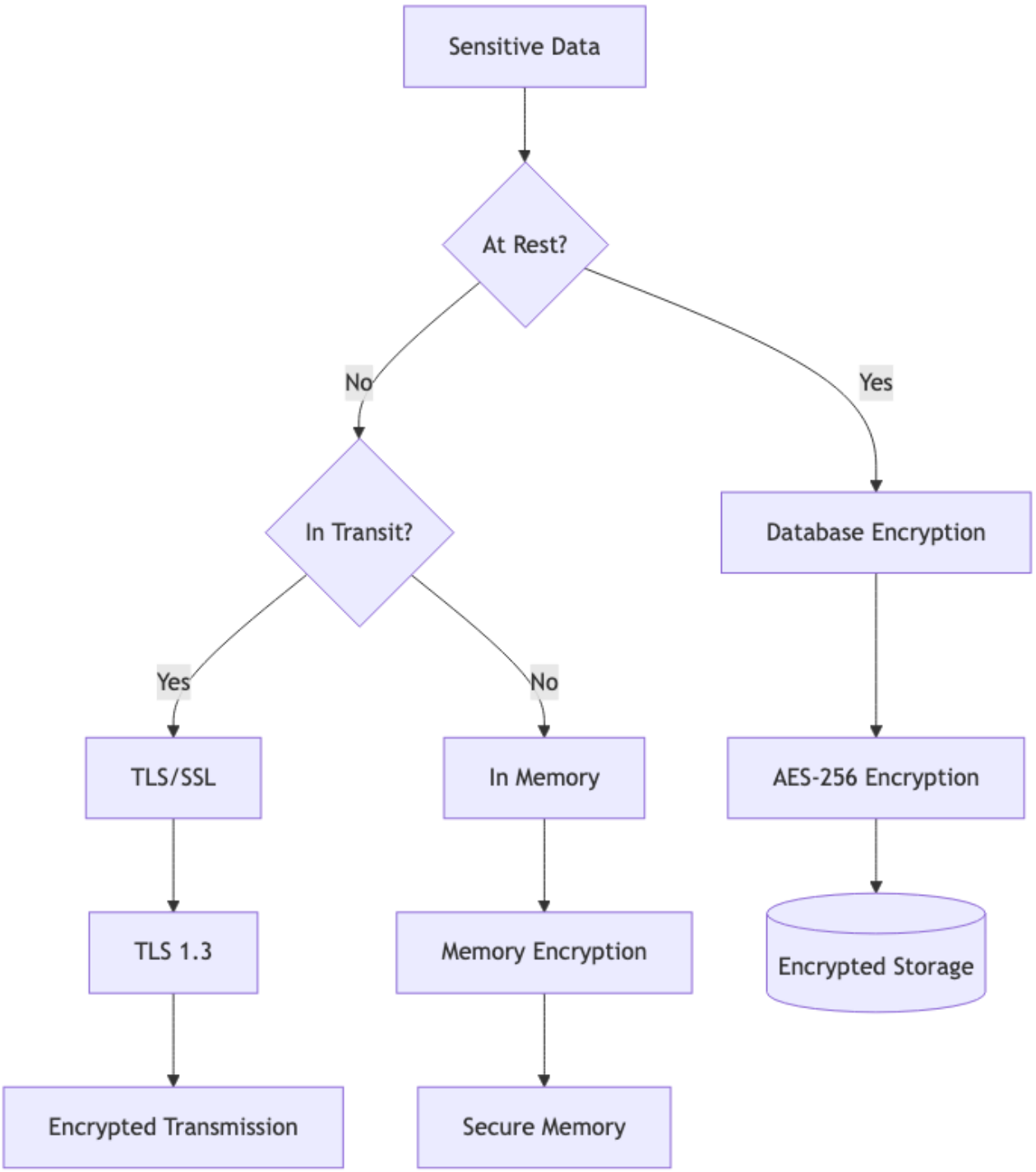

Incoming Request
Perimeter Security Layer
Firewall
IP Whitelisting
IP Allowed?
Yes
No
Web Application Firewall
OWASP Rules
WAF Rules
Pass?
Yes
No
DDoS Protection
Rate Limiting
Rate Limit
OK?
Yes
No
Network Security Layer
VPN Gateway
Encrypted Tunnel
Valid VPN
Connection?
Yes
No
VPC Isolation
Network Segmentation
Authorized
VPC?
Yes
No
Security Groups
Port & Protocol Rules
Security Group
Rules Pass?
Yes
No
Application Security Layer
Authentication
JWT Validation
Token Valid?
Yes
No
Authorization
RBAC/ABAC
User Has
Permission?
Yes
No
Encryption
TLS 1.3
Encryption
Active?
Yes
No
API Security
Input Validation
Input Valid?
Yes
No




## Authentication & Authorization Flow




## Data Encryption Flow




### Security Controls Matrix

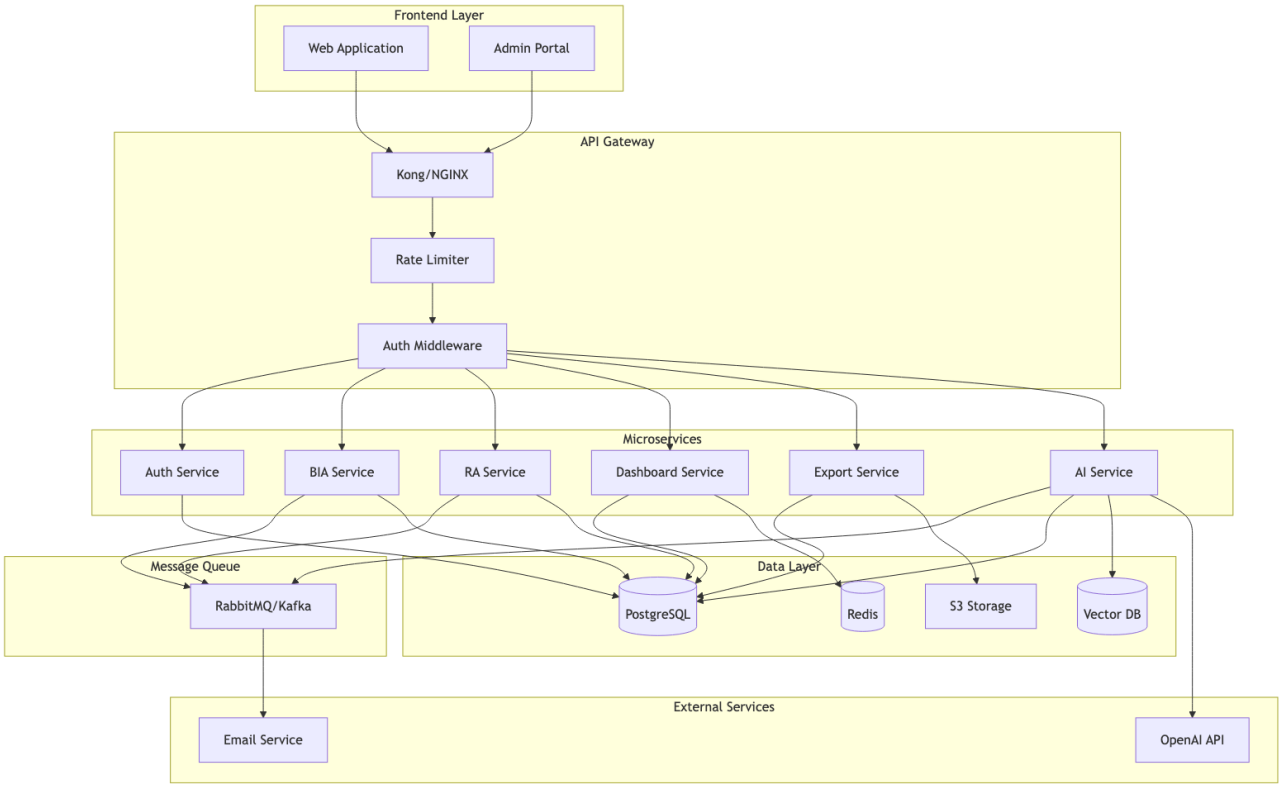
| Control Type | Implementation | Location |
|---|---|---|
| Authentication | JWT Tokens, OAuth 2.0 | API Gateway, Auth Service |
| Authorization | RBAC, ABAC | All Services |
| Encryption at Rest | AES-256 | Database, File Storage |
| Encryption in Transit | TLS 1.3 | All API Endpoints |
| Input Validation | Schema Validation | All Controllers |

| | | |
|---|---|---|
| SQL Injection Prevention | Parameterized Queries | All Repositories |
| XSS Prevention | Content Security Policy | Frontend |
| CSRF Protection | CSRF Tokens | API Gateway |
| Rate Limiting | Token Bucket | API Gateway |
| Audit Logging | Structured Logs | All Services |
| Secrets Management | Vault/KMS | Environment Variables |

---

## Module-Wise Architecture

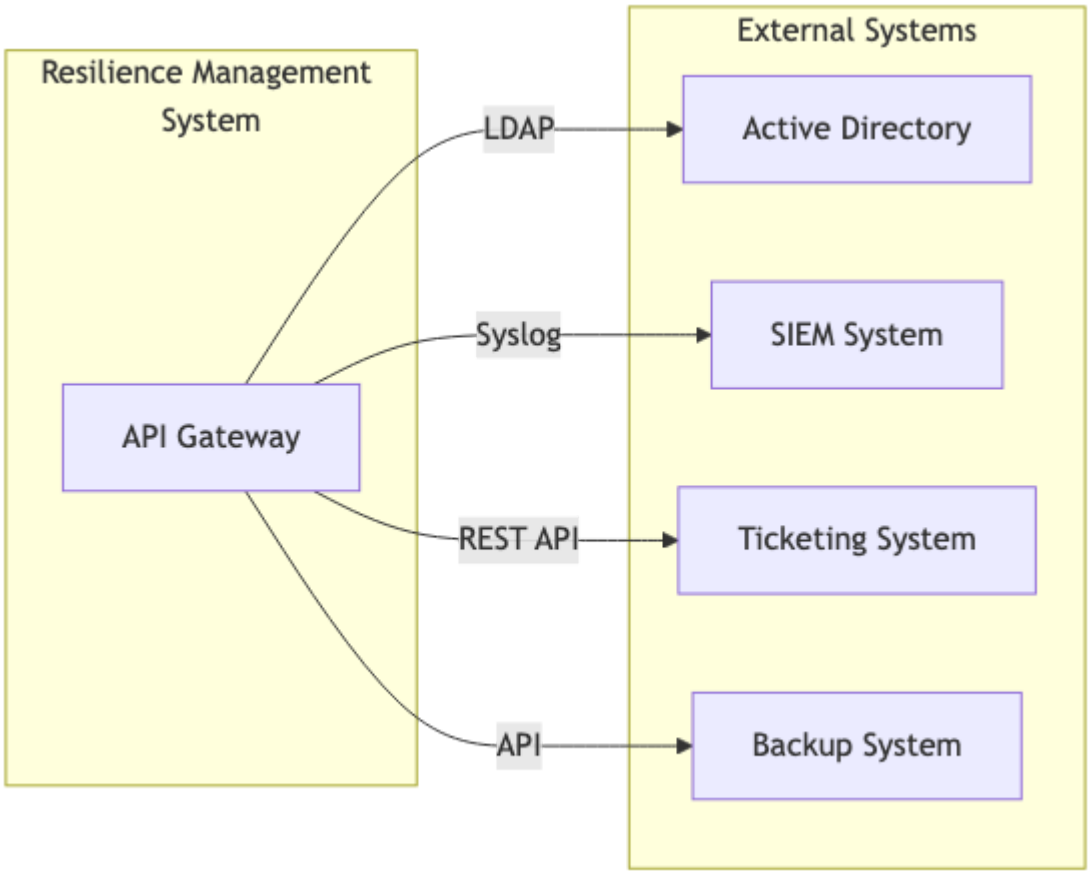
### Complete System Architecture




---

## Integration Architecture

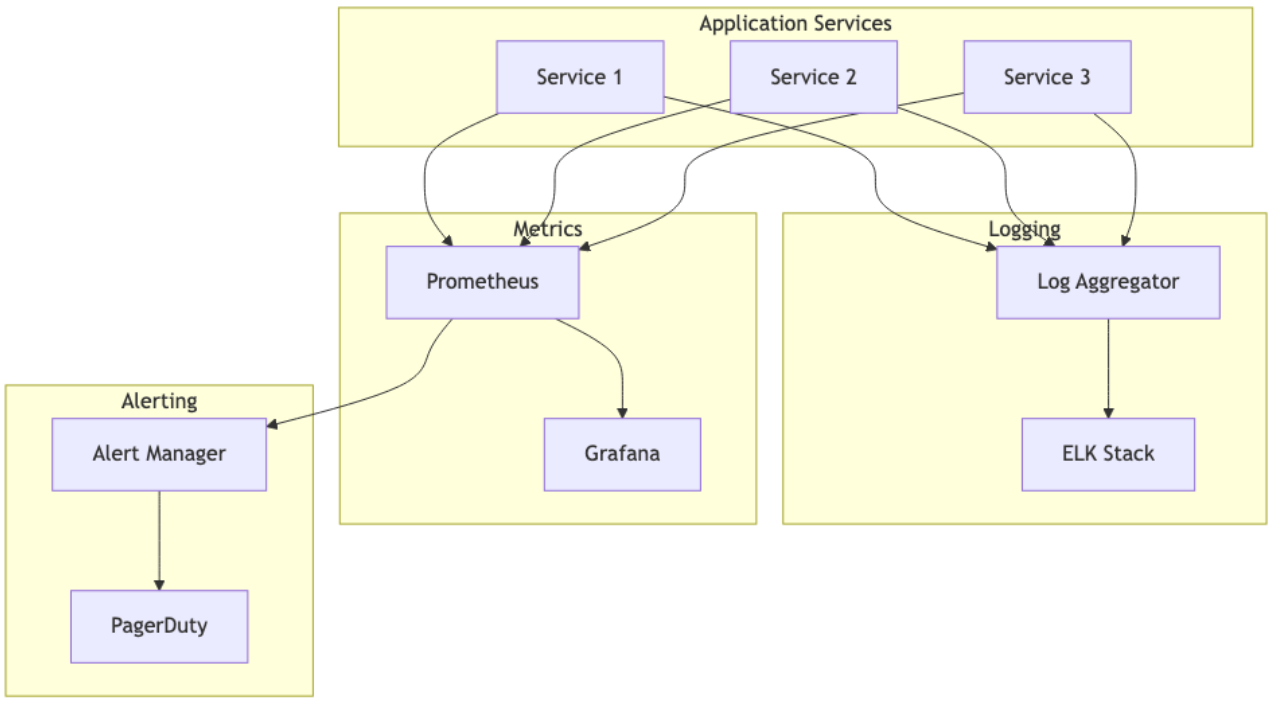
### External System Integration



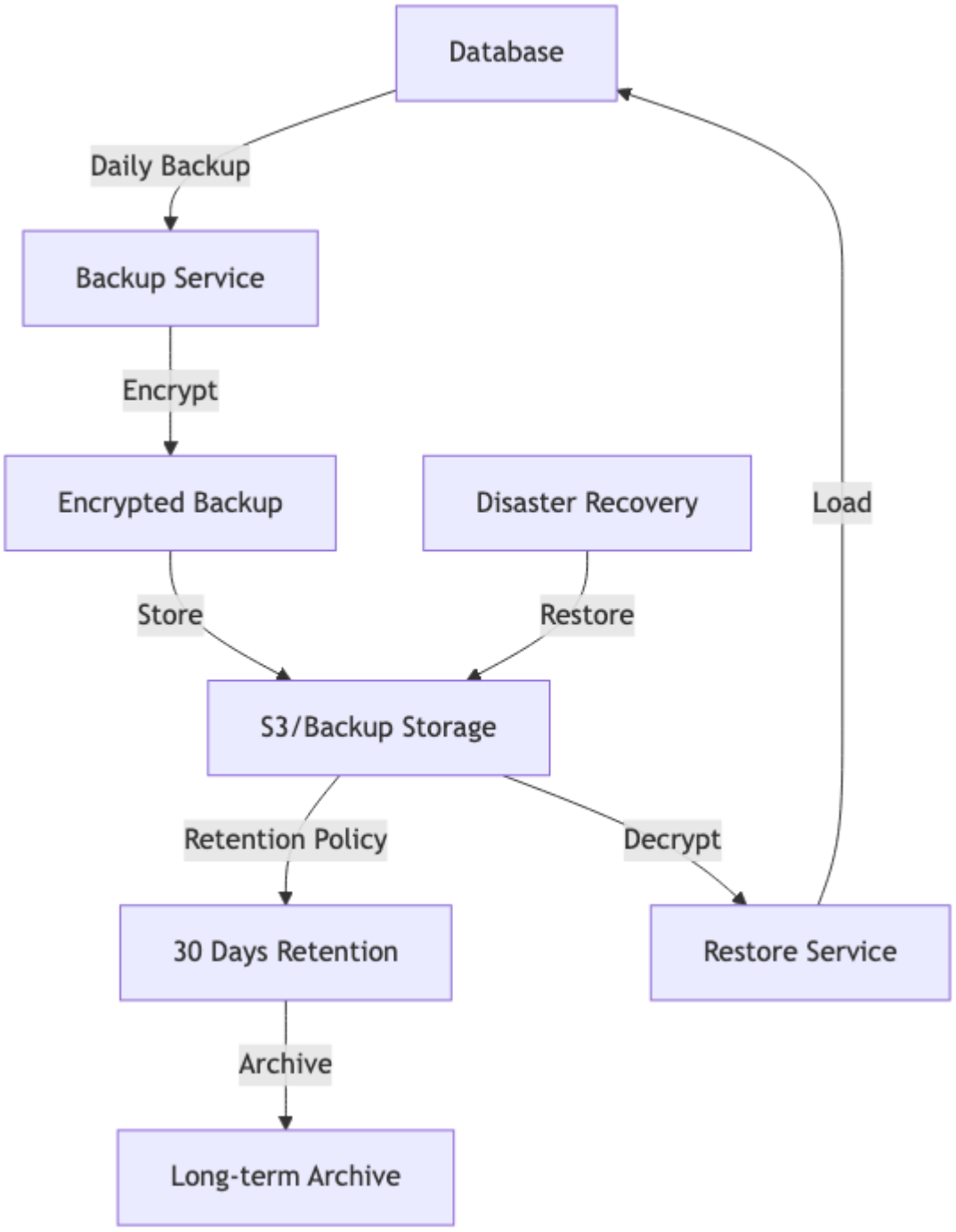

---

## Operational Architecture

### Monitoring & Logging



**Backup & Recovery**




---

## Appendix

### Technology Versions

- **PostgreSQL:** 14.x
- **Redis:** 7.x
- **Node.js:** 18.x LTS
- **Python:** 3.11+
- **Docker:** 24.x
- **Kubernetes:** 1.28+

### Performance Requirements

- **API Response Time:** < 200ms (p95)
- **Database Query Time:** < 100ms (p95)
- **Concurrent Users:** 1000+
- **Throughput:** 10,000 requests/minute

### Scalability Considerations

- Horizontal scaling for stateless services
- Database read replicas for read-heavy operations
- Caching strategy for frequently accessed data
- Async processing for long-running tasks

---

**Document End**