Supplier Q&A Dialogue System - Testing Guide

Implementation Date: November 30, 2025
Feature: STEP 21 - Supplier Q&A Dialogue System
Status: ✅ Ready for Testing

Prerequisites

1. Start the Development Server

```
cd /home/ubuntu/fyndr/nextjs_space
npm install
npm run dev
```

2. Database Verification

Ensure Prisma migrations are applied:

```
npx prisma generate
npx prisma db push
```

3. Test User Accounts

You’ll need:
- **Buyer Account**: john@doe.com (or create a new buyer)
- **Supplier Accounts**: Multiple supplier portal users for testing cross-supplier scenarios

---

Test Scenario 1: Question Submission Before Window Opens

Objective: Verify suppliers cannot submit questions before the window opens.

Setup

1. Login as buyer (john@doe.com)
2. Navigate to an existing RFP or create a new one
3. Click “Edit” on the RFP
4. Scroll to “RFP Timeline & Milestones” section
5. Set **Ask Questions Start** to a **future date** (e.g., tomorrow)
6. Set **Ask Questions End** to 7 days from the start date
7. Save the RFP

Test Steps

1. Invite a supplier to the RFP (if not already invited)

2. Copy the supplier's magic link from the invitation
3. Open an incognito browser window
4. Paste the magic link to access the supplier portal
5. Navigate to "Questions & Answers" page

Expected Results

✅ **Status Banner:** Yellow with Clock icon
✅ **Message:** "Questions window not open yet. Opens on [date]."
✅ **Form:** Textarea is disabled (grayed out)
✅ **Submit Button:** Disabled
✅ **Explanation:** "The questions window is not open yet…"

API Verification

```
# Attempt to submit a question via API
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"Test question"}' \
  --cookie "session_cookie"

# Expected: 400 error
# Response: {"error": "Questions window is not open. You cannot submit questions at
this time."}
```

Test Scenario 2: Question Submission During Window

Objective: Verify suppliers can submit questions when the window is open.

Setup

1. Login as buyer
2. Edit the same RFP
3. Set **Ask Questions Start** to **today** or a past date
4. Set **Ask Questions End** to a **future date** (e.g., 7 days from now)
5. Save the RFP

Test Steps

1. As supplier (in incognito window), refresh the Questions page
2. Verify the status banner is now green with CheckCircle icon
3. Enter a test question: "What is the expected timeline for implementation?"
4. Verify character counter updates (e.g., "56/500")
5. Click "Submit Question"

Expected Results

✅ **Status Banner:** Green with "Window Open" message
✅ **Form:** Enabled textarea
✅ **Character Counter:** Real-time updates
✅ **Submit Button:** Enabled when valid
✅ **Success Toast:** "Question submitted successfully!"

✅ **Question Appears:** Immediately in the list below with “Pending” badge
✅ **Question Details:** Shows timestamp, question text, and status

API Verification

```bash
# Submit via API
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"What is the expected timeline for implementation?"}' \
  --cookie "session_cookie"

# Expected: 201 success
# Response includes question object with status=PENDING
```

Database Verification

```sql
SELECT * FROM "SupplierQuestion"
WHERE "rfpId" = '[your_rfp_id]'
ORDER BY "askedAt" DESC LIMIT 5;

-- Verify:
-- - question field contains submitted text
-- - status = 'PENDING'
-- - answer = null
-- - answeredAt = null
```

Test Scenario 3: Question Submission After Window Closes

Objective: Verify suppliers cannot submit questions after the window closes.

Setup

1. Login as buyer
2. Edit the RFP
3. Set **Ask Questions End** to a **past date** (e.g., yesterday)
4. Save the RFP

Test Steps

1. As supplier, refresh the Questions page
2. Attempt to type in the textarea (if visible)
3. Attempt to click Submit button

Expected Results

✅ **Status Banner:** Red with AlertCircle icon
✅ **Message:** “Questions window is closed. Closed on [date].”
✅ **Form:** Textarea disabled
✅ **Submit Button:** Disabled
✅ **Explanation:** “The questions window has closed…”

API Verification

```
# Attempt to submit after window closes
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"Late question"}' \
  --cookie "session_cookie"

# Expected: 400 error
# Response: {"error": "Questions window is not open. You cannot submit questions at
this time."}
```

Test Scenario 4: Supplier Sees Only Own Questions

Objective: Verify suppliers cannot see questions from other suppliers.

Setup

1. Invite **two or more suppliers** to the same RFP (Supplier A, Supplier B)
2. Ensure Ask Questions window is **OPEN**

Test Steps

1. As **Supplier A** (Browser 1, incognito):
   - Login via magic link
   - Navigate to Questions page
   - Submit question: “What is the support model?”

2. As **Supplier B** (Browser 2, different incognito):
   - Login via magic link for Supplier B
   - Navigate to Questions page
   - Submit question: “What are the integration requirements?”

3. As **Supplier A** (Browser 1):
   - Refresh the page
   - Check the questions list

Expected Results

✅ **Supplier A sees:** Only “What is the support model?”
✅ **Supplier B sees:** Only “What are the integration requirements?”
✅ **No Cross-Supplier Visibility:** Each supplier sees ONLY their own questions
✅ **No Supplier Identity:** No names, emails, or organizations visible

API Verification

```
# As Supplier A
curl http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  --cookie "supplier_a_session"

# Expected: Array with only Supplier A's questions

# As Supplier B
curl http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  --cookie "supplier_b_session"

# Expected: Array with only Supplier B's questions
```

Database Verification

```
-- Check supplierContactId filtering
SELECT
  sq."id",
  sq."question",
  sc."name" as supplier_name,
  sc."email"
FROM "SupplierQuestion" sq
JOIN "SupplierContact" sc ON sq."supplierContactId" = sc."id"
WHERE sq."rfpId" = '[your_rfp_id]';

-- Verify: Different supplierContactId values for different suppliers
```

Test Scenario 5: Buyer Answers with Broadcast = TRUE

Objective: Verify broadcast messages are visible to all suppliers.

Setup

- Continue from Scenario 4 with Supplier A's question pending

Test Steps

1. As **Buyer**:
   - Navigate to RFP detail page
   - Scroll to "Supplier Questions & Answers" panel
   - Click on the "Pending" filter tab
   - Find Supplier A's question: "What is the support model?"
   - Click "Answer" button

2. In the Answer Modal:
   - Verify question text is displayed
   - Verify supplier name is shown: "From: [Supplier A Name]"
   - Enter answer: "We provide 24/7 phone and email support with a dedicated account manager."
   - Verify "Broadcast this answer to ALL suppliers" toggle is **checked by default**
   - Click "Submit Answer"

3. Verify Buyer UI:
   - Question status changes to "Answered" (green badge)

   - Success toast appears
   - Question moves to "Answered" filter

4. As **Supplier A** (Browser 1):
   - Navigate to Questions page
   - Verify question status changed to "Answered" (green badge)
   - Verify answer text is visible
   - Navigate to main RFP detail page
   - Scroll to "Buyer Messages & Announcements" panel
   - **Verify broadcast message appears**

5. As **Supplier B** (Browser 2):
   - Navigate to main RFP detail page
   - Scroll to "Buyer Messages & Announcements" panel
   - **Verify the same broadcast message appears**
   - Verify NO supplier identity is shown (no mention of Supplier A)

Expected Results

✅ **Question Answered:** Status changes to ANSWERED
✅ **Answer Visible:** Supplier A sees answer in their questions list
✅ **Broadcast Created:** SupplierBroadcastMessage record created
✅ **Visible to All:** Both Supplier A and Supplier B see the broadcast
✅ **No Identity Leakage:** No mention of which supplier asked the question
✅ **Broadcast Format:** Shows date, message content, announcement icon

API Verification

```
# Answer with broadcast
curl -X POST http://localhost:3000/api/dashboard/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{
    "questionId": "[question_id]",
    "answer": "We provide 24/7 support...",
    "broadcast": true
  }' \
  --cookie "buyer_session"

# Expected: 200 success, broadcast object in response

# Verify broadcast visibility
curl http://localhost:3000/api/supplier/rfps/[rfpId]/broadcasts \
  --cookie "supplier_a_session"

curl http://localhost:3000/api/supplier/rfps/[rfpId]/broadcasts \
  --cookie "supplier_b_session"

# Expected: Both return the same broadcast messages
```

Database Verification

```sql
-- Check SupplierQuestion was updated
SELECT "status", "answer", "answeredAt"
FROM "SupplierQuestion"
WHERE "id" = '[question_id]';

-- Verify: status = 'ANSWERED', answer is set, answeredAt is not null

-- Check SupplierBroadcastMessage was created
SELECT * FROM "SupplierBroadcastMessage"
WHERE "rfpId" = '[rfp_id]'
ORDER BY "createdAt" DESC LIMIT 1;

-- Verify: message contains the answer, createdBy is buyer user ID
```

Test Scenario 6: Buyer Answers with Broadcast = FALSE

Objective: Verify answer is visible only to the asking supplier when broadcast is off.

Setup

- Supplier B has a pending question: “What are the integration requirements?”

Test Steps

1. As **Buyer**:
   - Navigate to RFP detail page
   - Find Supplier B’s question in the Supplier Questions panel
   - Click “Answer”

2. In the Answer Modal:
   - Enter answer: “We support REST API and webhook integrations.”
   - **Uncheck** “Broadcast this answer to ALL suppliers” toggle
   - Click “Submit Answer”

3. As **Supplier B** (Browser 2):
   - Navigate to Questions page
   - Verify question is now “Answered”
   - Verify answer text is visible
   - Navigate to Buyer Messages & Announcements panel
   - **Verify NO new broadcast message appears** (only the previous one)

4. As **Supplier A** (Browser 1):
   - Navigate to Buyer Messages & Announcements panel
   - **Verify NO new broadcast message**
   - Navigate to Questions page (if accessible)
   - **Verify Supplier B’s question is NOT visible**

Expected Results

✅ **Question Answered:** Status changes to ANSWERED for Supplier B
✅ **Answer Visible:** Only Supplier B sees the answer in their questions list
✅ **No Broadcast Created:** No new SupplierBroadcastMessage record

✅ **Private to Supplier B:** Supplier A does NOT see this answer anywhere
✅ **Announcements Panel:** No change for any supplier

API Verification

```
# Answer without broadcast
curl -X POST http://localhost:3000/api/dashboard/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{
    "questionId": "[supplier_b_question_id]",
    "answer": "We support REST API...",
    "broadcast": false
  }' \
  --cookie "buyer_session"

# Expected: 200 success, broadcast: null in response

# Verify no new broadcast
curl http://localhost:3000/api/supplier/rfps/[rfpId]/broadcasts \
  --cookie "supplier_a_session"

# Expected: Only the previous broadcast (from Scenario 5)
```

Database Verification

```sql
-- Check question was updated
SELECT "status", "answer", "answeredAt"
FROM "SupplierQuestion"
WHERE "id" = '[supplier_b_question_id]';

-- Verify: status = 'ANSWERED', answer is set

-- Count broadcasts
SELECT COUNT(*) FROM "SupplierBroadcastMessage"
WHERE "rfpId" = '[rfp_id]';

-- Verify: Count should be 1 (only from Scenario 5, not increased)
```

Test Scenario 7: Supplier Cannot Access Buyer Routes

Objective: Verify role-based access control prevents suppliers from accessing buyer endpoints.

Test Steps

1. As **Supplier** (logged in via magic link):
   - Attempt to navigate directly to: `http://localhost:3000/dashboard/rfps/[rfpId]`
   - Attempt to navigate to: `http://localhost:3000/dashboard/rfps/[rfpId]/edit`

Expected Results

✅ **Redirect:** Supplier is redirected to `/supplier` or `/login`
✅ **403 Error:** Or receives a Forbidden error page
✅ **No Data Exposure:** No buyer dashboard content is visible

API Verification

```
# Attempt to access buyer Q&A endpoint as supplier
curl http://localhost:3000/api/dashboard/rfps/[rfpId]/questions \
  --cookie "supplier_session"

# Expected: 403 error
# Response: {"error": "Forbidden: Buyer access only"}

# Attempt to answer questions
curl -X POST http://localhost:3000/api/dashboard/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"questionId":"...", "answer":"...", "broadcast":true}' \
  --cookie "supplier_session"

# Expected: 403 error

# Attempt to create broadcasts
curl -X POST http://localhost:3000/api/dashboard/rfps/[rfpId]/broadcasts \
  -H "Content-Type: application/json" \
  -d '{"message":"Unauthorized broadcast"}' \
  --cookie "supplier_session"

# Expected: 403 error
```

Test Scenario 8: Buyer Cannot Access Other Buyers' RFPs

Objective: Verify ownership validation prevents cross-buyer data access.

Setup

1. Create **Buyer A** (john@doe.com) with RFP-1
2. Create **Buyer B** (jane@buyer.com) with RFP-2
3. Invite suppliers to both RFPs

Test Steps

1. As **Buyer B**:
   - Attempt to navigate to Buyer A's RFP questions:
   `http://localhost:3000/dashboard/rfps/[rfp_1_id]`
2. Attempt to access Buyer A's questions via direct API call

Expected Results

✅ **403 Forbidden:** Buyer B cannot view Buyer A's RFP details
✅ **No Questions Visible:** If somehow accessed, no questions appear
✅ **No Answer Access:** Buyer B cannot answer questions on RFP-1

API Verification

```
# As Buyer B, attempt to access Buyer A's questions
curl http://localhost:3000/api/dashboard/rfps/[rfp_1_id]/questions \
  --cookie "buyer_b_session"

# Expected: 403 error
# Response: {"error": "Forbidden: You do not own this RFP"}

# Attempt to answer questions on RFP-1 as Buyer B
curl -X POST http://localhost:3000/api/dashboard/rfps/[rfp_1_id]/questions \
  -H "Content-Type: application/json" \
  -d '{"questionId":"...", "answer":"...", "broadcast":true}' \
  --cookie "buyer_b_session"

# Expected: 403 error
```

Additional Security Tests

Test 9: Character Limit Enforcement

```
# Submit question exceeding 500 characters
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"'$(python3 -c 'print("A" * 501)')'"}'  \
  --cookie "supplier_session"

# Expected: 400 error
# Response: {"error": "Question cannot exceed 500 characters"}
```

Test 10: SQL Injection Prevention

```
# Attempt SQL injection in question
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"Test'; DROP TABLE SupplierQuestion; --"}' \
  --cookie "supplier_session"

# Expected: Question stored safely without SQL execution
```

Test 11: XSS Prevention

```
# Attempt XSS in question
curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
  -H "Content-Type: application/json" \
  -d '{"question":"<script>alert(\"XSS\")</script>"}' \
  --cookie "supplier_session"

# Verify: Question is stored and displayed as plain text, no script execution
```

Integration Tests

Test 12: Comparison Narrative Integration

Objective: Verify Q&A context appears in comparison narratives.

Test Steps

1. As Buyer:
   - Answer 2-3 supplier questions with broadcast enabled
2. Navigate to comparison page: `/dashboard/rfps/[rfpId]/compare`
3. Click “Generate Narrative”
4. Wait for AI narrative generation
5. Review the generated narrative

Expected Results

✅ **qaContext Section:** Present in narrative JSON if broadcasts exist
✅ **Broadcast Messages:** Listed with dates and content
✅ **No Supplier Identity:** No mention of who asked questions
✅ **Context Only:** Explicitly states “does not directly impact scoring”

API Verification

```
# Generate narrative
curl -X POST http://localhost:3000/api/rfps/[rfpId]/compare/narrative \
  --cookie "buyer_session"

# Expected: Response includes "qaContext" field with broadcast summaries
```

Performance Tests

Test 13: Load Testing

Scenario: Multiple suppliers submitting questions simultaneously

```
# Simulate 10 concurrent question submissions
for i in {1..10}; do
  curl -X POST http://localhost:3000/api/supplier/rfps/[rfpId]/questions \
    -H "Content-Type: application/json" \
    -d "{\"question\":\"Test question $i\"}" \
    --cookie "supplier_session" &
done
wait

# Verify: All questions saved correctly with unique IDs
```

Test 14: Large Questions List

Scenario: Render 50+ questions in buyer UI

1. As Supplier, submit 50 questions (via script or manually)

2. As Buyer, open Supplier Questions panel
3. Test filtering (All/Pending/Answered)
4. Verify pagination or scrolling works smoothly

Expected: No performance degradation, smooth rendering

Validation Checklist

Database Schema ✅

```sql
-- Verify models exist
SELECT tablename FROM pg_tables WHERE schemaname = 'public'
AND tablename IN ('SupplierQuestion', 'SupplierBroadcastMessage');

-- Expected: Both tables present
```

API Endpoints ✅

```bash
# Test all endpoints exist (should not return 404)
curl -I http://localhost:3000/api/supplier/rfps/[rfpId]/questions
curl -I http://localhost:3000/api/supplier/rfps/[rfpId]/broadcasts
curl -I http://localhost:3000/api/supplier/rfps/[rfpId]/timeline
curl -I http://localhost:3000/api/dashboard/rfps/[rfpId]/questions
curl -I http://localhost:3000/api/dashboard/rfps/[rfpId]/broadcasts
```

UI Components ✅

- [ ] Supplier Questions page renders at `/supplier/rfps/[id]/questions`
- [ ] Broadcasts panel appears on supplier RFP detail page
- [ ] Supplier Questions panel appears on buyer RFP detail page
- [ ] Question window status banners display correctly
- [ ] Character counter updates in real-time
- [ ] Modals open and close properly
- [ ] Loading spinners appear during API calls
- [ ] Error messages display when expected
- [ ] Success toasts appear after actions

Common Issues & Fixes

Issue 1: Questions Window Always Shows “Not Open”

Cause: RFP timeline dates not set or invalid
Fix: Edit RFP and set proper `askQuestionsStart` and `askQuestionsEnd` dates

Issue 2: Supplier Cannot Submit Questions

Cause: Not properly authenticated or supplierContactId mismatch
Fix: Verify supplier logged in via magic link, check session cookie

Issue 3: Broadcast Not Visible to All Suppliers

Cause: Supplier not invited to RFP
Fix: Ensure all test suppliers are invited via Supplier Contacts panel

Issue 4: 403 Errors on API Calls

Cause: Role mismatch or ownership validation failure
Fix: Verify user role and RFP ownership in database

Test Summary

Test	Status	Notes
1. Before Window Opens	⏳ Pending	Verify disabled form and error message
2. During Window	⏳ Pending	Verify submission works
3. After Window Closes	⏳ Pending	Verify disabled form and error message
4. Own Questions Only	⏳ Pending	Critical security test
5. Broadcast = TRUE	⏳ Pending	Verify visibility to all
6. Broadcast = FALSE	⏳ Pending	Verify privacy
7. Supplier Access Control	⏳ Pending	Critical security test
8. Buyer Ownership	⏳ Pending	Critical security test
9. Character Limit	⏳ Pending	Validation test
10. SQL Injection	⏳ Pending	Security test
11. XSS Prevention	⏳ Pending	Security test
12. Narrative Integration	⏳ Pending	Integration test
13. Load Testing	⏳ Pending	Performance test
14. Large Lists	⏳ Pending	Performance test

Post-Testing Actions

1. **Update Test Summary:** Mark each test as ✅ Pass or ❌ Fail
2. **Document Issues:** Log any bugs or unexpected behavior

3. **Performance Metrics:** Record response times for critical operations
4. **Security Audit:** Verify all security tests passed
5. **User Acceptance:** Have stakeholders review the feature
6. **Update Documentation:** Reflect any changes discovered during testing

---

Testing Complete: [Date]
Tested By: [Name]
Approval: [Stakeholder Signature]