Phase 4: Lead Creation & Family Inquiry Flow - Implementation Documentation

Date: December 7, 2025
Branch: feature/family-leads-mvp
Status: ✅ **COMPLETE**

Table of Contents

Overview

Phase 4 implements the complete inquiry submission system that allows Family users to submit care inquiries to Aides (Caregivers) or Providers directly from the marketplace detail pages. This creates Lead records in the database that can be managed by operators.

Key Features

- ✅ POST /api/leads endpoint with validation and RBAC
- ✅ InquiryForm modal component with client-side validation
- ✅ RequestCareButton component with authentication checks
- ✅ Updated Aide detail page with “Request Care” CTA
- ✅ Updated Provider detail page with “Request Care” CTA
- ✅ Authentication and role-based access control
- ✅ Audit logging for compliance
- ✅ Error handling and user feedback

Architecture

Data Flow

```
Family User
    ↓
Marketplace (Browse Aides/Providers)
    ↓
Detail Page (Aide or Provider)
    ↓
Click "Request Care" Button
    ↓
Authentication Check
    ↓ (if authenticated as FAMILY)
InquiryForm Modal Opens
    ↓
User Fills Form & Submits
    ↓
POST /api/leads
    ↓
Validation & Database Insert
    ↓
Lead Record Created
    ↓
Success Message
    ↓
(Optional) Redirect to Leads Page
```

Database Schema

Lead Model (from Phase 1):

```
model Lead {
  id                    String         @id @default(cuid())
  familyId              String
  targetType            LeadTargetType // AIDE or PROVIDER
  aideId                String?        // FK to Caregiver
  providerId            String?        // FK to Provider
  status                LeadStatus     @default(NEW)
  message               String?        @db.Text
  preferredStartDate    DateTime?
  expectedHoursPerWeek  Int?
  location              String?
  operatorNotes         String?        @db.Text
  assignedOperatorId    String?
  deletedAt             DateTime?
  createdAt             DateTime       @default(now())
  updatedAt             DateTime       @updatedAt

  // Relations
  family                Family         @relation(...)
  aide                  Caregiver?     @relation(...)
  provider              Provider?      @relation(...)
  assignedOperator      User?          @relation(...)
}
```

Implementation Details

1. Authentication Utility

File: `src/lib/inquiry-auth.ts`

Purpose: Centralized authentication and authorization checks for inquiry submission.

Functions:
- `canSubmitInquiry(session, currentUrl)` : Returns authorization check result
- `getInquiryErrorMessage(authCheck)` : Returns user-friendly error message

Logic:
1. Check if user is authenticated
2. Check if user has FAMILY role
3. Return authorization result with redirect URL or error message

Usage:

```
const authCheck = canSubmitInquiry(session, currentUrl);
if (!authCheck.canSubmit) {
  if (authCheck.redirectUrl) {
    router.push(authCheck.redirectUrl);
  } else {
    showError(getInquiryErrorMessage(authCheck));
  }
}
```

---

2. API Endpoint

File: `src/app/api/leads/route.ts`

Method: POST

Endpoint: `/api/leads`

Authentication: Required (FAMILY role only)

Request Body:

```
{
  "targetType": "AIDE" | "PROVIDER",
  "targetId": "string (required)",
  "message": "string (10-1000 chars, required)",
  "preferredStartDate": "ISO datetime (optional)",
  "expectedHoursPerWeek": "integer 1-168 (optional)",
  "location": "string max 255 (optional)"
}
```

Response (Success):

```
{
  "success": true,
  "message": "Your inquiry has been submitted successfully to {targetName}.",
  "data": {
    "leadId": "string",
    "targetType": "AIDE" | "PROVIDER",
    "targetName": "string",
    "status": "NEW",
    "createdAt": "ISO datetime"
  }
}
```

Response (Error):

```
{
  "error": "Error message",
  "details": "Optional error details"
}
```

Status Codes:
- `201` : Lead created successfully
- `400` : Validation error
- `401` : Not authenticated
- `403` : Not authorized (not FAMILY role)
- `404` : Target not found or Family profile not found
- `500` : Internal server error

Validation Rules:
- `targetType` : Must be “AIDE” or “PROVIDER”
- `targetId` : Required, non-empty string
- `message` : Required, 10-1000 characters
- `preferredStartDate` : Optional, valid ISO datetime
- `expectedHoursPerWeek` : Optional, integer 1-168
- `location` : Optional, max 255 characters

Security Features:
- Session-based authentication via NextAuth
- Role-based access control (FAMILY only)
- Input validation with Zod
- SQL injection prevention via Prisma
- Audit logging for compliance

---

3. UI Components

3.1 InquiryForm Component

File: `src/components/marketplace/InquiryForm.tsx`

Type: Client Component (modal)

Props:

```
{
  targetType: "AIDE" | "PROVIDER";
  targetId: string;
  targetName: string;
  onSuccess?: (leadId: string) => void;
  onClose: () => void;
}
```

Features:
- Modal overlay with backdrop
- Form fields:
- Message (textarea, required, 10-1000 chars)
- Preferred Start Date (date picker, optional)
- Expected Hours Per Week (number input, optional, 1-168)
- Location (text input, optional, max 255 chars)
- Client-side validation with error messages
- Character counter for message field
- Loading state during submission
- Success message with auto-close (2 seconds)
- Error message display
- Cancel button
- Submit button with loading spinner
- Responsive design

Validation:
- Real-time validation on input change
- Pre-submit validation
- Server-side validation error handling
- User-friendly error messages

UX Flow:
1. Modal opens
2. User fills form
3. Click “Submit Inquiry”
4. Loading state shown
5. Success: Show success message, auto-close after 2s
6. Error: Show error message, allow retry

3.2 RequestCareButton Component

File: `src/components/marketplace/RequestCareButton.tsx`

Type: Client Component

Props:

```
{
  targetType: "AIDE" | "PROVIDER";
  targetId: string;
  targetName: string;
  className?: string;
}
```

Features:
- Authentication check before opening modal
- Role-based access control
- Loading state during session fetch
- Error message display for non-FAMILY users
- Opens InquiryForm modal on click
- Handles success callback
- Customizable styling via className prop

Behavior:
- **Not authenticated**: Redirects to `/auth/login` with return URL
- **Wrong role**: Shows error message
- **FAMILY role**: Opens InquiryForm modal

Integration:

```
<RequestCareButton
  targetType="AIDE"
  targetId={caregiverId}
  targetName="John Doe"
/>
```

---

4. Updated Detail Pages

4.1 Aide/Caregiver Detail Page

File: `src/app/marketplace/caregivers/[id]/page.tsx`

Changes:
1. Added `RequestCareButton` import
2. Added prominent CTA section at top of body:
- Blue background box
- Headline: “Interested in hiring {name}?”
- Description text
- RequestCareButton with AIDE targetType

Placement: Top of page body, before details section

Visual Design:
- Background: `bg-primary-50`
- Border: `border-primary-200`
- Rounded corners
- Padding: 4 units
- Margin bottom: 6 units

4.2 Provider Detail Page

File: `src/app/marketplace/providers/[id]/page.tsx`

Changes:
1. Added `RequestCareButton` import
2. Replaced “Send Message” button with two CTAs:
- **Primary CTA**: RequestCareButton (top)
- **Secondary CTA**: Send Message button (below, outlined style)

Placement: Sidebar contact card

Visual Design:
- Primary CTA: Solid blue background
- Secondary CTA: White background with border
- Vertical stack with 3-unit spacing
- Help text at bottom

API Specifications

POST /api/leads

Authentication: Required (NextAuth session)

Authorization: FAMILY role only

Rate Limiting: None (consider adding in production)

Request Body Schema:

```
{
  targetType: "AIDE" | "PROVIDER",     // Required
  targetId: string,                     // Required
  message: string,                      // Required, 10-1000 chars
  preferredStartDate?: string,          // Optional, ISO datetime
  expectedHoursPerWeek?: number,        // Optional, 1-168
  location?: string                     // Optional, max 255 chars
}
```

Database Operations:
1. Fetch Family record for authenticated user
2. Validate target exists (Aide or Provider)
3. Create Lead record with NEW status
4. Create audit log entry

Audit Logging:

```
{
  userId: session.user.id,
  action: "CREATE",
  resourceType: "LEAD",
  resourceId: lead.id,
  description: "Created lead inquiry to {targetType}: {targetName}",
  metadata: {
    targetType,
    targetId,
    targetName,
    familyId
  }
}
```

Error Handling:
- Validation errors: Return field-specific errors
- Authentication errors: 401 Unauthorized
- Authorization errors: 403 Forbidden
- Not found errors: 404 Not Found
- Database errors: 500 Internal Server Error
- All errors logged to console

User Flows

Flow 1: Successful Inquiry Submission

Scenario: Family user submits inquiry to an Aide

Steps:
1. Family user logs in
2. Navigates to marketplace → caregivers
3. Clicks on a caregiver card
4. Views caregiver detail page
5. Sees “Request Care” button in blue box
6. Clicks “Request Care”
7. InquiryForm modal opens
8. Fills in:
- Message: “Looking for help with my elderly mother...”
- Preferred Start Date: Next Monday
- Hours Per Week: 20
- Location: “Seattle, WA”
9. Clicks “Submit Inquiry”
10. Loading spinner shown
11. Success message: “Your inquiry has been submitted successfully”
12. Modal auto-closes after 2 seconds
13. Lead record created in database with status=NEW

Expected Outcome: ✅ Lead created, family can view it in their dashboard

---

Flow 2: Non-Authenticated User

Scenario: User not logged in tries to submit inquiry

Steps:
1. User visits caregiver detail page (not logged in)
2. Clicks “Request Care”
3. Redirected to `/auth/login?callbackUrl=/marketplace/caregivers/[id]`
4. User logs in or registers as FAMILY
5. Redirected back to caregiver detail page
6. Can now submit inquiry

Expected Outcome: ✅ User redirected to login, returns after auth

---

Flow 3: Wrong Role

Scenario: CAREGIVER user tries to submit inquiry

Steps:
1. Caregiver user logs in
2. Visits aide detail page
3. Clicks “Request Care”
4. Error message shown: “Only family members can submit care inquiries”
5. Current role shown: “Currently logged in as: CAREGIVER”
6. Cannot submit inquiry

Expected Outcome: ✅ Error message shown, no inquiry created

Flow 4: Validation Error

Scenario: Family user submits incomplete form

Steps:
1. Family user opens InquiryForm
2. Types short message: “Help”
3. Clicks “Submit Inquiry”
4. Validation error shown: “Message must be at least 10 characters”
5. Message field highlighted in red
6. User corrects message
7. Resubmits successfully

Expected Outcome: ✅ Client-side validation prevents invalid submission

---

Flow 5: Provider Inquiry

Scenario: Family user submits inquiry to Provider

Steps:
1. Family user logs in
2. Navigates to marketplace → providers

3. Clicks on a provider card
4. Views provider detail page
5. Sees “Request Care” button in sidebar
6. Clicks “Request Care”
7. InquiryForm opens with provider name
8. Fills form and submits
9. Lead created with targetType=PROVIDER

Expected Outcome: ✅ Lead created for Provider target

Testing

Manual Testing Checklist

API Testing

1. Valid FAMILY User Submission

```
# Prerequisite: Get auth token for FAMILY user
TOKEN="your_family_user_token"
CAREGIVER_ID="clxyz123"

curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "AIDE",
    "targetId": "'$CAREGIVER_ID'",
    "message": "I need help caring for my elderly mother. She has mobility issues and
needs assistance with daily activities.",
    "preferredStartDate": "2025-12-15T00:00:00Z",
    "expectedHoursPerWeek": 20,
    "location": "Seattle, WA"
  }'
```

Expected: 201 Created with lead data

---

2. Missing Required Fields

```
curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "AIDE"
  }'
```

Expected: 400 Bad Request with validation errors

---

3. Invalid targetType

```
curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "INVALID",
    "targetId": "xyz",
    "message": "This is a test message for inquiry."
  }'
```

Expected: 400 Bad Request with validation error

---

4. Message Too Short

```
curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "AIDE",
    "targetId": "'$CAREGIVER_ID'",
    "message": "Help"
  }'
```

Expected: 400 Bad Request, “Message must be at least 10 characters”

---

5. Unauthenticated Request

```
curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -d '{
    "targetType": "AIDE",
    "targetId": "'$CAREGIVER_ID'",
    "message": "This is a test message."
  }'
```

Expected: 401 Unauthorized

---

6. Non-FAMILY User (e.g., CAREGIVER)

```
# Use token for CAREGIVER user
CAREGIVER_TOKEN="caregiver_token"

curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$CAREGIVER_TOKEN" \
  -d '{
    "targetType": "AIDE",
    "targetId": "'$CAREGIVER_ID'",
    "message": "This is a test message."
  }'
```

Expected: 403 Forbidden, “Only family members can submit care inquiries”

---

7. Non-existent Target

```
curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "AIDE",
    "targetId": "nonexistent123",
    "message": "This is a test message for inquiry."
  }'
```

Expected: 404 Not Found, “Caregiver not found”

---

8. Provider Inquiry

```
PROVIDER_ID="prv123"

curl -X POST http://localhost:3000/api/leads \
  -H "Content-Type: application/json" \
  -H "Cookie: next-auth.session-token=$TOKEN" \
  -d '{
    "targetType": "PROVIDER",
    "targetId": "'$PROVIDER_ID'",
    "message": "I need transportation services for my father.",
    "expectedHoursPerWeek": 10
  }'
```

Expected: 201 Created with lead data

UI Testing

1. Aide Detail Page - Request Care Button

- [ ] Visit `/marketplace/caregivers/[id]` as unauthenticated user
- [ ] Verify “Request Care” button is visible
- [ ] Click button → redirected to login
- [ ] Login as FAMILY user
- [ ] Redirected back to detail page
- [ ] Click “Request Care” → modal opens

2. InquiryForm Modal

- [ ] Modal displays with correct caregiver name
- [ ] All form fields present and functional
- [ ] Message field has character counter
- [ ] Date picker allows future dates only
- [ ] Hours input accepts 1-168 only
- [ ] Cancel button closes modal
- [ ] Submit with empty message → validation error

- [ ] Submit with valid data → success message
- [ ] Modal auto-closes after 2 seconds

3. Provider Detail Page - Request Care Button
- [ ] Visit `/marketplace/providers/[id]` as FAMILY user
- [ ] Verify “Request Care” button is primary CTA
- [ ] Click button → modal opens with provider name
- [ ] Submit inquiry successfully
- [ ] Lead created with targetType=PROVIDER

4. Authentication Checks
- [ ] Login as CAREGIVER user
- [ ] Visit aide detail page
- [ ] Click “Request Care”
- [ ] Error message shown: “Only family members can submit...”
- [ ] Current role displayed

5. Responsive Design
- [ ] Test modal on mobile (320px)
- [ ] Test modal on tablet (768px)
- [ ] Test modal on desktop (1024px+)
- [ ] Verify buttons and inputs are touch-friendly
- [ ] Check text readability

Database Verification

1. Lead Record Created

```sql
SELECT
  id,
  familyId,
  targetType,
  aideId,
  providerId,
  status,
  message,
  preferredStartDate,
  expectedHoursPerWeek,
  location,
  createdAt
FROM "Lead"
WHERE familyId = 'your_family_id'
ORDER BY createdAt DESC
LIMIT 5;
```

Verify:
- Status = NEW
- Correct targetType (AIDE or PROVIDER)
- Either aideId or providerId is set (not both)
- Message is stored
- Optional fields stored correctly

---

2. Audit Log Created

```
SELECT
  id,
  userId,
  action,
  resourceType,
  resourceId,
  description,
  metadata,
  createdAt
FROM "AuditLog"
WHERE resourceType = 'LEAD'
ORDER BY createdAt DESC
LIMIT 5;
```

Verify:
- Action = CREATE
- UserId matches FAMILY user
- ResourceId matches Lead id
- Metadata contains targetType, targetId, targetName, familyId

---

Integration Testing Scenarios

Scenario 1: End-to-End Aide Inquiry
1. Register as FAMILY user
2. Complete family profile setup
3. Browse marketplace → caregivers
4. Select a caregiver
5. Click “Request Care”
6. Fill inquiry form
7. Submit
8. Verify lead created in database
9. Verify audit log created
10. Check operator can view lead in /operator/leads

Scenario 2: Multiple Inquiries
1. Login as FAMILY user
2. Submit inquiry to Aide A
3. Submit inquiry to Aide B
4. Submit inquiry to Provider C
5. Verify 3 leads created
6. Verify all have status=NEW
7. Verify correct targetType for each

Scenario 3: Invalid Target Handling
1. Login as FAMILY user
2. Attempt to submit inquiry with deleted/non-existent targetId
3. Verify 404 error returned
4. Verify no lead created
5. Verify error message shown to user

Deployment

Pre-Deployment Checklist

- [x] Database migration applied (Phase 1)
- [x] API endpoint implemented and tested
- [x] UI components implemented
- [x] Detail pages updated
- [x] Authentication checks working
- [x] Validation working (client and server)
- [x] Audit logging functional
- [x] Error handling in place
- [x] Build passes without errors
- [x] Documentation complete

Environment Variables

No new environment variables required for this phase.

Database Migrations

Database schema from Phase 1 already includes Lead model. No new migrations needed for Phase 4.

Deployment Steps

1. **Merge to main branch**:
   ```
   bash
     git checkout main
     git merge feature/family-leads-mvp
   ```

2. **Push to repository**:
   ```
   bash
     git push origin main
   ```

3. **Trigger deployment** (auto-deploy on Render or manual deploy)

4. **Verify deployment**:
   - Check build logs
   - Test API endpoint in production
   - Test UI flows in production
   - Verify database connections

5. **Monitor**:
   - Check error logs
   - Monitor API response times

- Watch for failed requests
- Review audit logs

Future Enhancements

Short-term (Phase 5-6)

1. **Operator Lead Management**:
   - List view of all leads
   - Filter by status, target type, date
   - Lead detail page for operators
   - Status update functionality
   - Assignment to specific operators
   - Add internal notes

2. **Family Lead Dashboard**:
   - View submitted inquiries
   - Track inquiry status
   - Cancel inquiries
   - View responses from aides/providers

3. **Notifications**:
   - Email notification to aide/provider on new inquiry
   - Email notification to family on status change
   - In-app notifications
   - Real-time updates via websockets

Medium-term

1. **Enhanced Inquiry Form**:
   - Attach documents (e.g., care plan, medical info)
   - Schedule preference calendar view
   - Budget range selector
   - Special requirements checklist

2. **AI-Powered Matching**:
   - Suggest best aides/providers based on inquiry details
   - Compatibility scoring
   - Automated follow-up suggestions

3. **Communication Thread**:
   - Built-in messaging within lead
   - Video call scheduling
   - Interview scheduling

Long-term

1. **Analytics**:
   - Lead conversion rates
   - Response time metrics
   - Family satisfaction scores
   - Aide/Provider acceptance rates

2. **Smart Routing**:
   - Auto-assign leads to operators based on workload
   - Prioritize high-value leads
   - SLA tracking and alerts

3. **CRM Integration**:
   - Export leads to external CRM
   - Sync status updates
   - Automated workflows

Conclusion

Phase 4 implementation is complete and functional. The system now allows Family users to submit care inquiries to Aides and Providers through an intuitive UI with proper authentication, validation, and audit logging.

Next Steps: Proceed to Phase 5 (Operator Lead Management) to enable operators to view, manage, and respond to leads.

Change Log

December 7, 2025:
- ✅ Created `/src/lib/inquiry-auth.ts` - Authentication utility
- ✅ Created `/src/app/api/leads/route.ts` - POST endpoint
- ✅ Created `/src/components/marketplace/InquiryForm.tsx` - Modal component
- ✅ Created `/src/components/marketplace/RequestCareButton.tsx` - Button component
- ✅ Updated `/src/app/marketplace/caregivers/[id]/page.tsx` - Added Request Care CTA
- ✅ Updated `/src/app/marketplace/providers/[id]/page.tsx` - Added Request Care CTA
- ✅ Build verified (successful with warnings)
- ✅ Documentation created

Contact

For questions or issues related to this implementation, contact the development team or refer to the CareLinkAI project documentation.