🎉 SFG Aluminium Dashboard - Registration Document

Date: November 10, 2025
Version: v1.0.0
Status: ✅ **READY FOR REGISTRATION**

📋 Application Overview

Application Name: SFG Aluminium Dashboard
Purpose: Unified Application Inventory & Integration Portal
Type: Internal Dashboard & Integration Hub
Technology Stack: Next.js 14, TypeScript, Prisma, PostgreSQL, Shadcn UI

🔗 Application URLs

Production URL: To be deployed
GitHub Repository: https://github.com/sfgaluminium1-spec/sfg-app-portfolio
Location in Repo: `/satellites/sfg-aluminium-dashboard/`

📊 Capabilities

Core Features

- ✅ **Application Inventory Management** - Track all SFG applications
- ✅ **Integration Hub** - Centralized integration monitoring
- ✅ **Authentication System** - Secure login with NextAuth.js
- ✅ **GitHub Integration** - Self-registration capability
- ✅ **Webhook Management** - Real-time event processing
- ✅ **Message Handlers** - Bi-directional communication

Technical Capabilities

- ✅ Next.js 14 with App Router and TypeScript
- ✅ Prisma ORM with PostgreSQL database
- ✅ Shadcn UI component library
- ✅ Tailwind CSS styling
- ✅ GitHub API integration
- ✅ Webhook and message handler endpoints

🔌 Integration Endpoints

Webhook Endpoint

```
POST /api/webhooks/github
```

Purpose: Receive events from GitHub and other services
Authentication: HMAC signature verification
Supported Events:
- `repository.created`
- `repository.updated`
- `pull_request.opened`
- `pull_request.merged`
- `issues.opened`
- `issues.closed`

Message Handler Endpoint

```
POST /api/messages/handle
```

Purpose: Handle incoming messages from NEXUS and other apps
Authentication: API key in header
Supported Actions:
- `query.app_list` - Get list of registered apps
- `query.app_details` - Get specific app details
- `action.register_app` - Register new application
- `action.update_app` - Update app information

Health Check Endpoint

```
GET /api/health
```

Purpose: Monitor application health
Authentication: None
Response: `{ "status": "ok", "version": "1.0.0" }`

Registration API

```
POST /api/registration/execute
```

Purpose: Execute self-registration in GitHub portfolio
Authentication: Session-based
Payload: Registration metadata and app details

📤 Outbound Events

Events the Dashboard emits:

- `app.registered` - New app registered in dashboard
- `app.updated` - App information updated
- `integration.connected` - New integration established
- `webhook.received` - Webhook event processed
- `message.handled` - Message successfully processed

📥 Inbound Events

Events the Dashboard can handle:

- `query.app_list` - Return list of all registered apps
- `query.app_details` - Return details for specific app
- `query.integration_status` - Check integration health
- `action.register_app` - Register new app in dashboard
- `action.update_app` - Update app information
- `action.test_webhook` - Test webhook connectivity

🔐 Authentication & Security

Authentication Methods

- **NextAuth.js** - Primary authentication system
- **Credentials Provider** - Email/password authentication
- **Session Management** - Secure session handling
- **API Key** - For service-to-service communication
- **HMAC Signatures** - For webhook verification

Security Features

- ✅ HTTPS enforced
- ✅ CORS configured
- ✅ Rate limiting implemented
- ✅ Input validation
- ✅ SQL injection protection (Prisma ORM)
- ✅ XSS protection

💾 Data Models

Application

```
{
  id: string
  name: string
  description: string
  baseUrl: string
  authMethod: string
  webhookUrl: string
  messageHandlerUrl: string
  apiKey: string
  status: 'active' | 'inactive' | 'maintenance'
  createdAt: Date
  updatedAt: Date
}
```

Integration

```
{
  id: string
  sourceAppId: string
  targetAppId: string
  type: 'webhook' | 'api' | 'mcp'
  status: 'connected' | 'disconnected' | 'error'
  lastSyncAt: Date
  createdAt: Date
}
```

User

```
{
  id: string
  email: string
  password: string (hashed)
  name: string
  role: 'admin' | 'user'
  createdAt: Date
  updatedAt: Date
}
```

🎯 Business Rules

Application Management

- ✅ Each app must have unique name
- ✅ Base URL must be valid HTTPS URL
- ✅ Webhook and message handler URLs validated
- ✅ API keys generated securely
- ✅ Status changes logged

Integration Rules

- ✅ Integrations tested before activation
- ✅ Failed integrations auto-retry (3 attempts)
- ✅ Integration health checked every 5 minutes
- ✅ Stale integrations flagged after 24 hours

📋 Required Integrations

GitHub

- **Purpose:** Self-registration and repository management
- **Authentication:** GitHub App / OAuth
- **Endpoints Used:**
- Create issues
- Update repository files
- Manage webhooks

NEXUS (Future)

- **Purpose:** Central orchestration hub
- **Authentication:** API key
- **Events:** Bi-directional webhook and message communication

MCP Servers (Future)

- **MCP-SALES** - Sales and CRM integration
- **MCP-FINANCE** - Financial systems integration
- **MCP-OPERATIONS** - Operations management
- **MCP-COMMUNICATIONS** - Notifications
- **MCP-DATA** - Analytics and reporting

🔧 Configuration

Environment Variables

```
# Database
DATABASE_URL="postgresql://..."

# NextAuth
NEXTAUTH_SECRET="..."
NEXTAUTH_URL="https://..."

# GitHub
GITHUB_APP_ID="..."
GITHUB_PRIVATE_KEY="..."
GITHUB_INSTALLATION_ID="..."
GITHUB_CLIENT_ID="..."
GITHUB_CLIENT_SECRET="..."

# Application
NODE_ENV="production"
API_KEY="..."
```

📊 Monitoring & Logging

Health Checks

- **Endpoint:** `/api/health`
- **Frequency:** Every 60 seconds
- **Metrics:** Response time, database connectivity, memory usage

Logging

- **Level:** INFO, WARN, ERROR
- **Format:** JSON structured logs
- **Retention:** 30 days
- **Location:** `/app/.logs/`

Error Tracking

- **Failed webhooks** - Logged and retried
- **Failed messages** - Queued for retry
- **Integration errors** - Alerting enabled

🚀 Deployment

Requirements

- Node.js 18+
- PostgreSQL 14+
- 2GB RAM minimum
- HTTPS certificate

Build Command

```
npm install
npm run build
```

Start Command

```
npm start
```

Database Migration

```
npx prisma migrate deploy
npx prisma generate
```

📈 Version History

v1.0.0 (November 10, 2025) - Current

- Initial dashboard implementation
- GitHub integration complete
- Self-registration system
- Webhook and message handlers
- Authentication system
- Application inventory management

🎬 Next Steps

1. **Deploy Application**
   - Set up production environment
   - Configure environment variables
   - Run database migrations
   - Deploy to hosting platform

2. **Test Integration Endpoints**
   - Test webhook: `POST /api/webhooks/github`
   - Test message handler: `POST /api/messages/handle`
   - Test health check: `GET /api/health`
   - Test registration: `POST /api/registration/execute`

3. **Configure GitHub Webhooks**
   - Set webhook URL in GitHub repository settings
   - Configure events to receive
   - Test webhook delivery

4. **Connect to NEXUS**
   - Register in NEXUS orchestration hub

   - Configure bi-directional communication
   - Test end-to-end workflow

5. **Monitor & Optimize**
   - Set up monitoring dashboards
   - Configure alerts
   - Optimize performance
   - Review logs regularly

📞 Contact Information

Project Owner: Warren (SFG Director)
Email: warren@sfgaluminium.co.uk
Repository: https://github.com/sfgaluminium1-spec/sfg-app-portfolio
Location: `/satellites/sfg-aluminium-dashboard/`

✅ Registration Checklist

- ✅ **Code Complete** - All features implemented
- ✅ **Documentation** - Registration document created
- ✅ **Integration Endpoints** - Webhooks and message handlers ready
- ✅ **Authentication** - Security configured
- ✅ **Database Schema** - Prisma models defined
- ⏳ **Deployment** - Pending production deployment
- ⏳ **GitHub Webhooks** - To be configured post-deployment
- ⏳ **NEXUS Registration** - Awaiting deployment URL
- ⏳ **Testing** - End-to-end testing pending

Status: ✅ **READY FOR GITHUB UPLOAD**

Generated: November 10, 2025
By: DeepAgent
For: SFG Aluminium Ltd