COCHAIRS_MEETING_IMPLEMENTATION_PLAN

PRISM Co-Chairs Meeting - Implementation Plan

Meeting Date: November 13, 2025
Attendees: Adam, Rory, Scott, Marie, Halle, Nina, Connie (mentioned)

Executive Summary

This document outlines the technical updates and governance considerations identified during the MaLDReTH II Working Group co-chairs meeting demo session. The meeting focused on reviewing the PRISM platform's interaction entry workflows and resulted in several key decisions to simplify the user experience while maintaining data quality.

Key Outcome: Consolidate Quick Add and Main Form into a single, progressive disclosure interface that balances simplicity with rich metadata capture.

Critical Decisions Made

1. **Lifecycle Stage Handling** ✅ APPROVED

Decision: Remove lifecycle stage as a user-selectable field. Auto-populate from tool metadata.

Rationale:

- Too complex to ask users to decide which stage an interaction "belongs to"
- Source and target tools already have assigned lifecycle stages
- Avoids confusion about whether it's the source stage, target stage, or somewhere in between
- Keeps data model flexible for future analysis

Implementation:

```
# Current (to be removed):
lifecycle_stage = db.Column(db.String(50), nullable=False)

# New approach:
# lifecycle_stage becomes a computed property from source_tool.stage and
target_tool.stage
# Store as list: [source_stage, target_stage]

@property
def lifecycle_stages(self):
    """Return list of lifecycle stages involved in this interaction."""
    return [self.source_tool.stage.name, self.target_tool.stage.name]
```

Database Changes:

- Remove `lifecycle_stage` column from `tool_interactions` table
- OR repurpose it to store JSON array: `["COLLECT", "PROCESS"]`
- Update all existing interactions to populate from tool relationships
- Update migration script to handle this transformation

UI Changes:

- Remove lifecycle stage dropdown from all interaction forms
- Display computed stages in view/detail pages as read-only badges
- Update validation to only require source_tool_id and target_tool_id

Quote from meeting:

> "I think simple is best... it should just be the lifecycle stage of the source tool or it's both and just gets auto populated."

---

2. **Form Consolidation** ✅ APPROVED

Decision: Merge Quick Add and Main Form into a single unified form with progressive disclosure.

Rationale:

- "Quick Add is going to be the only thing which people will ever use because it's just so much work to add content"
- Reduce interface complexity - one way to add interactions
- Make quick entry the default, rich entry optional
- Better for general public access (not just co-chairs)

Design Pattern: Progressive Disclosure

- **Top section (always visible):** Required fields only
- **Bottom sections (collapsed by default):** Optional enrichment fields
- **Visual differentiation:** Color or styling to distinguish required vs optional

Implementation Structure:

```
<!-- Top Section: Required Fields (Always Visible) -->
<div class="required-fields-section">
    <h5>Add Interaction - Required Fields</h5>

    <!-- Source Tool -->
    <select name="source_tool_id" required>...</select>

    <!-- Target Tool -->
    <select name="target_tool_id" required>...</select>

    <!-- Interaction Type (with glossary link) -->
    <select name="interaction_type" required>
        <option value="">-- Select interaction type --</option>
        <option value="API Integration">API Integration</option>
        ...
    </select>
    <a href="{{ url_for('glossary') }}#api-integration" class="help-link"
target="_blank">
        <i class="fas fa-question-circle"></i> See glossary
    </a>

    <!-- Description (make mandatory) -->
    <textarea name="description" required
              placeholder="One sentence describing what happens in this
interaction"></textarea>

    <!-- Submitted By (optional but encouraged) -->
    <input type="text" name="submitted_by" placeholder="Your name (optional)">
</div>

<!-- Collapsible Section 1: Technical Details -->
<div class="card mt-3">
    <div class="card-header" data-bs-toggle="collapse" data-bs-target="#technical-
details">
        <h6 class="mb-0">
            <i class="fas fa-chevron-down me-2"></i>
            Technical Details (Optional)
        </h6>
    </div>
    <div id="technical-details" class="collapse">
        <div class="card-body">
            <textarea name="technical_details"
                      placeholder="How does the interaction work technically?"></
textarea>
            <textarea name="benefits"
                      placeholder="What are the benefits?"></textarea>
            <textarea name="challenges"
                      placeholder="What challenges exist?"></textarea>
            <textarea name="examples"
                      placeholder="Real-world examples"></textarea>
        </div>
    </div>
</div>

<!-- Collapsible Section 2: Contact Information -->
<div class="card mt-3">
    <div class="card-header" data-bs-toggle="collapse" data-bs-target="#contact-
info">
        <h6 class="mb-0">
            <i class="fas fa-chevron-down me-2"></i>
            Contact Information (Optional)
        </h6>
    </div>
    <div id="contact-info" class="collapse">
        <div class="card-body">
            <input type="text" name="contact_person" placeholder="Contact person">
            <input type="text" name="organization" placeholder="Organization">
            <input type="email" name="email" placeholder="Email">
        </div>
    </div>
</div>

<!-- Collapsible Section 3: Classification -->
<div class="card mt-3">
    <div class="card-header" data-bs-toggle="collapse" data-bs-
target="#classification">
        <h6 class="mb-0">
            <i class="fas fa-chevron-down me-2"></i>
            Classification (Optional)
        </h6>
    </div>
    <div id="classification" class="collapse">
        <div class="card-body">
            <select name="priority">
                <option value="Medium" selected>Medium Priority</option>
                <option value="High">High Priority</option>
                <option value="Low">Low Priority</option>
            </select>
            <select name="complexity">
                <option value="Medium" selected>Medium Complexity</option>
                <option value="High">High Complexity</option>
                <option value="Low">Low Complexity</option>
            </select>
            <select name="status">
                <option value="Active" selected>Active</option>
                <option value="Deprecated">Deprecated</option>
                <option value="Planned">Planned</option>
            </select>
        </div>
    </div>
</div>
```

Benefits:

- ✅ Simplicity by default (matches Quick Add experience)
- ✅ Power when needed (all fields still available)
- ✅ Single interface to maintain
- ✅ Encourages quick entry, allows detailed curation
- ✅ Can add details later (edit function supports this)

Quote from meeting:

> "One option is we're kind of assuming that people will add this when they make the initial entry. But another possibility is that people could... come back later and add some more detail."

---

3. **Required Fields Update** ✅ APPROVED

Current Required:

- source_tool_id
- target_tool_id
- interaction_type
- lifecycle_stage ← **REMOVING**
- description ← **Currently NOT required**

New Required:

- source_tool_id
- target_tool_id
- interaction_type
- description ← **MAKE REQUIRED**

Implementation:

```
# Update ToolInteraction model
class ToolInteraction(db.Model):
    # ... existing fields ...

    # Remove or make nullable
    # lifecycle_stage = db.Column(db.String(50), nullable=False)  # REMOVE

    # Make mandatory
    description = db.Column(db.Text, nullable=False)  # ADD nullable=False
```

Migration Required:

```
-- Make description NOT NULL (set default for existing NULL values first)
UPDATE tool_interactions SET description = 'Description pending' WHERE description
IS NULL;
ALTER TABLE tool_interactions ALTER COLUMN description SET NOT NULL;

-- Handle lifecycle_stage (option 1: remove)
ALTER TABLE tool_interactions DROP COLUMN lifecycle_stage;

-- Handle lifecycle_stage (option 2: repurpose as JSON array)
ALTER TABLE tool_interactions ALTER COLUMN lifecycle_stage TYPE JSON USING
    (SELECT json_build_array(
        (SELECT stage FROM maldreth_stages WHERE id = source_tool.stage_id),
        (SELECT stage FROM maldreth_stages WHERE id = target_tool.stage_id)
    ));
```

---

4. **Tooltips and Contextualization** ✅ KEEP & ENHANCE

Decision: Maintain all tooltips, help links, and glossary integrations.

Feedback Received:

> "The most useful thing... is this glossary of the interaction types. Because, frankly, a good half of them, I didn't know what you were referring to... This is absolutely brilliant."
>
> "The documentation that you've put together with this is very good and very helpful. Not just this glossary, but all of the different help, the tool tips, things like that."

Actions:

1. ✅ Keep all existing tooltips
2. ✅ Ensure glossary links work from form dropdowns
3. ✅ Add tooltip/help icon next to "Interaction Type" dropdown suggesting users check glossary
4. ✅ Maintain inline help text for all fields
5. ✅ Keep "Back to Top" button (specifically praised)

Enhancement Opportunity:
Consider adding a dismissible "First Time User" banner that points to:

- User Guide
- Glossary
- Example interactions

---

5. **Make Quick Add Public** ✅ APPROVED

Decision: Remove access restrictions. Make the simplified entry available to all users.

Current State:

- Quick Add route exists but not in navigation
- Only accessible via direct URL: `/quick-add`
- Treated as "internal tool for co-chairs"

New State:

- Single unified "Add Interaction" form (per Decision #2)
- Accessible from main navigation
- Default collapsed sections make it "quick" for everyone
- Optional sections available for those with detailed knowledge

Quote from meeting:

> "Why are we limiting the quick add to us and not making it available generally?"

Deferred Decisions - Governance Discussion Required

The following items require a dedicated governance meeting (scheduled before next working group meeting on Nov 20):

1. Personal Data & GDPR Compliance ⚠️ GOVERNANCE

Issues Identified:

- Currently collecting names and emails without privacy notice
- No GDPR compliance framework in place
- Data stored in public Postgres on Heroku (Adam's personal account)
- No data retention policy
- No right-to-erasure mechanism

Questions for Governance Meeting:

- Do we need a formal privacy notice?
- Does RDA have a standard privacy notice for working group tools?
- Should we require consent checkbox for contact information?
- What's the data retention policy?
- Who owns the data - RDA, MaLDReTH II WG, or community?

Potential Solutions:

- Add privacy notice link to footer
- Make contact info collection truly optional with clear explanation

- Add "By submitting, you agree to..." checkbox
- Implement data minimization (don't ask for info we don't need)
- Consider moving to RDA infrastructure

Quote from meeting:

> "If we start storing personal data of some kind in any way other than the very most basic of prototype... we need to think about GDPR, privacy notices and all of that."

---

2. **Attribution & ORCID Integration** ⚠️ GOVERNANCE

Current State:

- `submitted_by` is a simple text field (can be anything)
- No authentication system
- No audit trail of edits
- Edits overwrite previous submitter name
- No affiliation tracking

Problems:

- Can't verify who submitted what
- Can't attribute contributions properly
- Can't contact contributors reliably
- Can't build reputation/contribution system
- Difficult to cite/acknowledge community

Options to Discuss:

Option A: ORCID Required

- Require ORCID login to submit interactions
- Automatic attribution with verified identity
- Can pull affiliation from ORCID profile
- Enables contributor tracking
- Best for academic community

Option B: ORCID Optional

- Allow anonymous submissions
- Offer ORCID login for attribution
- More accessible to industry/practitioners
- Harder to verify/curate

Option C: Hybrid

- Anonymous quick submissions allowed
- ORCID recommended and rewarded (badges, credits)
- Curators must have ORCID
- Transition period before requirement

Related Technical Work:

```
# If ORCID integration approved:
class ToolInteraction(db.Model):
    # Replace:
    submitted_by = db.Column(db.String(100))

    # With:
    submitted_by_orcid = db.Column(db.String(19))  # 0000-0001-2345-6789
    submitted_by_name = db.Column(db.String(200))  # From ORCID
    submitted_by_affiliation = db.Column(db.String(200))  # From ROR via ORCID

    # Add audit trail:
    created_by_orcid = db.Column(db.String(19))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    modified_by_orcid = db.Column(db.String(19))
    modified_at = db.Column(db.DateTime, onupdate=datetime.utcnow)
```

---

3. **Curation & Quality Control** ⚠️ GOVERNANCE

Questions:

- Who can curate/validate interactions?
- What's the approval workflow?
- How do we flag low-quality submissions?
- Can anyone edit, or only curators?
- How do we handle disputes about interaction descriptions?

Potential Models:

Model 1: Wikipedia Style

- Anyone can submit
- Anyone can edit
- Community self-regulates
- Flagging system for problems
- Reversible edits

Model 2: Moderated

- Anyone can submit
- Submissions pending until curator approves
- Only curators can edit after approval
- Clear quality bar

Model 3: Hybrid (Likely Best)

- Anyone can submit (goes live immediately)
- Community members can edit (tracked with audit trail)
- Curators can mark as "verified" or "high quality"
- Flagging system for review
- Disputed items go to curator review

GORC Parallel:
The meeting identified GORC as the exemplar to follow:

> "GORC pioneered a new model, which has been outstandingly successful... they found ways to keep the community governance continuing even after it became well established."

Action: Study GORC's governance model and adapt for PRISM.

---

4. **Persistent Identifiers** ⚠️ GOVERNANCE

Current State:

- Interactions identified by sequential integer (1, 2, 3...)
- Not persistent across database migrations
- Can't be cited reliably
- No external reference capability

Quote from meeting:

> "We might want to think in the longer term about identifiers for interactions... it would need to be an identifier at the very least like Handle."

Options:

Option A: Handle System

- Persistent, citable identifiers
- Used by many digital repositories
- Requires Handle.net registration
- Cost implications

Option B: Internal UUID

- Free, immediately implementable
- Not as "official" as Handle/DOI
- Still unique and persistent
- Example: `a1b2c3d4-e5f6-7890-abcd-ef1234567890`

Option C: DOI (via DataCite)

- Most prestigious for research
- Enables formal citation
- Requires DataCite membership
- Annual costs
- Interactions become "citable research objects"

Option D: Hybrid Approach

- Start with UUID for all interactions
- Offer DOI minting for "verified" or high-quality interactions
- Tier system (all get UUID, curated get DOI)

Implementation (UUID Example):

```
import uuid

class ToolInteraction(db.Model):
    id = db.Column(db.Integer, primary_key=True)  # Keep for internal use
    pid = db.Column(db.String(36), unique=True, nullable=False, default=lambda:
str(uuid.uuid4()))

    # Display URLs would use PID:
    # /interaction/a1b2c3d4-e5f6-7890-abcd-ef1234567890
    # Instead of:
    # /interaction/42
```

---

5. **Infrastructure & Hosting** ⚠️ GOVERNANCE

Current State:

- Hosted on Heroku (Adam's personal account)
- PostgreSQL database
- ~$7/month cost
- No backup strategy
- No disaster recovery plan

- No SLA or uptime guarantee

Quote from meeting:

> "It's all on a postgres database, which is public inside a Heroku instance, which I pay for... It's as secure as I choose to make it, which is not the great way to approach this."

Questions:

- Should this move to RDA infrastructure?
- Who pays for hosting long-term?
- What's the backup strategy?
- Do we need high availability?
- What about data sovereignty (GDPR implications)?

Options:

Option A: RDA Hosting

- Official RDA infrastructure
- Institutional backing
- Potentially free or subsidized
- More bureaucracy
- Check with Connie (RDA staff) about options

Option B: Continue Heroku (Formalized)

- Fast, flexible
- Easy for Adam to maintain
- MaLDReTH II WG or RDA pays for account
- Transfer ownership to organization account
- Set up automated backups
- Document disaster recovery

Option C: Alternative Cloud

- EU-based hosting for GDPR compliance
- Open Science Cloud infrastructure
- Potentially free for research projects

Implementation Roadmap

Phase 1: Form Consolidation (Week 1-2)

Priority: HIGH

1.1 Update Database Schema

- [ ] Create migration to remove `lifecycle_stage` column OR repurpose as JSON
- [ ] Update existing interactions with auto-computed lifecycle stages
- [ ] Make `description` field NOT NULL
- [ ] Test migration on local database
- [ ] Document rollback procedure

1.2 Update ToolInteraction Model

```
# streamlined_app.py

class ToolInteraction(db.Model):
    # Remove or update:
    # lifecycle_stage = db.Column(db.String(50), nullable=False)

    # Make required:
    description = db.Column(db.Text, nullable=False)

    # Add computed property:
    @property
    def lifecycle_stages(self):
        """Return list of lifecycle stages from source and target tools."""
        stages = []
        if self.source_tool and self.source_tool.stage:
            stages.append(self.source_tool.stage.name)
        if self.target_tool and self.target_tool.stage:
            stages.append(self.target_tool.stage.name)
        return stages

    @property
    def lifecycle_stages_display(self):
        """Return formatted stage names for display."""
        stages = self.lifecycle_stages
        if len(set(stages)) == 1:
            return stages[0]  # Same stage
        return f"{stages[0]} → {stages[1]}"
```

1.3 Create New Unified Form Template

- [ ] Create `templates/add_interaction_unified.html`
- [ ] Implement required fields section (always visible)
- [ ] Create 3 collapsible optional sections:
    - Technical Details (technical_details, benefits, challenges, examples)
    - Contact Information (contact_person, organization, email)
    - Classification (priority, complexity, status)
- [ ] Add Bootstrap collapse JavaScript
- [ ] Style collapsed sections with visual hierarchy

- [ ] Add "Expand All" / "Collapse All" helper buttons
- [ ] Maintain all existing tooltips and help links
- [ ] Add glossary quick-link icon next to Interaction Type

1.4 Update Routes

```
# Deprecate old routes:
# /quick-add -> redirect to /add-interaction
# /add-interaction -> new unified form

@app.route('/add-interaction', methods=['GET', 'POST'])
def add_interaction():
    """Unified form for adding interactions (replaces both quick-add and old main
form)."""
    if request.method == 'POST':
        try:
            # Description is now required
            if not request.form.get('description'):
                flash('Description is required.', 'danger')
                return redirect(url_for('add_interaction'))

            # No lifecycle_stage needed - computed from tools
            new_interaction = ToolInteraction(
                source_tool_id=request.form['source_tool_id'],
                target_tool_id=request.form['target_tool_id'],
                interaction_type=request.form['interaction_type'],
                description=request.form['description'],

                # Optional fields (from collapsed sections):
                technical_details=request.form.get('technical_details'),
                benefits=request.form.get('benefits'),
                challenges=request.form.get('challenges'),
                examples=request.form.get('examples'),
                contact_person=request.form.get('contact_person'),
                organization=request.form.get('organization'),
                email=request.form.get('email'),
                priority=request.form.get('priority', 'Medium'),
                complexity=request.form.get('complexity', 'Medium'),
                status=request.form.get('status', 'Active'),
                submitted_by=request.form.get('submitted_by', 'Anonymous')
            )

            db.session.add(new_interaction)
            db.session.commit()

            flash('Interaction added successfully!', 'success')
            return redirect(url_for('view_interaction',
interaction_id=new_interaction.id))

        except Exception as e:
            logger.error(f"Error adding interaction: {e}")
            flash('An error occurred. Please try again.', 'danger')
            db.session.rollback()

    # GET request
    all_tools =
ExemplarTool.query.filter_by(is_active=True).order_by(ExemplarTool.name).all()

    return render_template('add_interaction_unified.html',
                         tools=all_tools,
                         interaction_types=INTERACTION_TYPES)

# Redirect old URL for backward compatibility
@app.route('/quick-add')
def quick_add_redirect():
    return redirect(url_for('add_interaction'))
```

1.5 Update Navigation

```
<!-- templates/streamlined_base.html -->
<!-- Update navigation to point to unified form -->
<li class="nav-item">
    <a class="nav-link" href="{{ url_for('add_interaction') }}">
        <i class="fas fa-plus-circle me-1"></i>Add Interaction
    </a>
</li>
```

1.6 Update Display Templates

- [ ] Remove lifecycle_stage from display (use computed property)
- [ ] Update `templates/view_interaction.html` to show `lifecycle_stages_display`
- [ ] Update `templates/view_interactions.html` table columns
- [ ] Update `templates/edit_interaction.html` to match new unified form

1.7 Testing

- [ ] Test form submission with only required fields
- [ ] Test form submission with all optional fields
- [ ] Test collapsed section expansion/collapse
- [ ] Test validation (required fields)
- [ ] Test lifecycle stage display (computed from tools)
- [ ] Test with different tool combinations
- [ ] Test edit functionality
- [ ] Mobile responsive testing

Phase 2: Heroku Deployment Update (Week 2)

Priority: HIGH

2.1 Update Migration Script

```
# heroku_release.py - update migrate_database_schema()

def migrate_database_schema():
    """Safely migrate database schema to add new fields without data loss."""
    try:
        inspector = db.inspect(db.engine)
        tables = inspector.get_table_names()

        if 'exemplar_tools' not in tables:
            logger.info("Tables don't exist yet, creating all tables...")
            db.create_all()
            logger.info("✅ Database tables created successfully")
            return

        # Check if feedback table exists
        if 'feedback' not in tables:
            logger.info("Feedback table missing, creating it...")
            Feedback.__table__.create(db.engine, checkfirst=True)
            logger.info("✅ Feedback table created successfully")

        # NEW: Handle lifecycle_stage removal from tool_interactions
        try:
            interaction_columns = [col['name'] for col in
inspector.get_columns('tool_interactions')]

            if 'lifecycle_stage' in interaction_columns:
                logger.info("Migrating lifecycle_stage field...")
                # Option: Just drop it if we're not storing it anymore
                db.session.execute(db.text('ALTER TABLE tool_interactions DROP
COLUMN lifecycle_stage'))
                db.session.commit()
                logger.info("✅ Lifecycle_stage column removed")

            # Ensure description is NOT NULL
            if 'description' in interaction_columns:
                # First, fill in any NULL values
                db.session.execute(db.text(
                    "UPDATE tool_interactions SET description = 'Description
pending' WHERE description IS NULL"
                ))
                # Then make NOT NULL
                db.session.execute(db.text(
                    'ALTER TABLE tool_interactions ALTER COLUMN description SET NOT
NULL'
                ))
                db.session.commit()
                logger.info("✅ Description field set to NOT NULL")

        except Exception as e:
            logger.error(f"Error migrating tool_interactions: {e}")
            db.session.rollback()

        # ... rest of existing migration code ...
```

2.2 Test Heroku Deployment

- [ ] Test migration on Heroku staging/dev instance
- [ ] Verify existing interactions display correctly
- [ ] Verify new interactions can be created
- [ ] Check for any database errors in logs
- [ ] Verify lifecycle stages compute correctly

2.3 Documentation

- [ ] Update user guide with new unified form instructions
- [ ] Update screenshots in user guide
- [ ] Document lifecycle stage change for users
- [ ] Add migration notes to CHANGELOG

Phase 3: UI/UX Polish (Week 3)

Priority: MEDIUM

3.1 Visual Enhancements

- [ ] Add visual distinction for required vs optional sections
    - Use color (e.g., blue border for required, gray for optional)
    - Use icons (e.g., asterisk for required, plus for optional)
- [ ] Improve collapsed section headers
    - Add preview of filled fields when collapsed
    - Show badge count (e.g., "3 fields completed")
- [ ] Add animation for collapse/expand (smooth transitions)

3.2 User Guidance

- [ ] Add "First Time User?" callout banner (dismissible)
- [ ] Add field counter (e.g., "2 of 4 required fields completed")
- [ ] Add "Why provide this?" tooltips for optional sections
- [ ] Add example links (e.g., "See example interaction" button)

3.3 Form Behavior

- [ ] Remember collapsed/expanded state (localStorage)
- [ ] Auto-expand section if validation error in that section
- [ ] Add "Save Draft" functionality (future enhancement)
- [ ] Add keyboard shortcuts (Ctrl+S to save)

Phase 4: Documentation Updates (Week 3)

Priority: MEDIUM

4.1 User Guide Updates

- [ ] Update "Adding Interactions" section
- [ ] Remove references to "Quick Add vs Main Form"
- [ ] Add section on "Required vs Optional Fields"
- [ ] Update screenshots throughout guide
- [ ] Add GIF/video of form in action

4.2 Glossary Updates

- [ ] Ensure all interaction types fully documented
- [ ] Add "How to Choose" decision tree
- [ ] Add visual examples for each type

4.3 Help Text Review

- [ ] Review all tooltip text for accuracy
- [ ] Ensure consistent voice/tone
- [ ] Add more examples where helpful
- [ ] Fix any typos or unclear wording

Phase 5: Governance Preparation (Week 4)

Priority: CRITICAL for long-term sustainability

5.1 Governance Discussion Agenda

Prepare materials for co-chairs governance meeting:

Topic 1: Personal Data & GDPR

- [ ] Research RDA's existing privacy notices
- [ ] Draft privacy notice for PRISM
- [ ] Propose consent mechanism

- [ ] Document current data collected
- [ ] Propose data minimization strategy

Topic 2: Attribution System

- [ ] Research ORCID integration cost/effort
- [ ] Prepare pros/cons of each attribution option
- [ ] Mock up ORCID login flow
- [ ] Propose phased implementation

Topic 3: Curation Model

- [ ] Document GORC's curation approach
- [ ] Propose 3 models with examples
- [ ] Draft curator role description
- [ ] Propose flagging/review workflow

Topic 4: Persistent Identifiers

- [ ] Research Handle vs DOI vs UUID costs
- [ ] Propose recommendation with rationale
- [ ] Estimate implementation effort
- [ ] Document migration path

Topic 5: Infrastructure

- [ ] Contact Connie about RDA hosting options
- [ ] Document current costs and limitations
- [ ] Propose 3-year sustainability plan
- [ ] Estimate hosting costs for each option

5.2 Governance Document Draft

Create `docs/GOVERNANCE.md` template:

```
# PRISM Governance Framework

## Ownership & Stewardship
- Platform owned by: [RDA MaLDReTH II WG / RDA / Community]
- Technical maintenance: [Role/Person]
- Content curation: [Role/Team]

## Data Governance
- Personal data policy: [TBD]
- Privacy notice: [Link]
- Data retention: [Policy]
- GDPR compliance: [Approach]
```

```
## Content Governance
- Who can submit: [Open / ORCID only / Restricted]
- Who can edit: [Anyone / Curators / Submitter only]
- Approval workflow: [None / Moderated / Hybrid]
- Quality standards: [Link to guidelines]

## Attribution
- Identity verification: [ORCID / Optional / None]
- Contributor credits: [How acknowledged]
- Citation policy: [How to cite interactions]

## Curation Team
- Roles: [Curator / Admin / Moderator]
- Responsibilities: [Defined]
- Appointment process: [How curators are selected]

## Technical Governance
- Code repository: [GitHub]
- Issue tracking: [GitHub Issues]
- Feature requests: [Process]
- Release schedule: [Cadence]

## Dispute Resolution
- Content disputes: [Process]
- Appeals: [Process]
```

Quick Wins (Can Implement Immediately)

1. Make Description Required ✅

Effort: 5 minutes
Impact: High (data quality)

```
# In ToolInteraction model:
description = db.Column(db.Text, nullable=False)

# In add_interaction route:
if not request.form.get('description'):
    flash('Description is required.', 'danger')
    return redirect(url_for('add_interaction'))
```

2. Add Tooltip for Interaction Type ✅

Effort: 10 minutes
Impact: Medium (user guidance)

```
<label for="interaction_type">
    Interaction Type
    <span class="text-danger">*</span>
    <a href="{{ url_for('glossary') }}#interaction-types"
       class="text-muted ms-2"
       title="Not sure? Check the glossary"
       target="_blank">
        <i class="fas fa-question-circle"></i>
    </a>
</label>
```

3. Add "Expand All" Button to Current Forms ✅

Effort: 15 minutes
Impact: Low (UX improvement)

```
<button type="button" class="btn btn-sm btn-outline-secondary"
onclick="expandAll()">
    <i class="fas fa-expand-alt"></i> Expand All Sections
</button>

<script>
function expandAll() {
    document.querySelectorAll('.collapse').forEach(el => {
        bootstrap.Collapse.getOrCreateInstance(el).show();
    });
}
</script>
```

4. Update Lifecycle Stage Display to Badge Format ✅

Effort: 10 minutes
Impact: Medium (visual clarity)

```
<!-- In view_interaction.html: -->
<div class="mb-3">
    <strong>Lifecycle Stages:</strong>
    {% for stage in interaction.lifecycle_stages %}
        <span class="badge bg-primary">{{ stage }}</span>
    {% endfor %}
</div>
```

Testing Checklist

Unit Tests

- [ ] Test lifecycle_stages property with different tool combinations
- [ ] Test description validation (required field)
- [ ] Test form submission with minimal fields
- [ ] Test form submission with all fields
- [ ] Test collapsed section state persistence

Integration Tests

- [ ] Test full interaction creation workflow
- [ ] Test interaction editing workflow
- [ ] Test interaction display with computed lifecycle stages
- [ ] Test database migration (local and Heroku)

User Acceptance Testing

- [ ] Have co-chairs test new unified form
- [ ] Collect feedback on collapsed sections UX
- [ ] Test with non-technical users
- [ ] Mobile device testing

Performance Testing

- [ ] Test form load time with 200+ tools in dropdowns
- [ ] Test view_interactions page with 500+ interactions
- [ ] Test search/filter performance

Rollback Plan

If major issues encountered after deployment:

Step 1: Immediate Rollback

```
# Revert to previous Heroku release
heroku releases:rollback -a prism-int-dev

# Or rollback specific commit
git revert HEAD
git push heroku feature/form-consolidation:main
```

Step 2: Database Rollback

```
-- Restore lifecycle_stage column if needed
ALTER TABLE tool_interactions ADD COLUMN lifecycle_stage VARCHAR(50);

-- Repopulate from source tool stage (temporary)
UPDATE tool_interactions ti
SET lifecycle_stage = (
    SELECT ms.name FROM maldreth_stages ms
    JOIN exemplar_tools et ON et.stage_id = ms.id
    WHERE et.id = ti.source_tool_id
);

-- Make description nullable again if needed
ALTER TABLE tool_interactions ALTER COLUMN description DROP NOT NULL;
```

Step 3: Revert Template Changes

- Restore old `add_interaction.html` and `quick_add.html`
- Restore old route handlers
- Update navigation links

Success Metrics

Adoption Metrics

- Number of interactions added via new unified form
- Percentage of submissions with optional fields completed
- Number of users who expand optional sections
- Time to complete form (vs old Quick Add)

Quality Metrics

- Percentage of interactions with complete descriptions
- Percentage with technical details filled
- Percentage with contact information
- Number of edits per interaction (lower is better initially)

UX Metrics

- Form abandonment rate
- User feedback sentiment (via feedback form)
- Support questions about form usage
- Mobile vs desktop usage

Communication Plan

Internal (Co-Chairs)

- **Pre-deployment:** Share staging link for testing
- **Deployment:** Email notification with changelog
- **Post-deployment:** Collect feedback via dedicated meeting

Working Group

- **Next meeting (Nov 20):** Demo new unified form
- **Q&A session:** Address questions about changes
- **Documentation:** Point to updated user guide

Public Users

- **Announcement:** Update alpha banner with "New improved form!"
- **Changelog:** Add entry to /about page
- **User guide:** Update with new screenshots and instructions

---

Open Questions for Governance Meeting

1. **ORCID Integration:** Required, optional, or not at all?
2. **Privacy Notice:** Can we use RDA's standard notice?
3. **Curation Model:** Who can edit? What's the approval process?
4. **Infrastructure:** Move to RDA hosting or formalize current setup?
5. **Persistent IDs:** UUID now, DOI later, or Handle system?
6. **Data Ownership:** Who owns the data - RDA, WG, or community commons?
7. **Long-term Funding:** How do we sustain infrastructure costs?
8. **Versioning:** Do we need interaction versioning/history?
9. **API Access:** Should there be an API for programmatic access?
10. **Licensing:** What license for the interaction data (CC0, CC-BY)?

Appendix: Meeting Quotes

On Simplicity

> "I think simple is best. Totally agree... I also think we could have multiple [stages]. I don't know, it could just get messy."

On Documentation

> "Everything you've done is this glossary of the interaction types. This is absolutely brilliant. It's actually this is the essential thing which is going to make this project work."

On Quick Add

> "I also think that the quick add is going to be the only thing which people will ever use because it's just so much work to add content."

On Form Consolidation

> "Could we just do the you do it a different way by having just one add option... and then somehow by some visual clue, color, or something else, just have the in effect, the quick add bit at the top."

On Community Ownership

> "I really really want to make sure that this does the same thing [as original Maldreth]. It reflects what people think and have done rather than just me."

On GORC as Model

> "GORC pioneered a new model, which has been outstandingly successful... they found ways to keep the community governance continuing even after it became well established."

Next Steps

Immediate (This Week):

1. ✅ Create this implementation plan document
2. Create feature branch: `feature/unified-interaction-form`
3. Implement database migration for lifecycle_stage
4. Create unified form template
5. Update routes

Short-term (Next 2 Weeks):

1. Deploy to Heroku dev instance
2. Test with co-chairs
3. Refine based on feedback
4. Update all documentation
5. Deploy to production

Medium-term (Next Month):

1. Governance meeting with co-chairs
2. Document governance decisions
3. Implement ORCID (if approved)
4. Add persistent identifiers (if approved)
5. Privacy notice implementation

Long-term (Next Quarter):

1. Transition to sustainable infrastructure
2. Formalize curation team
3. Community announcement
4. Integration with GORC (if applicable)

Document Version: 1.0
Created: November 13, 2025
Author: Adam Vials Moore
Status: DRAFT for co-chairs review