STEP 23: Activity Log & Audit Trail System - Testing Guide

Last Updated: November 30, 2025
Test Coverage: 30+ Test Scenarios
Status: Ready for Testing

Prerequisites

1. Database Setup

```
cd /home/ubuntu/fyndr/nextjs_space
npx prisma generate
npx prisma db push
```

2. Start Development Server

```
npm run dev
```

3. Test Data Setup

- Create at least 2 test RFPs
- Invite at least 2 suppliers to each RFP
- Ensure suppliers have access tokens

4. Browser Setup

- Open browser in incognito mode for clean session testing
- Have two browser windows (one for buyer, one for supplier)

Test Scenarios

Category 1: RFP Events (3 Tests)

Test 1.1: RFP Creation Logging

Objective: Verify `RFP_CREATED` event is logged when creating an RFP

Steps:
1. Log in as buyer (john@doe.com)
2. Navigate to `/dashboard/rfps/new`
3. Fill in RFP form:
- Title: “Test RFP for Activity Log”
- Description: “Testing activity logging”
- Company: Select any

- Supplier: Select any
- Budget: 50000
- Priority: HIGH
4. Click “Create RFP”

Expected Results:
- ✅ RFP created successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “RFP_CREATED”
- `actorRole`: “BUYER”
- `summary`: “RFP "Test RFP for Activity Log" created”
- `rfpId`: Set
- `userId`: Set
- `ipAddress`: Set
- `userAgent`: Set
- `details`: Contains rfpId, title, status, companyName, supplierName

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'RFP_CREATED'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 1.2: RFP Update Logging

Objective: Verify `RFP_UPDATED` event is logged when updating an RFP

Steps:
1. Navigate to RFP detail page
2. Click “Edit”
3. Update title to “Updated Test RFP”
4. Update budget to 60000
5. Click “Save”

Expected Results:
- ✅ RFP updated successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “RFP_UPDATED”
- `actorRole`: “BUYER”
- `summary`: “RFP "Updated Test RFP" updated”
- `details.updatedFields`: Array includes “title”, “budget”
- `details.stageChanged`: false

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'RFP_UPDATED'
AND "rfpId" = '<your-rfp-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 1.3: Timeline Update Logging

Objective: Verify `RFP_TIMELINE_UPDATED` event is logged separately

Steps:
1. Edit same RFP
2. Update timeline fields:
- Ask Questions Start: Tomorrow
- Ask Questions End: In 7 days
- Submission Start: In 8 days
- Submission End: In 15 days
3. Click “Save”

Expected Results:
- ✅ Both `RFP_UPDATED` and `RFP_TIMELINE_UPDATED` events logged
- ✅ `RFP_TIMELINE_UPDATED` contains:
- `summary`: “RFP timeline updated”
- `details.timelineFields`: All timeline dates included

Verification:

```
SELECT "eventType", "summary", "details"
FROM "ActivityLog"
WHERE "rfpId" = '<your-rfp-id>'
AND "eventType" IN ('RFP_UPDATED', 'RFP_TIMELINE_UPDATED')
ORDER BY "createdAt" DESC LIMIT 2;
```

Category 2: Supplier Portal Events (2 Tests)

Test 2.1: Supplier Invitation Logging

Objective: Verify `SUPPLIER_INVITATION_SENT` event is logged

Steps:
1. Navigate to RFP detail page
2. Scroll to “Supplier Contacts” section
3. Click “Invite Supplier”
4. Fill in form:
- Name: “Test Supplier”
- Email: “test.supplier@example.com”
- Organization: “Test Corp”
5. Click “Send Invitation”

Expected Results:
- ✅ Invitation sent successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_INVITATION_SENT”
- `actorRole`: “BUYER”
- `summary`: “Invitation sent to test.supplier@example.com”
- `supplierContactId`: Set
- `details`: Contains email, name, organization

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_INVITATION_SENT'
AND "rfpId" = '<your-rfp-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 2.2: Supplier Login Logging

Objective: Verify `SUPPLIER_PORTAL_LOGIN` event is logged

Steps:
1. Copy magic link from invitation email (or database)
2. Open magic link in incognito window
3. Observe auto-login process

Expected Results:
- ✅ Supplier logged in successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_PORTAL_LOGIN”
- `actorRole`: “SUPPLIER”
- `summary`: “Supplier Test Supplier logged in”
- `supplierContactId`: Set
- `userId`: Set (supplier user ID)
- `details`: Contains email, name

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_PORTAL_LOGIN'
AND "supplierContactId" = '<supplier-contact-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Category 3: Supplier Response Events (4 Tests)

Test 3.1: Response Draft Save Logging

Objective: Verify `SUPPLIER_RESPONSE_SAVED_DRAFT` event is logged

Steps:
1. As supplier, navigate to RFP page
2. Fill in “Executive Summary” field
3. Click “Save Draft”

Expected Results:
- ✅ Draft saved successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_RESPONSE_SAVED_DRAFT”
- `actorRole`: “SUPPLIER”

- `summary`: “Response draft saved”
- `supplierResponseId`: Set
- `supplierContactId`: Set

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_RESPONSE_SAVED_DRAFT'
AND "supplierContactId" = '<supplier-contact-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 3.2: Response Submission Logging

Objective: Verify `SUPPLIER_RESPONSE_SUBMITTED` event is logged

Steps:
1. Complete response form
2. Click “Submit Final Response”
3. Confirm submission in modal

Expected Results:
- ✅ Response submitted successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_RESPONSE_SUBMITTED”
- `actorRole`: “SUPPLIER”
- `summary`: “Response submitted by “
- `supplierResponseId`: Set
- `details.attachmentCount`: Included

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_RESPONSE_SUBMITTED'
AND "supplierResponseId" = '<response-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 3.3: Attachment Upload Logging

Objective: Verify `SUPPLIER_ATTACHMENT_UPLOADED` event is logged

Steps:
1. In response form, click “Upload Attachment”
2. Select a PDF file
3. Upload completes

Expected Results:
- ✅ Attachment uploaded successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_ATTACHMENT_UPLOADED”
- `actorRole`: “SUPPLIER”

- `summary`: “Attachment ‘’ uploaded”
- `details.fileName`: Set
- `details.fileSize`: Set

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_ATTACHMENT_UPLOADED'
AND "rfpId" = '<rfp-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Test 3.4: Attachment Deletion Logging

Objective: Verify `SUPPLIER_ATTACHMENT_DELETED` event is logged

Steps:
1. Click “Delete” on an attachment
2. Confirm deletion

Expected Results:
- ✅ Attachment deleted successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_ATTACHMENT_DELETED”
- `actorRole`: “SUPPLIER”
- `summary`: “Attachment ‘’ deleted”
- `details.fileName`: Set

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'SUPPLIER_ATTACHMENT_DELETED'
AND "rfpId" = '<rfp-id>'
ORDER BY "createdAt" DESC LIMIT 1;
```

Category 4: AI / Comparison Events (5 Tests)

Test 4.1: AI Extraction Run Logging

Objective: Verify `AI_EXTRACTION_RUN` event is logged

Steps:
1. As buyer, navigate to RFP with submitted response
2. Click “Run AI Extraction”
3. Wait for completion

Expected Results:
- ✅ Extraction completed successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “AI_EXTRACTION_RUN”
- `actorRole`: “BUYER” or “SYSTEM”

- `summary`: “AI extraction completed for responses”
- `details.responseCount`: Set

Test 4.2: Supplier Comparison Run Logging

Objective: Verify `SUPPLIER_COMPARISON_RUN` event is logged

Steps:
1. Navigate to `/dashboard/rfps/[id]/compare`
2. Click “Run Comparison”
3. Wait for completion

Expected Results:
- ✅ Comparison completed successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_COMPARISON_RUN”
- `actorRole`: “BUYER”
- `summary`: “Supplier comparison completed”
- `details.supplierCount`: Set

Test 4.3: AI Summary Generation Logging

Objective: Verify `COMPARISON_AI_SUMMARY_RUN` event is logged

Steps:
1. On comparison page, click “Generate AI Summary”
2. Wait for AI to complete

Expected Results:
- ✅ Summary generated successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “COMPARISON_AI_SUMMARY_RUN”
- `actorRole`: “BUYER”
- `summary`: “AI comparison summary generated”

Test 4.4: Narrative Generation Logging

Objective: Verify `COMPARISON_NARRATIVE_GENERATED` event is logged

Steps:
1. Click “Generate Narrative”
2. Wait for completion

Expected Results:
- ✅ Narrative generated successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “COMPARISON_NARRATIVE_GENERATED”
- `actorRole`: “BUYER”
- `summary`: “Comparison narrative generated”

Test 4.5: Report Generation Logging

Objective: Verify `COMPARISON_REPORT_GENERATED` event is logged

Steps:
1. Click “Generate Report”
2. Wait for PDF generation

Expected Results:
- ✅ PDF generated successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “COMPARISON_REPORT_GENERATED”
- `actorRole`: “BUYER”
- `summary`: “Comparison report PDF generated”
- `details.reportUrl`: Set

Category 5: Q&A Events (3 Tests)

Test 5.1: Question Creation Logging

Objective: Verify `SUPPLIER_QUESTION_CREATED` event is logged

Steps:
1. As supplier, navigate to “Questions & Answers”
2. Type a question: “What is the project timeline?”
3. Click “Submit Question”

Expected Results:
- ✅ Question submitted successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_QUESTION_CREATED”
- `actorRole`: “SUPPLIER”
- `summary`: “Supplier asked a question”
- `details.questionId`: Set

Test 5.2: Question Answered Logging

Objective: Verify `SUPPLIER_QUESTION_ANSWERED` event is logged

Steps:
1. As buyer, navigate to “Supplier Questions & Answers”
2. Click “Answer” on pending question
3. Type answer and submit
4. Do NOT check “Broadcast”

Expected Results:
- ✅ Answer saved successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_QUESTION_ANSWERED”

- `actorRole`: “BUYER”
- `summary`: “Question answered by buyer”
- `details.broadcast`: false

Test 5.3: Broadcast Creation Logging

Objective: Verify `SUPPLIER_BROADCAST_CREATED` event is logged

Steps:
1. Answer a question
2. CHECK “Broadcast this answer to ALL suppliers”
3. Submit

Expected Results:
- ✅ Broadcast sent successfully
- ✅ `ActivityLog` record created with:
- `eventType`: “SUPPLIER_BROADCAST_CREATED”
- `actorRole`: “BUYER”
- `summary`: “Broadcast message sent to all suppliers”

Category 6: Buyer Activity UI (6 Tests)

Test 6.1: View Activity Page

Objective: Verify activity page displays logs correctly

Steps:
1. Navigate to `/dashboard/rfps/[id]/activity`

Expected Results:
- ✅ Page loads successfully
- ✅ Activity logs displayed in reverse chronological order
- ✅ Event type badges color-coded correctly:
- Blue: RFP events
- Green: Supplier events
- Purple: AI events
- Amber: Q&A events
- ✅ Timestamps shown (e.g., “2 hours ago”)
- ✅ Actor role indicators visible

Test 6.2: Filter by Event Type

Objective: Verify event type filter works

Steps:
1. Select “Supplier Response Events” from filter dropdown
2. Observe results

Expected Results:
- ✅ Only response-related events shown
- ✅ Other event types hidden
- ✅ Results update without page reload

Test 6.3: Filter by Actor Role

Objective: Verify actor role filter works

Steps:
1. Select “SUPPLIER” from actor role filter
2. Observe results

Expected Results:
- ✅ Only supplier-initiated events shown
- ✅ Buyer and system events hidden

Test 6.4: Filter by Date Range

Objective: Verify date range filter works

Steps:
1. Set “From” date to yesterday
2. Set “To” date to today
3. Click “Apply”

Expected Results:
- ✅ Only events within date range shown
- ✅ Older events excluded

Test 6.5: Pagination

Objective: Verify pagination works correctly

Steps:
1. Generate 25+ activity logs
2. Navigate to activity page
3. Click “Next Page”

Expected Results:
- ✅ 20 items per page
- ✅ “Next” button navigates to page 2
- ✅ “Previous” button returns to page 1
- ✅ Page number displayed

Test 6.6: CSV Export

Objective: Verify CSV export works and logs event

Steps:
1. Click “Export CSV” button
2. Wait for download

Expected Results:
- ✅ CSV file downloaded
- ✅ CSV contains columns:
- Timestamp
- Event Type
- Actor Role
- Summary
- Details (JSON string)
- IP Address
- User Agent
- ✅ `ACTIVITY_EXPORTED_CSV` event logged

Verification:

```
SELECT * FROM "ActivityLog"
WHERE "eventType" = 'ACTIVITY_EXPORTED_CSV'
ORDER BY "createdAt" DESC LIMIT 1;
```

Category 7: Supplier Activity UI (4 Tests)

Test 7.1: View Supplier Activity Page

Objective: Verify supplier sees only relevant events

Steps:
1. Log in as supplier
2. Navigate to `/supplier/rfps/[id]/activity`

Expected Results:
- ✅ Page loads successfully
- ✅ Only shows:
- Supplier’s own response events
- Broadcast messages
- Answers to supplier’s questions
- ✅ Simplified view (no expandable details)

Test 7.2: Broadcast Visibility

Objective: Verify broadcasts visible to all suppliers

Steps:
1. Create broadcast as buyer
2. Log in as Supplier A
3. View activity page

4. Log in as Supplier B
5. View activity page

Expected Results:
- ✅ Both suppliers see broadcast event
- ✅ Same timestamp and message

Test 7.3: Other Supplier Events Hidden

Objective: Verify supplier cannot see other suppliers' events

Steps:
1. Supplier A submits response
2. Supplier B views activity page

Expected Results:
- ❌ Supplier B does NOT see Supplier A's submission event
- ✅ Only Supplier B's own events visible

Test 7.4: Internal AI Logs Hidden

Objective: Verify suppliers cannot see internal logs

Steps:
1. Run AI extraction as buyer
2. Supplier views activity page

Expected Results:
- ❌ `AI_EXTRACTION_RUN` event NOT visible to supplier
- ❌ `SUPPLIER_COMPARISON_RUN` event NOT visible
- ❌ Other internal events hidden

Category 8: Security Tests (4 Tests)

Test 8.1: Buyer Cross-Company Access

Objective: Verify buyer cannot access other company's logs

Steps:
1. Create RFP as Buyer A (Company A)
2. Log in as Buyer B (Company B)
3. Try to access `/api/dashboard/rfps/[rfp-a-id]/activity`

Expected Results:
- ❌ 403 Forbidden error
- ✅ No data returned

Test 8.2: Supplier Cross-Supplier Access

Objective: Verify supplier cannot access other supplier's logs

Steps:
1. Supplier A logs in
2. Try to access API with Supplier B's `supplierContactId` in query

Expected Results:
- ❌ 403 Forbidden error
- ✅ No data returned

Test 8.3: Supplier Cannot See Buyer-Only Logs

Objective: Verify filtering works at API level

Steps:
1. Make API call to `/api/supplier/rfps/[id]/activity` as supplier
2. Inspect response

Expected Results:
- ❌ `RFP_CREATED`, `RFP_UPDATED` events NOT in response
- ❌ AI extraction events NOT in response
- ✅ Only supplier-scoped events returned

Test 8.4: Unauthenticated Access

Objective: Verify authentication is enforced

Steps:
1. Log out
2. Try to access `/api/dashboard/rfps/[id]/activity`

Expected Results:
- ❌ 401 Unauthorized error
- ✅ No data returned

Category 9: Logging Failure Tests (2 Tests)

Test 9.1: Primary Action Succeeds on Logging Failure

Objective: Verify logging failures don't break primary actions

Simulation Steps:
1. Temporarily modify `logActivity` to throw error:

```
// In lib/activity-log.ts
export async function logActivity(options) {
  throw new Error('SIMULATED LOGGING FAILURE');
}
```

1. Create a new RFP
2. Restore original function

Expected Results:
- ✅ RFP created successfully
- ✅ No error shown to user
- ❌ No `ActivityLog` record created
- ✅ Error logged to console

Test 9.2: Logging Errors Logged to Console

Objective: Verify errors are logged for debugging

Steps:
1. Open browser console
2. Perform action with simulated logging failure

Expected Results:
- ✅ Console shows error: `[ActivityLog] Failed to log activity`
- ✅ Error includes event type and summary

Test Summary Matrix

Category	Test #	Status	Notes
RFP Events	1.1	⬜	RFP Creation
RFP Events	1.2	⬜	RFP Update
RFP Events	1.3	⬜	Timeline Update
Supplier Portal	2.1	⬜	Invitation Sent
Supplier Portal	2.2	⬜	Supplier Login
Response Events	3.1	⬜	Draft Save
Response Events	3.2	⬜	Response Submit
Response Events	3.3	⬜	Attachment Upload
Response Events	3.4	⬜	Attachment Delete
AI Events	4.1	⬜	AI Extraction
AI Events	4.2	⬜	Comparison Run
AI Events	4.3	⬜	AI Summary
AI Events	4.4	⬜	Narrative Gen
AI Events	4.5	⬜	Report Gen
Q&A Events	5.1	⬜	Question Created
Q&A Events	5.2	⬜	Question Answered
Q&A Events	5.3	⬜	Broadcast Created
Buyer UI	6.1	⬜	View Page
Buyer UI	6.2	⬜	Event Filter
Buyer UI	6.3	⬜	Role Filter
Buyer UI	6.4	⬜	Date Filter
Buyer UI	6.5	⬜	Pagination
Buyer UI	6.6	⬜	CSV Export
Supplier UI	7.1	⬜	View Page
Supplier UI	7.2	⬜	Broadcast Visibility

Category	Test #	Status	Notes
Supplier UI	7.3	⬜	Other Supplier Hidden
Supplier UI	7.4	⬜	AI Logs Hidden
Security	8.1	⬜	Cross-Company
Security	8.2	⬜	Cross-Supplier
Security	8.3	⬜	Buyer-Only Logs
Security	8.4	⬜	Unauth Access
Logging Failures	9.1	⬜	Action Succeeds
Logging Failures	9.2	⬜	Console Logging

Legend:
⬜ Not Tested
✅ Passed
❌ Failed

Database Verification Queries

Check Total Activity Logs

```
SELECT COUNT(*) as total_logs FROM "ActivityLog";
```

Check Event Type Distribution

```
SELECT "eventType", COUNT(*) as count
FROM "ActivityLog"
GROUP BY "eventType"
ORDER BY count DESC;
```

Check Actor Role Distribution

```
SELECT "actorRole", COUNT(*) as count
FROM "ActivityLog"
GROUP BY "actorRole";
```

Check Logs for Specific RFP

```
SELECT "eventType", "actorRole", "summary", "createdAt"
FROM "ActivityLog"
WHERE "rfpId" = '<rfp-id>'
ORDER BY "createdAt" DESC;
```

Check IP Address Coverage

```
SELECT COUNT(*) as logs_with_ip
FROM "ActivityLog"
WHERE "ipAddress" IS NOT NULL;
```

Check User Agent Coverage

```
SELECT COUNT(*) as logs_with_ua
FROM "ActivityLog"
WHERE "userAgent" IS NOT NULL;
```

Find Duplicate Prevention Test

```
-- Should show no exact duplicates (same rfpId + eventType + createdAt)
SELECT "rfpId", "eventType", "createdAt", COUNT(*) as count
FROM "ActivityLog"
GROUP BY "rfpId", "eventType", "createdAt"
HAVING COUNT(*) > 1;
```

Performance Checks

1. Page Load Time

- Activity page with 100 logs: < 2 seconds
- Activity page with 1000 logs (paginated): < 2 seconds

2. Filter Performance

- Apply event type filter: < 500ms
- Apply date range filter: < 1 second

3. CSV Export Time

- Export 100 logs: < 3 seconds
- Export 1000 logs: < 10 seconds

4. Logging Overhead

- Time to create RFP with logging: ~100-200ms overhead
- Time to create RFP without logging (baseline): Measure for comparison

---

Troubleshooting Common Issues

Issue: No logs appearing in database

Check:
1. Prisma schema generated: `npx prisma generate`
2. Database migration applied: `npx prisma db push`
3. Console for errors: `[ActivityLog] Failed to log activity`

Fix:
- Ensure Prisma Client is regenerated
- Check database connection string

Issue: Logging errors break primary actions

Check:
1. Verify `logActivity` has try/catch wrapper
2. Check that no `throw` statements exist outside try/catch

Fix:
- Ensure all calls to `logActivity` use `await` (but inside try/catch)
- Verify fire-and-forget pattern

Issue: Supplier sees other supplier’s events

Check:
1. Filter logic in `/api/supplier/rfps/[rfpId]/activity/route.ts`
2. Verify `supplierContactId` filtering

Fix:
- Ensure WHERE clause includes `supplierContactId = session.user.supplierContactId`
- Check broadcast events have special handling

Issue: CSV export fails

Check:
1. CORS headers if downloading
2. File permissions on server
3. CSV generation library

Fix:
- Add proper `Content-Disposition` header
- Ensure CSV escaping for special characters

Post-Testing Actions

1. Document Results

- Update test summary matrix
- Note any failed tests
- Document workarounds

2. Bug Reporting

For each failed test:
1. Document expected vs actual behavior
2. Include SQL queries showing incorrect data
3. Note console errors
4. Provide steps to reproduce

3. Performance Review

- Document any performance issues
- Note slow queries for optimization
- Suggest indexes if needed

4. Code Review Checklist

- [ ] All 18 integration points implemented
- [ ] All logging wrapped in try/catch
- [ ] No breaking changes to existing features
- [ ] Security rules enforced at API level
- [ ] UI components follow design patterns
- [ ] CSV export format validated

Next Steps After Testing

If All Tests Pass:

1. Mark STEP 23 as complete
2. Commit changes with message: “feat(STEP 23): Implement Activity Log & Audit Trail System”
3. Update main documentation
4. Deploy to staging

If Tests Fail:

1. Fix issues one category at a time
2. Re-run failed tests
3. Document root causes
4. Update implementation guide if patterns need adjustment

Testing Prepared By: DeepAgent
Last Updated: November 30, 2025
Test Coverage: 30+ scenarios across 9 categories