**System Architecture Overview**

**Document Version:** 1.0  
**Last Updated:** September 2025  
**Author:** Dr. Yishuang Xu, University of Manchester

**Executive Summary**

This document provides a comprehensive overview of the system architecture for the AI-Powered Sustainability Regulatory Compliance Platform. It defines the high-level system design, component interactions, data flow, and deployment architecture necessary to support EU sustainability regulation compliance for real estate portfolios.

**Architecture Principles**

**Design Philosophy**

* **Microservices Architecture:** Loosely coupled, independently deployable services
* **Event-Driven Design:** Asynchronous processing with message queues
* **API-First Approach:** All functionality exposed through well-defined APIs
* **Cloud-Native:** Designed for containerized deployment and auto-scaling
* **Security by Design:** Security integrated at every architectural layer

**Quality Attributes**

* **Scalability:** Support 10,000+ properties and 100+ concurrent users
* **Reliability:** 99.9% uptime with automated failover
* **Performance:** Sub-second response times for critical operations
* **Maintainability:** Modular design with clear separation of concerns
* **Extensibility:** Plugin architecture for new regulations and data sources

**High-Level System Architecture**

**System Context Diagram**

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

│ External Systems & Users │

├─────────────────────────────────────────────────────────────────┤

│ │

│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │

│ │ Users │ │ Property │ │ Regulatory │ │

│ │(Web/Mobile) │ │ Management │ │ Databases │ │

│ │ │ │ Systems │ │ │ │

│ └─────────────┘ └─────────────┘ └─────────────┘ │

│ │ │ │ │

└─────────┼────────────────┼────────────────┼────────────────────┘

│ │ │

▼ ▼ ▼

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

│ API Gateway Layer │

│ ┌─────────────────────────────────────────────────────────────┐│

│ │ Authentication │ Rate Limiting │ Load Balancing │ SSL Term. ││

│ └─────────────────────────────────────────────────────────────┘│

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

│

▼

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

│ Application Services │

│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌────────────┐│

│ │ Property │ │ Compliance │ │ AI/ML │ │ Reporting ││

│ │ Service │ │ Service │ │ Service │ │ Service ││

│ └─────────────┘ └─────────────┘ └─────────────┘ └────────────┘│

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

│

▼

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

│ Data & Infrastructure │

│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌────────────┐│

│ │ PostgreSQL │ │ Redis Cache │ │ File Storage│ │ Message ││

│ │ Database │ │ │ │ (S3/Blob) │ │ Queue ││

│ └─────────────┘ └─────────────┘ └─────────────┘ └────────────┘│

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

**Core Component Architecture**

**1. API Gateway Layer**

**Purpose**

Central entry point for all client requests, providing cross-cutting concerns.

**Components**

Load Balancer:

- Technology: NGINX or AWS ALB

- Function: Distribute traffic across service instances

- Health Checks: Automatic failover for unhealthy instances

API Gateway:

- Technology: Kong, AWS API Gateway, or custom FastAPI

- Functions:

- Request routing and protocol translation

- Authentication and authorization

- Rate limiting and throttling

- Request/response transformation

- API versioning and documentation

Security Layer:

- SSL/TLS termination

- Web Application Firewall (WAF)

- DDoS protection

- Input validation and sanitization

**2. Application Services Layer**

**2.1 Property Management Service**

Responsibilities:

- Property master data management

- Building information modeling

- Asset hierarchy management

- Utility and consumption data tracking

Key Components:

- Property Repository: CRUD operations for property data

- Data Validation Engine: Ensure data quality and consistency

- Integration Hub: Connect to external property management systems

- Audit Trail: Track all data changes with versioning

API Endpoints:

- GET/POST/PUT/DELETE /properties

- GET/POST /properties/{id}/utilities

- GET /properties/{id}/hierarchy

- POST /properties/bulk-import

**2.2 Compliance Service**

Responsibilities:

- EU Taxonomy alignment assessment

- SFDR PAI calculations

- CSRD reporting preparation

- Regulatory requirement tracking

Key Components:

- Taxonomy Engine: Map properties to technical screening criteria

- PAI Calculator: Compute Principal Adverse Impact indicators

- Compliance Scorer: Generate compliance ratings and evidence

- Regulatory Monitor: Track requirement changes and updates

API Endpoints:

- GET /compliance/{property\_id}/taxonomy

- POST /compliance/pai-calculation

- GET /compliance/requirements/{regulation}

- POST /compliance/bulk-assessment

**2.3 AI/ML Service**

Responsibilities:

- Document processing and data extraction

- Energy consumption forecasting

- Performance optimization recommendations

- Regulatory text analysis

Key Components:

- NLP Pipeline: Process regulatory documents and requirements

- Computer Vision Engine: Extract data from property documents

- Forecasting Models: Predict energy and sustainability metrics

- Recommendation Engine: Generate improvement suggestions

API Endpoints:

- POST /ml/document-processing

- GET /ml/forecasts/{property\_id}

- POST /ml/recommendations

- GET /ml/model-performance

**2.4 Reporting Service**

Responsibilities:

- Generate compliance reports

- Create data visualizations

- Export capabilities

- Dashboard data aggregation

Key Components:

- Report Generator: Create PDF/Excel reports from templates

- Chart Builder: Generate interactive visualizations

- Export Engine: Multiple format support with scheduling

- Aggregation Service: Real-time dashboard metrics

API Endpoints:

- POST /reports/generate

- GET /reports/{report\_id}/status

- GET /dashboards/metrics

- POST /exports/schedule

**AI/ML Architecture Deep Dive**

**ML Pipeline Architecture**

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

│ ML Pipeline Overview │

├─────────────────────────────────────────────────────────────────┤

│ │

│ Data Ingestion → Feature Engineering → Model Training │

│ │ │ │ │

│ ▼ ▼ ▼ │

│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │

│ │ Raw Data │ │ Features │ │ Models │ │

│ │ Validation │ │ Extraction │ │ Training │ │

│ └─────────────┘ └─────────────┘ └─────────────┘ │

│ │ │ │ │

│ ▼ ▼ ▼ │

│ Model Evaluation → Model Deployment → Inference Pipeline │

│ │

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

**1. Natural Language Processing Pipeline**

**Regulatory Document Processing**

Input: EU regulation PDFs, updates, amendments

Pipeline Stages:

1. Document Preprocessing:

- PDF text extraction with layout preservation

- Text cleaning and normalization

- Language detection and encoding

2. Information Extraction:

- Named Entity Recognition (NER) for regulatory entities

- Relationship extraction between requirements

- Technical criteria identification

3. Semantic Analysis:

- Document embedding generation

- Similarity matching with property descriptions

- Change detection in regulatory updates

4. Knowledge Graph Construction:

- Entity relationship mapping

- Regulatory requirement hierarchies

- Cross-reference link creation

Output: Structured regulatory knowledge base

Technology Stack: spaCy, Transformers, BERT-multilingual

Performance: 1000+ pages/hour, 95%+ accuracy

**2. Computer Vision Pipeline**

**Document Data Extraction**

Input: Utility bills, EPCs, certificates (PDF/Image)

Pipeline Stages:

1. Image Preprocessing:

- Deskewing and rotation correction

- Noise reduction and contrast enhancement

- Resolution optimization

2. Layout Analysis:

- Document structure detection

- Table and form recognition

- Field boundary identification

3. Text Recognition:

- OCR with confidence scoring

- Multiple language support

- Custom training for utility bill formats

4. Data Validation:

- Value range checking

- Format validation

- Cross-field consistency checks

Output: Structured property data with confidence scores

Technology Stack: Tesseract, OpenCV, custom CNN models

Performance: 100 docs/hour, 95%+ accuracy for printed text

**3. Time Series Forecasting Pipeline**

**Energy Consumption Prediction**

Input: Historical energy consumption, weather, building data

Pipeline Stages:

1. Data Preprocessing:

- Missing value imputation

- Outlier detection and treatment

- Seasonal decomposition

2. Feature Engineering:

- Lag features and rolling statistics

- Weather variable integration

- Calendar and holiday effects

3. Model Ensemble:

- LSTM neural networks for complex patterns

- Prophet for seasonal trends

- ARIMA for statistical modeling

4. Uncertainty Quantification:

- Prediction intervals

- Confidence bounds

- Model agreement scoring

Output: 12-24 month forecasts with uncertainty bands

Technology Stack: PyTorch, Prophet, scikit-learn

Performance: <10% MAPE, <5 seconds inference time

**Data Architecture**

**Data Flow Diagram**

External Data Sources → Data Ingestion → Processing → Storage → Serving

│ │ │ │ │

▼ ▼ ▼ ▼ ▼

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

│Property Mgmt│ │ ETL │ │ ML Pipeline │ │ OLTP │ │ OLAP │

│Systems │────│ Service │───│ Processing │─│ DB │ │ DB │

│ │ │ │ │ │ │ │ │ │

│Utility Bills│ │Data Quality │ │Feature Store│ │ │ │ │

│Weather APIs │ │Validation │ │Model Store │ │ │ │ │

│Regulatory DB│ │ │ │ │ │ │ │ │

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

**Database Design**

**1. Operational Database (PostgreSQL)**

-- Core Domain Models

CREATE TABLE properties (

id UUID PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address JSONB,

property\_type property\_type\_enum,

floor\_area DECIMAL(10,2),

construction\_year INTEGER,

energy\_label VARCHAR(2),

created\_at TIMESTAMP DEFAULT NOW(),

updated\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE sustainability\_metrics (

id UUID PRIMARY KEY,

property\_id UUID REFERENCES properties(id),

metric\_type metric\_type\_enum,

value DECIMAL(15,4),

unit VARCHAR(50),

measurement\_date DATE,

source VARCHAR(100),

confidence\_score DECIMAL(3,2),

created\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE compliance\_assessments (

id UUID PRIMARY KEY,

property\_id UUID REFERENCES properties(id),

regulation\_type regulation\_enum,

assessment\_date DATE,

overall\_score DECIMAL(5,2),

status compliance\_status\_enum,

evidence JSONB,

created\_at TIMESTAMP DEFAULT NOW()

);

CREATE TABLE ml\_predictions (

id UUID PRIMARY KEY,

property\_id UUID REFERENCES properties(id),

model\_type VARCHAR(100),

prediction\_date DATE,

target\_date DATE,

predicted\_value DECIMAL(15,4),

confidence\_interval JSONB,

model\_version VARCHAR(50),

created\_at TIMESTAMP DEFAULT NOW()

);

-- Indexing Strategy

CREATE INDEX idx\_properties\_type\_location ON properties USING GIN (property\_type, address);

CREATE INDEX idx\_metrics\_property\_date ON sustainability\_metrics (property\_id, measurement\_date);

CREATE INDEX idx\_compliance\_property\_regulation ON compliance\_assessments (property\_id, regulation\_type);

CREATE INDEX idx\_predictions\_property\_model ON ml\_predictions (property\_id, model\_type);

**2. Analytics Database (PostgreSQL + TimescaleDB)**

-- Time-series optimization for large-scale analytics

CREATE TABLE time\_series\_metrics (

time TIMESTAMPTZ NOT NULL,

property\_id UUID NOT NULL,

metric\_name VARCHAR(100),

value DOUBLE PRECISION,

tags JSONB

);

-- Convert to hypertable for time-series optimization

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

-- Materialized views for common aggregations

CREATE MATERIALIZED VIEW monthly\_energy\_consumption AS

SELECT

DATE\_TRUNC('month', time) as month,

property\_id,

SUM(value) as total\_consumption,

AVG(value) as avg\_consumption

FROM time\_series\_metrics

WHERE metric\_name = 'energy\_consumption'

GROUP BY month, property\_id;

**Caching Strategy**

**Redis Cache Architecture**

Cache Layers:

1. Application Cache (L1):

- User sessions and authentication

- Frequently accessed property data

- API response caching

TTL: 15 minutes - 1 hour

2. ML Model Cache (L2):

- Prediction results

- Feature computations

- Model inference outputs

TTL: 24 hours

3. Report Cache (L3):

- Generated compliance reports

- Dashboard aggregations

- Export files

TTL: 7 days

Cache Invalidation:

- Time-based expiration (TTL)

- Event-driven invalidation

- Manual cache clearing for critical updates

**Security Architecture**

**Security Layers**

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

│ Security Architecture │

├─────────────────────────────────────────────────────────────┤

│ │

│ ┌─────────────────────────────────────────────────────────┐│

│ │ Network Security Layer ││

│ │ WAF │ DDoS Protection │ SSL/TLS │ VPN │ Firewall ││

│ └─────────────────────────────────────────────────────────┘│

│ │ │

│ ┌─────────────────────────────────────────────────────────┐│

│ │ Application Security Layer ││

│ │ Authentication │ Authorization │ Input Validation ││

│ └─────────────────────────────────────────────────────────┘│

│ │ │

│ ┌─────────────────────────────────────────────────────────┐│

│ │ Data Security Layer ││

│ │ Encryption │ Key Management │ Access Control │ Audit ││

│ └─────────────────────────────────────────────────────────┘│

│ │

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

**Authentication & Authorization**

**Identity Management**

Authentication:

- Multi-factor Authentication (MFA) required

- OAuth 2.0 / OpenID Connect integration

- JWT tokens with refresh mechanism

- Session management with secure cookies

Authorization:

- Role-Based Access Control (RBAC)

- Fine-grained permissions

- Resource-level access control

- Audit logging for all access attempts

Roles and Permissions:

Admin: Full system access, user management, system configuration

Compliance Manager: All compliance features, report generation, regulatory updates

ESG Analyst: Analytics, forecasting, performance analysis, data export

Portfolio Manager: Read-only dashboards, property overview, basic reports

API User: Programmatic access with scope-limited permissions

**Data Encryption**

Encryption at Rest:

- AES-256 encryption for database storage

- Encrypted file storage for documents

- Hardware Security Modules (HSM) for key storage

- Regular key rotation policies

Encryption in Transit:

- TLS 1.3 for all API communications

- Certificate pinning for mobile apps

- VPN tunnels for internal service communication

- End-to-end encryption for sensitive data flows

Key Management:

- Centralized key management service

- Automated key rotation

- Secure key backup and recovery

- Compliance with FIPS 140-2 standards

**Deployment Architecture**

**Cloud-Native Deployment**

**Container Orchestration**

Kubernetes Cluster Setup:

Node Configuration:

- Master Nodes: 3 instances for high availability

- Worker Nodes: Auto-scaling from 3 to 20 instances

- Node Types: Compute-optimized for app services, Memory-optimized for ML

Namespace Strategy:

- development: Development environment

- staging: Pre-production testing

- production: Live production environment

- monitoring: Observability stack

Resource Management:

- CPU limits and requests for all pods

- Memory limits with proper scaling policies

- Storage classes for different performance tiers

- Network policies for service isolation

**Service Mesh Architecture**

Istio Service Mesh:

Traffic Management:

- Intelligent load balancing

- Circuit breaker patterns

- Retry and timeout policies

- Canary deployments

Security:

- Mutual TLS between services

- Authorization policies

- Service-to-service authentication

- Traffic encryption

Observability:

- Distributed tracing

- Metrics collection

- Access logging

- Service topology visualization

**Multi-Environment Strategy**

**Environment Configuration**

Development Environment:

Purpose: Feature development and unit testing

Resources: Minimal resource allocation

Data: Synthetic test data and anonymized samples

Deployment: Manual or feature branch triggers

Staging Environment:

Purpose: Integration testing and pre-production validation

Resources: Production-like but smaller scale

Data: Masked production data for realistic testing

Deployment: Automated from main branch

Production Environment:

Purpose: Live system serving real users

Resources: Full resource allocation with auto-scaling

Data: Live production data with full encryption

Deployment: Blue-green deployment with approval gates

**Integration Architecture**

**External System Integration**

**Property Management System Integration**

Integration Patterns:

1. API-First Integration:

- REST APIs for real-time data sync

- Webhook notifications for data changes

- OAuth 2.0 authentication

- Rate limiting and retry mechanisms

2. Batch Processing:

- Scheduled ETL jobs for bulk data import

- File-based integration (CSV, Excel, XML)

- Error handling and data validation

- Processing status tracking

3. Event-Driven Integration:

- Message queue integration (Kafka/RabbitMQ)

- Event sourcing for audit trails

- Asynchronous processing

- Dead letter queue handling

Supported Systems:

- Yardi Voyager/Matrix: REST API integration

- MRI Software: API key authentication, JSON data format

- Argus Enterprise: SOAP/REST hybrid integration

- RealPage: OAuth 2.0, webhook notifications

- Buildium: REST API with pagination support

**Regulatory Database Integration**

EU Official Sources:

EUR-Lex Database:

- Daily automated scraping for new publications

- Document change detection algorithms

- Full-text search and categorization

- Multi-language support

ESMA Technical Standards:

- RSS feed monitoring

- PDF document processing

- Version comparison and diff generation

- Automated alert generation

National Regulatory Bodies:

- Country-specific database connections

- Regulatory calendar integration

- Local translation services

- Compliance deadline tracking

**Message Queue Architecture**

**Event-Driven Communication**

Apache Kafka Configuration:

Topics:

- property.created/updated/deleted

- document.uploaded/processed/failed

- compliance.calculated/updated

- regulatory.change.detected

- ml.prediction.completed

Partitioning Strategy:

- Partition by property\_id for related events

- Partition by user\_id for user-specific events

- Partition by region for geographic distribution

Consumer Groups:

- compliance-service: Process compliance-related events

- reporting-service: Generate reports and notifications

- ml-service: Trigger model retraining and predictions

- audit-service: Log all events for compliance tracking

**Monitoring and Observability**

**Three Pillars of Observability**

**1. Metrics Collection**

Application Metrics:

- Request latency (P50, P95, P99)

- Request rate and error rate

- Service availability and uptime

- Business metrics (properties processed, reports generated)

Infrastructure Metrics:

- CPU, memory, disk, network utilization

- Database connection pool status

- Queue depth and processing rate

- Container resource usage

Custom Metrics:

- ML model accuracy and drift detection

- Data quality scores

- Compliance calculation performance

- User engagement metrics

Technology Stack: Prometheus, Grafana, custom exporters

**2. Distributed Tracing**

Jaeger Tracing Setup:

Trace Collection:

- HTTP request tracing across all services

- Database query performance tracking

- ML pipeline execution tracing

- External API call monitoring

Performance Analysis:

- Service dependency mapping

- Bottleneck identification

- Error propagation tracking

- Performance regression detection

Business Process Tracing:

- End-to-end compliance calculation flows

- Document processing pipelines

- Report generation workflows

- User journey tracking

**3. Centralized Logging**

ELK Stack Configuration:

Elasticsearch:

- Multi-node cluster for high availability

- Index lifecycle management

- 90-day retention for application logs

- 7-year retention for audit logs

Logstash:

- Log parsing and enrichment

- Multi-format support (JSON, structured text)

- Sensitive data masking

- Error handling and dead letter queues

Kibana:

- Real-time log analysis dashboards

- Alert configuration and management

- User access control and role management

- Custom visualization and reporting

**Alerting Strategy**

**Alert Categories**

Critical Alerts (P1):

- Service downtime or unavailability

- Database connection failures

- Security breach indicators

- Data corruption detection

Response Time: 5 minutes

Escalation: Immediate PagerDuty notification

High Priority Alerts (P2):

- API response time > 5 seconds

- Error rate > 5%

- ML model accuracy degradation

- Compliance calculation failures

Response Time: 30 minutes

Escalation: Slack notification + email

Medium Priority Alerts (P3):

- Resource utilization > 80%

- Queue processing delays

- Data quality issues

- Integration failures

Response Time: 2 hours

Escalation: Email notification

Low Priority Alerts (P4):

- Performance degradation trends

- Capacity planning warnings

- Non-critical feature failures

Response Time: Next business day

Escalation: Dashboard notification

**Disaster Recovery and Business Continuity**

**Backup Strategy**

**Database Backup**

PostgreSQL Backup:

Full Backups: Daily at 2 AM UTC

Incremental Backups: Every 6 hours

Point-in-Time Recovery: 5-minute granularity

Retention: 30 days local, 1 year offsite

Testing: Monthly restore validation

Redis Backup:

RDB Snapshots: Every 6 hours

AOF Persistence: Enabled for durability

Retention: 7 days

Recovery Time: < 15 minutes

**File Storage Backup**

Document Storage:

Cross-Region Replication: Real-time to secondary region

Versioning: Enabled with 30-day retention

Lifecycle Policies: Move to cold storage after 1 year

Integrity Checks: Weekly checksum validation

Model Storage:

Version Control: Git-based model versioning

Artifact Storage: Cloud storage with replication

Backup Frequency: After each model training

Retention: All versions for 2 years

**High Availability Design**

**Multi-Region Architecture**

Primary Region (EU-West):

- Active production environment

- Read-write database instances

- Primary API endpoints

- Real-time processing

Secondary Region (EU-Central):

- Passive standby environment

- Read-only database replicas

- Backup API endpoints

- Disaster recovery ready

Failover Capabilities:

- Automated health checks every 30 seconds

- DNS-based traffic routing

- Database promotion procedures

- Application state synchronization

- RTO: 4 hours, RPO: 1 hour

**Performance Optimization**

**Caching Strategy**

**Multi-Level Caching**

Level 1 - Application Cache:

Technology: In-memory caching (Redis)

Content: API responses, computed metrics, user sessions

TTL: 15 minutes to 1 hour

Hit Ratio Target: >90%

Level 2 - Database Query Cache:

Technology: PostgreSQL query cache + Redis

Content: Complex aggregation results, dashboard data

TTL: 1-24 hours based on data freshness requirements

Hit Ratio Target: >80%

Level 3 - CDN Cache:

Technology: CloudFlare or AWS CloudFront

Content: Static assets, reports, documentation

TTL: 24 hours to 30 days

Hit Ratio Target: >95%

**Database Optimization**

**Query Performance**

Indexing Strategy:

- B-tree indexes for equality and range queries

- GIN indexes for JSONB and full-text search

- Partial indexes for filtered queries

- Composite indexes for multi-column queries

Query Optimization:

- Connection pooling (PgBouncer)

- Read replica distribution

- Query plan analysis and optimization

- Slow query monitoring and alerting

Partitioning:

- Time-based partitioning for metrics tables

- Hash partitioning for large datasets

- Automated partition management

- Partition pruning optimization

**Future Architecture Considerations**

**Scalability Roadmap**

**Phase 1 (Current): Single Region**

* Support for 10,000 properties
* 100 concurrent users
* Basic high availability
* Manual scaling

**Phase 2 (6-12 months): Multi-Region**

* Support for 50,000 properties
* 500 concurrent users
* Automated failover
* Auto-scaling capabilities

**Phase 3 (12-18 months): Global Scale**

* Support for 100,000+ properties
* 1,000+ concurrent users
* Multi-cloud deployment
* Edge computing integration

**Technology Evolution**

**Emerging Technologies**

Artificial Intelligence:

- Large Language Models for regulatory interpretation

- Computer vision advances for document processing

- Federated learning for privacy-preserving ML

- Automated machine learning (AutoML) pipelines

Cloud Native:

- Serverless computing for event processing

- Service mesh security and observability

- GitOps deployment strategies

- Infrastructure as Code maturity

Data Architecture:

- Real-time streaming analytics

- Data lake and lakehouse architectures

- Privacy-preserving analytics

- Blockchain for audit trails

**Architecture Decision Records (ADRs)**

**Key Architectural Decisions**

**ADR-001: Microservices vs Monolith**

**Decision**: Adopt microservices architecture **Rationale**:

* Independent scaling of AI/ML components
* Technology diversity for specialized services
* Team autonomy and development velocity
* Better fault isolation

**ADR-002: Database Technology Selection**

**Decision**: PostgreSQL as primary database **Rationale**:

* Strong ACID compliance for financial data
* Excellent JSON support for flexible schemas
* Rich ecosystem and expertise availability
* Time-series extensions (TimescaleDB) availability

**ADR-003: AI/ML Technology Stack**

**Decision**: Python-based ML stack with PyTorch **Rationale**:

* Rich ecosystem for NLP and computer vision
* Strong academic and research community support
* Excellent model deployment and serving options
* Team expertise and hiring considerations

**ADR-004: Cloud Provider Strategy**

**Decision**: Cloud-agnostic with initial AWS deployment **Rationale**:

* Avoid vendor lock-in for future flexibility
* Leverage containerization for portability
* AWS provides comprehensive service offerings
* European data residency requirements support

**Document Control:**

* Document ID: SA-001
* Version: 1.0
* Review Date: December 2025
* Dependencies: FR-001, TR-001
* Approval: Innovation Factory, University of Manchester