ASSESSMENT & OPPORTUNITY IMPLEMENTATION ROADMAP

Project: Premier Insulation - Quote & Job Management System
Phase: Phase 1A (MVP)
Created: December 2025
Developer: Team

IMPLEMENTATION SEQUENCE (Order of Development)

STEP 1: Database Verification & Schema (Day 1 - 2 hours)

Task 1.1: Verify Existing Tables

```
Check in Supabase:
□ assessments table structure
□ assessment_areas table exists
□ assessment_photos table exists
□ assessment_wordings table (verify structure matches SQL above)
□ opportunities table has all required columns
□ team_members table has 'role' column with 'Installer', 'Crew Lead' values
□ clients table structure
□ companies table structure
```

Task 1.2: Create Missing Tables

```
Run SQL scripts:
□ CREATE assessment_installers table
□ CREATE job_installers table (if not exists)
□ CREATE tasks table (if not exists)
□ Add assessment_id column to opportunities table
□ Add opportunity_id column to jobs table
□ Create all indexes (see schema section above)
```

Task 1.3: Auto-Generation Functions

```
Create SQL functions or TypeScript helpers:
□ Generate assessment_number (ASS-2025-XXX format)
□ Generate opportunity_number (OPP-2025-XXX format)
□ Generate job_number (JOB-2025-XXX format)
```

Deliverable: Verified & updated database schema ready for application

---

STEP 2: Utility Functions (Day 1 - 3 hours)

Create: src/lib/assessment-helpers.ts

```typescript
Functions needed:
□ fetchAssessmentWithRelations(assessmentId)
  └ Returns assessment + areas + installers + photos + client + opportunity

□ createAssessment(data)
  └ Creates assessment record, areas, installers

□ updateAssessmentStatus(assessmentId, status)
  └ Updates status, completes workflow if "Completed"

□ getAssessmentWordings(assessmentId, areaId, resultType)
  └ Queries assessment_wordings table with filters

□ deleteAssessmentArea(areaId)
  └ Soft delete or hard delete

□ sendAssessmentLink(installerId, assessmentId, method: 'sms'|'email')
  └ Prepares SMS/email with mobile link
```

Create: src/lib/opportunity-helpers.ts

```typescript
```

```
Functions needed:
□ createOpportunity(data)
  └─ Creates opportunity record with auto-generated opp_number

□ updateOpportunityStage(opportunityId, newStage)
  └─ Updates stage and sub_status

□ fetchOpportunityWithDetails(opportunityId)
  └─ Returns opportunity + client + assessments + quotes + jobs

□ linkAssessmentToOpportunity(assessmentId, opportunityId)
  └─ Links assessment to opportunity

□ getOpportunitiesForClient(clientId)
  └─ Returns all opportunities for a client
```

Create: src/lib/task-helpers.ts

```typescript
Functions needed:
□ createTask(data: {
    taskType: string,
    assignedToUserId: string,
    relatedTo: string,
    relatedId: string,
    description: string,
    priority: string,
    dueDate: date
  })
  └─ Creates task record in tasks table

□ closeTask(taskId)
  └─ Marks task as completed
```

Deliverable: All utility functions working with proper error handling

---

STEP 3: Component Creation (Day 2 - 4 hours)

Create: src/app/components/AssessmentCard.tsx

Purpose: Reusable card for displaying assessment summary

Props:

□ assessment: Assessment object

□ showActions: boolean

□ onEdit: () => void

□ onDelete: () => void

□ onReschedule: () => void

Displays:

□ Assessment number

□ Status badge

□ Client name

□ Scheduled date/time

□ Installer(s)

□ Action buttons

Create: src/app/components/AssessmentAreaSection.tsx

Purpose: Collapsible section for each assessment area

Props:

□ area: assessment_area object

□ colorBadge: hex color

□ editable: boolean

□ wordings?: assessment_wordings[]

□ onSave: (updatedArea) => void

□ onRemove: () => void

□ photos?: assessment_photos[]

Content:

□ Collapsible header with color badge

□ Description of insulation (textarea)

□ Status radios (Pass/Fail/Exempt/Pending)

□ Description of work required (textarea)

□ Suggested wording dropdown

□ Photos gallery

□ Remove button

Create: src/app/components/AddOpportunityModal.tsx

```
Purpose: Modal for creating new opportunity
Props:
□ clientId: string (pre-filled client)
□ clientName: string
□ isOpen: boolean
□ onClose: () => void
□ onSuccess: (opportunity) => void

Fields to handle:
□ Contact details (read-only, auto-populated)
□ Opportunity name
□ Pipeline
□ Stage (default "New")
□ Sub status
□ Estimated value
□ Due date
□ Sales rep
□ Opportunity source
□ Notes
□ Status

Validation:
□ Opportunity name required
□ Estimated value required (numeric)
□ Auto-generate opp_number
```

Create: src/app/components/CrewDetailsCard.tsx

```
Purpose: Display crew assigned to assessment/job
Props:
□ crew: assessment_installers[] or job_installers[]
□ editable: boolean
□ onEdit?: () => void

Displays:
□ Primary installer with role
□ Support crew members
□ Contact info (phone, email)
□ Edit/Add buttons if editable
```

Create: src/app/components/SendDropdown.tsx

Purpose: Dropdown for Send button (Link vs Email)

Props:

□ assessmentId: string

□ clientEmail: string

□ installerPhone: string

□ onSend: (method: 'link'|'email') => void

Options:

□ Send via Link (SMS)

□ Send via Email

□ Handle appropriate delivery method

Deliverable: All components created, tested, and styled consistently

---

STEP 4: Assessment Detail Page (Day 2-3 - 5 hours)

Create/Update: src/app/assessments/[id]/page.tsx

State: SCHEDULED

Layout:

□ Header with back link, assessment #, status, action buttons

□ Three-card top row (Client, Scheduled Details, Site Info)

□ Assigned Crew section

□ Areas to assess (collapsible sections)

□ Action buttons (Save, Mark Complete, Create Task)

Functionality:

□ Fetch assessment with all relations
□ Display read-only info in SCHEDULED state
□ [Edit] button shows same form for editing
□ [Mark Complete] button:
  └ Updates status to "Completed"
  └ If opportunity linked, update opportunity stage to "QUALIFIED"
  └ Create task for sales rep
  └ Show success message
  └ Page refreshes to show COMPLETED state

□ [Send ▼] dropdown:
  └ Send via Link: Prepare SMS with unique mobile link
  └ Send via Email: Prepare email with assessment details

□ [Reassign] button opens crew selection modal

□ [Cancel] button soft-deletes assessment

State: COMPLETED

Display:

□ Full assessment report view
□ All fields read-only
□ Photos gallery populated
□ Wordings text visible
□ [Download PDF] button (Phase 1B)
□ [Email Report] button
□ [Create Quote] button (for quote workflow)

Styling:

□ Use card component for consistent look
□ Responsive grid for three cards
□ Color badges for areas
□ Collapsible sections for areas
□ Status badges with appropriate colors

Error Handling:

□ Handle 404 if assessment not found
□ Handle loading state
□ Handle save errors
□ Handle update errors gracefully

Deliverable: Assessment Detail page fully functional for both states

---

STEP 5: Schedule Assessment Page (Day 3 - 5 hours)

Create: src/app/assessments/new/page.tsx

Layout:

□ Header with back link and title
□ Single-page form with sections

□ SECTION 1: Client Selection
  └ Search existing client
  └ Auto-populate contact details

□ SECTION 2: Opportunity Source & Link
  └ Radio buttons for opportunity source
  └ Optional opportunity link

□ SECTION 3: Assessment Scheduling
  └ Date picker
  └ Time picker
  └ Assessment type dropdown
  └ Time estimate

□ SECTION 4: Crew Assignment
  └ Primary installer dropdown
  └ Add additional installers
  └ Role selection for each installer

□ SECTION 5: Site Information
  └ Property type dropdown
  └ Year built input
  └ Site access difficulty
  └ Existing insulation
  └ Removal required toggle
  └ Hazards checkboxes

□ SECTION 6: Areas to Assess
  └ Pre-populated default areas (Ceiling, Underfloor, External Walls)
  └ Editable fields for each area
  └ Remove area button
  └ Add custom area button

□ SECTION 7: Special Instructions
  └ Notes textarea with char count

□ SECTION 8: Notifications
  └ Checkboxes for email/SMS/reminder

Functionality:

☐ Fetch client details when selected from search
☐ Fetch opportunity details if linked
☐ Auto-calculate areas if possible
☐ Fetch available installers for dropdown
☐ Validate required fields before submit

[Schedule Assessment] button:
☐ Validate all required fields
☐ Create assessment record
☐ Create assessment_areas records
☐ Create assessment_installers records
☐ Update opportunity if linked
☐ Send notifications if checked
☐ Generate assessment_number
☐ Redirect to assessment detail page
☐ Show success message

Form State Management:

☐ Client selection
☐ Opportunity link
☐ Multiple area forms
☐ Multiple installer forms
☐ Form validation errors
☐ Loading/saving states

Styling:

☐ Responsive form layout
☐ Consistent input styling
☐ Collapsible sections for areas/installers
☐ Color badges for areas
☐ Clear required field indicators

Deliverable: Schedule Assessment page fully functional

STEP 6: Opportunity Detail Page Updates (Day 3-4 - 4 hours)

Update: src/app/opportunities/[id]/page.tsx

Add/Update:

□ Three-card top row with linked records card

□ NEW: ASSESSMENTS Section
  └─ List all assessments linked to this opportunity
  └─ Show assessment #, status, date, installer
  └─ [+ Schedule Assessment] button
  └─ [View] link to assessment detail
  └─ [Reschedule] button
  └─ Remove (×) button

□ NEW: CREW DETAILS Section
  └─ Only visible when job_id is set
  └─ Show all crew from job_installers
  └─ Display name, role, phone, email
  └─ Fetched from job_installers JOIN team_members

□ EXISTING: Notes section remains

Functionality:

□ Fetch opportunity with all related data
□ Fetch all assessments for this opportunity
□ Fetch job if created (for crew details)
□ Fetch job_installers for crew display

[Schedule Assessment] button:
└─ Navigate to /assessments/new?opportunity_id=this.id
└─ Pre-populate client info
└─ Pre-select this opportunity

[View Details] link:
└─ Navigate to /assessments/[id]

[Reschedule] button:
└─ Open edit form for assessment

Remove (×) button:
└─ Soft delete assessment
└─ Remove from opportunity

Styling:

- □ Card-based layout for assessments list
- □ Crew details in read-only card format
- □ Status badges for assessments
- □ Action buttons consistent with design system

Deliverable: Opportunity Detail page updated with Assessment & Crew sections

---

STEP 7: Add Opportunity Modal (Day 4 - 3 hours)

Create/Update Modal in Client Detail Page

Add button in `/customers/[id]/page.tsx`:

- □ [+ New Opportunity] button visible on client detail page
- □ Opens AddOpportunityModal
- □ Passes clientId, clientName, email, phone, company, address

Modal Component (in AssessmentOpportunityModal.tsx or AddOpportunityModal.tsx):

Props:
☐ isOpen: boolean
☐ onClose: () => void
☐ clientId: string
☐ clientName: string
☐ onSuccess: (opportunity) => void

Content:
☐ Heading: "Create a new sales opportunity"
☐ Contact details (read-only, auto-populated)
☐ Opportunity details form
☐ Action buttons: [Cancel] [Create Opportunity]

Form Fields:
☐ Opportunity Name *
☐ Pipeline (default "Sales Pipeline")
☐ Stage (default "New")
☐ Sub Status (default "Awaiting Contact")
☐ Estimated Value (NZD) *
☐ Due Date
☐ Sales Rep
☐ Opportunity Source
☐ Notes
☐ Status

Functionality:
☐ Validate required fields
☐ Generate opp_number (OPP-2025-XXX)
☐ Create opportunity record
☐ Set created_by_user_id = current user
☐ Close modal on success
☐ Show success message
☐ Redirect to opportunity detail page OR
☐ Refresh client detail page with new opportunity listed

Styling:

☐ Modal wrapper with overlay
☐ Form-styled inputs
☐ Consistent button styling
☐ Clear validation errors
☐ Loading state during submit

Deliverable: Add Opportunity modal fully functional

STEP 8: Testing & Integration (Day 5 - 3 hours)

End-to-End Workflow Testing:

Test Flow 1: Complete Workflow
□ Create new client in Contacts
□ Create new opportunity from client detail page
□ Schedule assessment from opportunity detail page
□ Verify assessment scheduled with correct details
□ Complete assessment (Mark Complete button)
□ Verify opportunity stage updated to "QUALIFIED"
□ Verify task created for sales rep
□ View opportunity and see assessment listed
□ (Later) Create recommendation from assessment
□ (Later) Create quote from recommendation
□ (Later) Create job and assign crew
□ Verify crew visible in opportunity detail page

Test Flow 2: Assessment Without Opportunity
□ Schedule assessment without linking opportunity
□ Verify assessment can be created independently
□ Verify no opportunity updates occur
□ Can link to opportunity later

Test Flow 3: Multiple Assessments per Opportunity
□ Create opportunity
□ Schedule first assessment
□ Schedule second assessment
□ Verify both appear in opportunity detail page
□ Complete first assessment only
□ Verify opportunity still shows both assessments

Test Flow 4: Send Assessment
□ Schedule assessment
□ Click Send dropdown
□ Test "Send via Link" (verify SMS would be sent)
□ Test "Send via Email" (verify email would be sent)

Test Flow 5: Edit Assessment
□ Schedule assessment
□ Click Edit button
□ Modify areas, crew, notes
□ Save changes
□ Verify changes persist
□ In SCHEDULED state - allow editing

Test Flow 6: Assessment Wordings

- □ Complete assessment
- □ View detail page in COMPLETED state
- □ Verify assessment_wordings dropdown populated
- □ Select different wording
- □ Verify text auto-populates
- □ Verify correct filtering by area_id, result_type

Responsive Design Testing:

- □ Test on mobile (375px width)
- □ Test on tablet (768px width)
- □ Test on desktop (1920px width)
- □ Verify cards stack correctly
- □ Verify dropdowns usable on mobile
- □ Verify buttons touch-target size (44×44px minimum)
- □ Verify text readable at all sizes
- □ Verify forms not overcrowded on mobile

Error Handling Testing:

- □ Test missing required fields
- □ Test database errors
- □ Test network failures
- □ Test 404 errors
- □ Verify error messages user-friendly
- □ Verify loading states visible

Accessibility Testing:

- □ Verify color contrast (WCAG AA)
- □ Verify form labels associated with inputs
- □ Verify keyboard navigation works
- □ Verify screen reader compatible (basic)

Deliverable: All tests passing, workflows verified

DAILY BREAKDOWN

Day 1 (8 hours)

- **2 hours:** Database verification & schema
- **3 hours:** Create utility functions
- **3 hours:** Create reusable components

Day 2 (8 hours)

- **5 hours:** Assessment Detail page
- **3 hours:** Start Schedule Assessment page

Day 3 (8 hours)

- **5 hours:** Complete Schedule Assessment page
- **3 hours:** Opportunity Detail page updates

Day 4 (8 hours)

- **4 hours:** Add Opportunity modal
- **4 hours:** Testing & bug fixes

Day 5 (4 hours)

- **4 hours:** Final testing, responsive design, polish

Total: 36 hours (4.5 days)

DEPENDENCIES & BLOCKERS

Must Be Ready Before Implementation:

```
□ Supabase project accessible
□ GitHub repository set up
□ Team member table populated with at least one installer
□ Client records exist in database
□ All required tables in Supabase schema
□ Tailwind CSS configured in project
□ Next.js 14 with App Router working
□ TypeScript configured correctly
```

Optional (Can be Phase 1B):

- ☐ Authentication/RBAC (can show all assessments in Phase 1A)
- ☐ Email notifications via Resend
- ☐ SMS notifications via Twilio
- ☐ PDF generation
- ☐ Advanced calendar with drag-drop
- ☐ Mobile app

CODE REVIEW CHECKLIST

Before marking as complete:

- ☐ All TypeScript types defined
- ☐ No eslint warnings
- ☐ No unused variables
- ☐ Error handling for all async operations
- ☐ Loading states visible to user
- ☐ Form validation working correctly
- ☐ Database queries optimized (proper indexes)
- ☐ No hardcoded values
- ☐ Comments for complex logic
- ☐ Responsive design tested
- ☐ Accessibility standards met
- ☐ Security: No SQL injection vulnerabilities
- ☐ Performance: Queries complete in <1s
- ☐ Code style consistent with project
- ☐ Git history clean (meaningful commits)

HANDOFF TO PHASE 1B

When Phase 1A complete, these become Phase 1B priorities:

□ Authentication with Supabase Auth
□ Role-based access control (VA, Premier, Admin)
□ Email notifications (Resend integration)
□ SMS notifications (Twilio integration)
□ PDF report generation & download
□ Mobile installer job completion (existing form)
□ Advanced calendar with scheduling
□ Dashboard with metrics & widgets
□ Settings page
□ Audit logging
□ More comprehensive error handling
□ Performance optimization

Questions or clarifications needed before starting development?