Testing Guide - SFG Aluminium Dashboard

Version: 1.0.0
Date: November 10, 2025
Status: Ready for Testing

Overview

This guide provides comprehensive testing procedures for the SFG Aluminium Dashboard, including webhook integration, message handlers, authentication, and end-to-end workflows.

Prerequisites

- Dashboard deployed to production
- Production URL available
- GitHub webhooks configured
- Test credentials created
- Database accessible

1. Health Check Testing

Test Application Health

Endpoint: `GET /api/health`

Request:

```
curl https://[YOUR-DASHBOARD-URL]/api/health
```

Expected Response:

```json
{
  "status": "ok",
  "version": "1.0.0",
  "timestamp": "2025-11-10T21:00:00Z"
}
```

Success Criteria:
- ✅ Status code: 200
- ✅ Response contains version
- ✅ Response time < 500ms

2. Authentication Testing

Test Login Flow

Endpoint: `POST /api/auth/[...nextauth]`

Test Account:

```
{
  "email": "test@sfgaluminium.co.uk",
  "password": "TestPassword123!"
}
```

Steps:
1. Navigate to `https://[YOUR-DASHBOARD-URL]/login`
2. Enter test credentials
3. Click “Sign In”
4. Verify redirect to dashboard

Success Criteria:
- ✅ Login successful
- ✅ Session created
- ✅ Redirect to dashboard
- ✅ User information displayed

Test Session Management

Endpoint: `GET /api/auth/session`

Request:

```
curl -H "Cookie: next-auth.session-token=TOKEN" \
  https://[YOUR-DASHBOARD-URL]/api/auth/session
```

Expected Response:

```
{
  "user": {
    "email": "test@sfgaluminium.co.uk",
    "name": "Test User"
  },
  "expires": "2025-11-17T21:00:00Z"
}
```

Success Criteria:
- ✅ Session valid
- ✅ User data returned
- ✅ Expiry date present

3. Webhook Testing

Test Webhook Endpoint

Endpoint: `POST /api/webhooks/github`

Test Script:

```bash
#!/bin/bash

DASHBOARD_URL="https://[YOUR-DASHBOARD-URL]"
WEBHOOK_SECRET="your-webhook-secret"

# Test payload
PAYLOAD='{"action":"opened","repository":{"name":"test-repo"}}'

# Calculate signature
SIGNATURE="sha256=$(echo -n "$PAYLOAD" | openssl dgst -sha256 -hmac "$WEBHOOK_SECRET"
| sed 's/^.* //')"

# Send webhook
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "Content-Type: application/json" \
  -H "X-GitHub-Event: ping" \
  -H "X-GitHub-Delivery: test-delivery-$(date +%s)" \
  -H "X-Hub-Signature-256: ${SIGNATURE}" \
  -d "$PAYLOAD"
```

Expected Response:

```json
{
  "success": true,
  "message": "Webhook processed successfully",
  "event": "ping",
  "timestamp": "2025-11-10T21:00:00Z"
}
```

Success Criteria:
- ✅ Status code: 200
- ✅ Signature verified
- ✅ Event processed
- ✅ Database recorded

Test GitHub Webhook Events

Repository Event:

```bash
# Test repository.created event
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "Content-Type: application/json" \
  -H "X-GitHub-Event: repository" \
  -H "X-GitHub-Delivery: test-$(date +%s)" \
  -H "X-Hub-Signature-256: ${SIGNATURE}" \
  -d '{
    "action": "created",
    "repository": {
      "name": "new-test-repo",
      "full_name": "sfgaluminium1-spec/new-test-repo",
      "private": false,
      "description": "Test repository"
    }
  }'
```

Issue Event:

```bash
# Test issues.opened event
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "Content-Type: application/json" \
  -H "X-GitHub-Event: issues" \
  -H "X-GitHub-Delivery: test-$(date +%s)" \
  -H "X-Hub-Signature-256: ${SIGNATURE}" \
  -d '{
    "action": "opened",
    "issue": {
      "number": 1,
      "title": "Test Issue",
      "body": "This is a test issue"
    },
    "repository": {
      "name": "test-repo"
    }
  }'
```

Pull Request Event:

```bash
# Test pull_request.opened event
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "Content-Type: application/json" \
  -H "X-GitHub-Event: pull_request" \
  -H "X-GitHub-Delivery: test-$(date +%s)" \
  -H "X-Hub-Signature-256: ${SIGNATURE}" \
  -d '{
    "action": "opened",
    "pull_request": {
      "number": 1,
      "title": "Test PR",
      "state": "open"
    },
    "repository": {
      "name": "test-repo"
    }
  }'
```

---

4. Message Handler Testing

Test Message Endpoint

Endpoint: `POST /api/messages/handle`

Query Message:

```
curl -X POST https://[YOUR-DASHBOARD-URL]/api/messages/handle \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your-api-key" \
  -d '{
    "type": "query",
    "action": "app_list",
    "timestamp": "2025-11-10T21:00:00Z"
  }'
```

Expected Response:

```
{
  "success": true,
  "type": "query_response",
  "data": {
    "apps": [
      {
        "id": "1",
        "name": "SFG Website",
        "status": "active"
      }
    ]
  },
  "timestamp": "2025-11-10T21:00:00Z"
}
```

Action Message:

```
curl -X POST https://[YOUR-DASHBOARD-URL]/api/messages/handle \
  -H "Content-Type: application/json" \
  -H "X-API-Key: your-api-key" \
  -d '{
    "type": "action",
    "action": "register_app",
    "data": {
      "name": "Test App",
      "baseUrl": "https://test.example.com",
      "authMethod": "api_key"
    },
    "timestamp": "2025-11-10T21:00:00Z"
  }'
```

Success Criteria:
- ✅ Status code: 200
- ✅ API key validated
- ✅ Message processed
- ✅ Response returned

5. Registration API Testing

Test Self-Registration

Endpoint: `POST /api/registration/execute`

Request:

```
curl -X POST https://[YOUR-DASHBOARD-URL]/api/registration/execute \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=TOKEN" \
  -d '{
    "appName": "SFG Aluminium Dashboard",
    "baseUrl": "https://[YOUR-DASHBOARD-URL]",
    "webhookUrl": "https://[YOUR-DASHBOARD-URL]/api/webhooks/github",
    "messageHandlerUrl": "https://[YOUR-DASHBOARD-URL]/api/messages/handle"
  }'
```

Expected Response:

```
{
  "success": true,
  "message": "Registration successful",
  "issueNumber": 59,
  "issueUrl": "https://github.com/sfgaluminium1-spec/sfg-app-portfolio/issues/59"
}
```

Success Criteria:
- ✅ Registration successful
- ✅ GitHub issue created/updated
- ✅ Database updated
- ✅ Response contains issue URL

---

6. Database Testing

Verify Webhook Events Recording

```
-- Check webhook events
SELECT
  id,
  "eventType",
  "deliveryId",
  status,
  "processedAt"
FROM "WebhookEvent"
ORDER BY "processedAt" DESC
LIMIT 10;
```

Expected Results:
- ✅ Events recorded in database
- ✅ Correct event types

- ✅ Status marked as success
- ✅ Timestamps present

Verify Application Records

```sql
-- Check registered applications
SELECT
  id,
  name,
  "baseUrl",
  status,
  "createdAt"
FROM "Application"
ORDER BY "createdAt" DESC;
```

Expected Results:
- ✅ Dashboard application present
- ✅ Status is active
- ✅ URLs correct

---

7. Integration Testing

Test GitHub Integration

Steps:
1. Create a test issue in GitHub repository
2. Verify webhook is received by dashboard
3. Check database for recorded event
4. Verify event appears in dashboard UI

Success Criteria:
- ✅ Webhook received within 5 seconds
- ✅ Event recorded in database
- ✅ Event visible in dashboard
- ✅ No errors in logs

Test Message Communication

Steps:
1. Send query message to dashboard
2. Verify response received
3. Send action message to register app
4. Verify app appears in database

Success Criteria:
- ✅ Query response received
- ✅ Action executed successfully
- ✅ Database updated
- ✅ No errors in logs

---

8. Performance Testing

Load Testing

Tool: Apache Bench

```
# Test health endpoint
ab -n 1000 -c 10 https://[YOUR-DASHBOARD-URL]/api/health

# Test webhook endpoint (with valid signature)
ab -n 100 -c 5 -p webhook-payload.json \
  -H "X-GitHub-Event: ping" \
  -H "X-Hub-Signature-256: SIGNATURE" \
  https://[YOUR-DASHBOARD-URL]/api/webhooks/github
```

Success Criteria:
- ✅ Response time < 500ms (average)
- ✅ Success rate > 99%
- ✅ No 5xx errors
- ✅ Database handles concurrent requests

Stress Testing

```
# Concurrent webhooks
for i in {1..50}; do
  curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
    -H "Content-Type: application/json" \
    -H "X-GitHub-Event: ping" \
    -H "X-Hub-Signature-256: ${SIGNATURE}" \
    -d "$PAYLOAD" &
done
wait
```

Success Criteria:
- ✅ All requests processed
- ✅ No timeouts
- ✅ Database consistency maintained
- ✅ Memory usage stable

---

9. Security Testing

Test Signature Verification

Invalid Signature:

```
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "Content-Type: application/json" \
  -H "X-GitHub-Event: ping" \
  -H "X-Hub-Signature-256: sha256=invalid" \
  -d '{"test":"data"}'
```

Expected Response:

```
{
  "error": "Invalid signature",
  "status": 401
}
```

Success Criteria:
- ✅ Status code: 401
- ✅ Request rejected
- ✅ Error message clear
- ✅ Event not processed

Test API Key Validation

Missing API Key:

```
curl -X POST "${DASHBOARD_URL}/api/messages/handle" \
  -H "Content-Type: application/json" \
  -d '{"type":"query","action":"app_list"}'
```

Expected Response:

```
{
  "error": "Unauthorized",
  "status": 401
}
```

Success Criteria:
- ✅ Status code: 401
- ✅ Request rejected
- ✅ Message not processed

---

10. End-to-End Testing

Complete Workflow Test

Scenario: Register new app via GitHub issue

Steps:
1. Create GitHub issue with registration label
2. Webhook triggers dashboard
3. Dashboard processes registration
4. App appears in inventory
5. Integration endpoints tested
6. Status updated in dashboard

Test Script:

```bash
#!/bin/bash

echo "Starting end-to-end test..."

# 1. Health check
echo "1. Testing health endpoint..."
curl -s "${DASHBOARD_URL}/api/health"

# 2. Login
echo "2. Testing authentication..."
# (login and get session token)

# 3. Create registration
echo "3. Testing registration..."
curl -X POST "${DASHBOARD_URL}/api/registration/execute" \
  -H "Cookie: session-token" \
  -d '{"appName":"Test App",...}'

# 4. Trigger webhook
echo "4. Testing webhook..."
curl -X POST "${DASHBOARD_URL}/api/webhooks/github" \
  -H "X-GitHub-Event: issues" \
  -d '{"action":"opened",...}'

# 5. Query apps
echo "5. Testing message handler..."
curl -X POST "${DASHBOARD_URL}/api/messages/handle" \
  -H "X-API-Key: key" \
  -d '{"type":"query","action":"app_list"}'

echo "End-to-end test complete!"
```

Success Criteria:
- ✅ All steps complete successfully
- ✅ No errors encountered
- ✅ Data consistent across systems
- ✅ Complete workflow in < 10 seconds

---

11. Monitoring & Logging

Check Application Logs

```bash
# View recent logs
tail -f /app/.logs/application.log

# Search for errors
grep ERROR /app/.logs/application.log

# Search for specific event
grep "webhook.received" /app/.logs/application.log
```

Monitor Database

```sql
-- Check recent activity
SELECT * FROM "WebhookEvent"
WHERE "processedAt" > NOW() - INTERVAL '1 hour'
ORDER BY "processedAt" DESC;

-- Check error rates
SELECT status, COUNT(*)
FROM "WebhookEvent"
GROUP BY status;
```

---

12. Troubleshooting

Common Issues

Issue: Webhook not received

Solution:
1. Check webhook is active in GitHub
2. Verify URL is correct and accessible
3. Check SSL certificate is valid
4. Review GitHub delivery logs

Issue: Signature verification fails

Solution:
1. Verify webhook secret matches in both locations
2. Check secret has no extra spaces
3. Ensure using raw request body for verification

Issue: Database connection errors

Solution:
1. Verify DATABASE_URL is correct
2. Check database is running and accessible
3. Verify credentials are valid
4. Check connection pool settings

---

13. Test Checklist

Pre-Deployment

- [ ] All unit tests pass
- [ ] All integration tests pass
- [ ] Code review complete
- [ ] Documentation updated

Post-Deployment

- [ ] Health check passing

- [ ] Authentication working
- [ ] Webhooks configured
- [ ] Message handlers responding
- [ ] Database accessible
- [ ] Logs being written
- [ ] Monitoring active

Functional Tests

- [ ] Login/logout works
- [ ] Dashboard loads correctly
- [ ] App registration works
- [ ] Webhook events received
- [ ] Messages processed
- [ ] Database updates correctly

Performance Tests

- [ ] Response time < 500ms
- [ ] Concurrent requests handled
- [ ] No memory leaks
- [ ] Database queries optimized

Security Tests

- [ ] Signature verification works
- [ ] API key validation works
- [ ] Unauthorized access blocked
- [ ] Input validation active
- [ ] HTTPS enforced

📞 Support

Issues or Questions:
- Create issue in repository
- Contact: warren@sfgaluminium.co.uk

Testing Resources:
- GitHub Webhooks: https://docs.github.com/webhooks
- Next.js Testing: https://nextjs.org/docs/testing
- Prisma Testing: https://www.prisma.io/docs/guides/testing

Document Version: 1.0
Last Updated: November 10, 2025
Status: Ready for Use