Award Nomination Application - Architecture Documentation

System Overview

The Award Nomination Application is a cloud-native, globally distributed web application built on Microsoft Azure, enabling employees to nominate colleagues for monetary awards with integrated fraud detection.

Architecture Components

1. Frontend Layer

Service: Azure Static Web Apps
Name: `award-nomination-frontend`
Technology: React SPA
Deployment: GitHub Actions CI/CD

Features:

- Single Page Application (SPA)
- OAuth2 authentication flow
- Mobile-responsive design
- Deployed to Azure edge locations globally

URL: `https://awards.terian-services.com`

---

2. API Gateway & Load Balancer

Service: Azure Front Door
Name: `Award-Nomination-ADF`
Location: Global (Microsoft Edge Network)

Features:

- SSL/TLS termination
- Web Application Firewall (WAF)
- Global load balancing
- Health probe monitoring
- Automatic failover between regions
- DDoS protection
- Route optimization

Routing Rules:

- `/api/*` → Backend Container Apps
- Health checks on both backends
- Automatic failover on 3 consecutive failures

---

3. Application Backend

Service: Azure Container Apps
Names:

- `award-api-eastus` (Primary)
- `award-api-westus` (Secondary/Failover)

Technology Stack:

- Python 3.11
- FastAPI framework
- Uvicorn ASGI server
- Docker containerized

Features:

- RESTful API endpoints
- JWT token validation
- Real-time fraud detection using ML
- Email notifications via SendGrid
- Role-based access control (RBAC)
- Admin impersonation with audit logging

Key Endpoints:

```
GET  /                          - Health check
GET  /health                    - Health status
GET  /docs                      - Swagger UI
GET  /whoami                    - Region diagnostic (admin only)
GET  /api/users                 - Get all users
POST /api/nominations/create    - Create nomination (with fraud scoring)
GET  /api/nominations/pending   - Get pending approvals
POST /api/nominations/approve   - Approve/reject nomination
GET  /api/nominations/history   - Get nomination history
GET  /api/admin/audit-logs      - Get impersonation logs (admin only)
GET  /api/admin/fraud-stats     - Fraud detection statistics (admin only)
```

Environment Variables:

```
SQL_SERVER=david64-sql.database.windows.net
SQL_DATABASE=AwardNominations
TENANT_ID=4d5f34d3-d97b-40c7-8704-edff856d3654
CLIENT_ID=4ab22340-6807-4bc4-a814-2722f21ba16d
SENDGRID_API_KEY=***
AZURE_STORAGE_ACCOUNT=awardnominationmodels
MODEL_CONTAINER=ml-models
```

---

4. Database Layer

Service: Azure SQL Database
Name: `AwardNominations@david64-sql`
Server: `david64-sql.database.windows.net`
Version: SQL Server 12.0

Tables:

- **Users** - Employee directory with manager hierarchy
- **Nominations** - Award nomination records with status tracking

- **FraudScores** - ML fraud detection scores per nomination
- **FraudAnalytics** - Detected fraud patterns and metrics
- **Impersonation_AuditLog** - Admin impersonation audit trail

Security:

- Firewall rules for Container App outbound IPs
- Encrypted connections (TLS 1.2+)
- SQL Authentication (dev) / Managed Identity (prod option)

---

5. ML Model Storage

Service: Azure Blob Storage
Account: `awardnominationmodels`
Container: `ml-models`

Contents:

- `fraud_detection_model.pkl` - Trained Random Forest classifier
- Model versioning support

Access:

- Container Apps use System-Assigned Managed Identity
- Role: Storage Blob Data Reader

---

6. Container Image Registry

Service: Azure Container Registry
Name: `acrawardnomination.azurecr.io`

Images:

- `award-nomination-api:latest` - Latest production backend
- `award-nomination-api:v1` - Version-tagged releases
- `award-nomination-api:<git-sha>` - Git commit-specific builds

Access:

- GitHub Actions uses Service Principal
- Container Apps pull with admin credentials

---

7. Identity & Authentication

Service: Microsoft Entra ID (Azure AD)
Tenant ID: `4d5f34d3-d97b-40c7-8704-edff856d3654`

App Registration:

- **Client ID:** `4ab22340-6807-4bc4-a814-2722f21ba16d`
- **Redirect URIs:** Configured for SWA and Swagger UI
- **API Permissions:** User.Read, custom API scope

Roles:

- `AWard_Nomination_Admin` - Full access + impersonation capability

- Default users - Create nominations, approve as manager

Authentication Flow:

1. User accesses SWA
2. Redirect to Microsoft login
3. OAuth2 authorization code flow
4. Return JWT token
5. Token validated on every API request

---

8. Email Service

Service: SendGrid API
Usage: Nomination approval notifications to managers

Email Triggers:

- New nomination created → Email to approving manager
- Includes nomination details and approval link

---

9. CI/CD Pipeline

Service: GitHub Actions

Workflows:

Frontend Deployment:

```
Trigger: Push to main (frontend/** changes)
Steps:
1. Build React app
2. Deploy to Azure Static Web Apps
3. Automatic cache invalidation
```

Backend Deployment:

```
Trigger: Push to main (backend/** changes)
Steps:
1. Login to Azure
2. Build Docker image
3. Tag with latest + git SHA
4. Push to ACR
5. Update award-api-eastus
6. Update award-api-westus
7. Health check verification
```

Secrets:

- `AZURE_CREDENTIALS` - Service Principal for deployment
- `AZURE_STATIC_WEB_APPS_API_TOKEN` - SWA deployment token

Data Flow Diagrams

User Creates Nomination

```
1. User submits nomination form in React SPA
2. React → POST /api/nominations/create via Azure Front Door
3. AFD routes to award-api-eastus (or westus)
4. FastAPI validates JWT token with Entra ID
5. FastAPI calls fraud_ml.get_fraud_assessment()
   ├─ Load ML model from Blob Storage (if not cached)
   ├─ Query historical data from SQL
   ├─ Calculate 20+ fraud features
   ├─ Run Random Forest prediction
   └─ Return fraud score & risk level
6. If CRITICAL risk → Block nomination
7. If acceptable risk → Insert into SQL Nominations table
8. Save fraud assessment to SQL FraudScores table
9. Get manager email from SQL
10. Send email via SendGrid API
11. Return success response to frontend
```

Manager Approves Nomination

```
1. Manager clicks approve in SWA
2. React → POST /api/nominations/approve
3. FastAPI validates manager is the approver
4. Update Nominations.Status = 'Approved'
5. Update Nominations.ApprovedDate = NOW()
6. Generate payroll extract CSV
7. Update Nominations.Status = 'Paid'
8. Return success response
```

Admin Impersonates User

```
1. Admin adds X-Impersonate-User header in Swagger UI
2. FastAPI validates admin has AWard_Nomination_Admin role
3. Log impersonation action to Impersonation_AuditLog
4. Execute API call as impersonated user
5. All queries use impersonated user context
6. All actions logged with admin + impersonated user
```

Security Architecture

Authentication & Authorization

- **Authentication:** OAuth2 / OpenID Connect via Entra ID
- **Authorization:** Role-based (RBAC) with JWT claims
- **Token Validation:** Every API request validates JWT signature
- **Admin Impersonation:** Logged to audit table with IP tracking

Network Security

- **Frontend:** HTTPS only, Azure CDN

- **API:** HTTPS only, TLS 1.2+
- **Database:** Encrypted connections, firewall rules
- **Blob Storage:** Private access, Managed Identity auth

Data Protection

- **In Transit:** TLS 1.2+ encryption everywhere
- **At Rest:** Azure SQL encryption, Blob encryption
- **Secrets:** Azure Key Vault (recommended) or Environment Variables
- **PII:** User data stored in SQL, access logged

Compliance

- **Audit Logging:** All admin actions logged
- **Fraud Detection:** ML-based anomaly detection
- **Access Control:** Manager approval required
- **Data Retention:** Configurable in SQL

High Availability & Disaster Recovery

High Availability

- **Frontend:** Global CDN, 99.95% SLA
- **API Gateway:** Azure Front Door multi-region, 99.99% SLA
- **Backend:** Dual-region deployment (East US + West US)
- **Database:** Azure SQL geo-replication available
- **Automatic Failover:** AFD health probes + auto-routing

Disaster Recovery

- **RTO (Recovery Time Objective):** < 5 minutes (automatic failover)
- **RPO (Recovery Point Objective):** Near-zero (active-active regions)
- **Backup Strategy:**
  - SQL: Automated backups (7-35 days retention)
  - ML Models: Versioned in Blob Storage
  - Code: GitHub source control

Monitoring

- **Application Insights:** API performance, errors, traces
- **Azure Monitor:** Container Apps metrics, health checks
- **Log Analytics:** Centralized logging
- **Alerts:** Set up for failures, high latency, fraud spikes

---

Fraud Detection System

Detection Layers

Layer 1: Rule-Based (SQL Stored Procedure)

- High frequency nominations (>50 in period)
- Repeated beneficiary pattern (>5 same person)
- Circular nominations (reciprocal schemes)
- Unusually high amounts (>2 std dev)

- Rapid approvals (<1 hour)
- Self-dealing networks (limited diversity)

Layer 2: Machine Learning (Python/scikit-learn)

- Random Forest classifier
- 20+ engineered features
- User behavior patterns
- Temporal analysis
- Relationship graphs
- Amount anomalies

Risk Levels:

- **CRITICAL (70-100):** Block nomination
- **HIGH (50-69):** Require manual review
- **MEDIUM (30-49):** Flag for monitoring
- **LOW (1-29):** Log and proceed
- **NONE (0):** Normal processing

Scalability

Current Capacity

- **Container Apps:** Auto-scale 0-10 replicas per region
- **Database:** DTU-based, can scale up/out
- **Blob Storage:** Unlimited, globally distributed

Growth Strategy

1. Increase Container Apps max replicas
2. Add more regions (Central US, North Europe, etc.)
3. Implement caching layer (Azure Redis)
4. Database read replicas for reporting
5. Event-driven architecture for async processing

Cost Optimization

Current Monthly Estimate

- **Static Web Apps:** ~$10/month (free tier available)
- **Front Door:** ~$50-100/month
- **Container Apps:** ~$100-200/month (2 regions)
- **Azure SQL:** ~$150-300/month (depends on DTU)
- **Blob Storage:** ~$5/month
- **Container Registry:** ~$5/month (Basic tier)
- **SendGrid:** Free tier (100 emails/day) or ~$15/month

Total: ~$320-620/month

Optimization Tips

- Use consumption-based pricing for Container Apps
- Implement caching to reduce database queries
- Archive old nominations to cheaper storage

- Use Azure Reserved Instances for predictable workloads

Deployment Instructions

Prerequisites

1. Azure subscription
2. GitHub account
3. Domain name (awards.terian-services.com)

Setup Steps

1. Create resource group: `rg_award_nomination`
2. Deploy Azure SQL Database
3. Create Container Registry
4. Create Container Apps (East + West)
5. Configure Azure Front Door
6. Deploy Static Web App
7. Configure Entra ID app registration
8. Set up GitHub Actions secrets
9. Push code to trigger deployment

Maintenance Tasks

Daily

- Monitor application health in Azure Portal
- Review fraud detection flags
- Check error logs

Weekly

- Review fraud detection dashboard
- Analyze performance metrics
- Check for failed deployments

Monthly

- Retrain fraud detection ML model
- Review and optimize database queries
- Update dependencies and security patches
- Audit impersonation logs

Quarterly

- Review architecture for optimization
- Update disaster recovery plan
- Security audit
- Cost analysis and optimization

Future Enhancements

1. **Real-time Notifications:** Azure SignalR for live updates
2. **Advanced Analytics:** Power BI dashboards

3. **Mobile App:** React Native app
4. **Batch Processing:** Azure Functions for scheduled tasks
5. **API Rate Limiting:** Azure API Management
6. **Enhanced ML:** Deep learning models, AutoML
7. **Multi-tenant:** Support for multiple organizations
8. **Reporting:** Automated monthly reports to executives

Support & Documentation

- **Architecture Diagrams:** This document + Mermaid diagrams
- **API Documentation:** Swagger UI at `/docs`
- **Source Code:** GitHub repository
- **Runbooks:** Deployment and troubleshooting guides
- **Contact:** IT Support / Cloud Operations team

Document Version: 1.0
Last Updated: January 29, 2026
Next Review: April 2026